

TDK-dolgozat

Matuszka Tamás



Eötvös Loránd Tudományegyetem
Informatikai Kar
Média- és Oktatásinformatika Tanszék

Kiterjesztett valóság alkalmazások fejlesztése, elemzése és a fejlesztőeszközök összehasonlítása

Szerző:

Matuszka Tamás

Programtervező informatikus MSc hallgató
2. évfolyam

Témavezető:

Dr. Turcsányiné Szabó Márta

Egyetemi docens

Budapest, 2012.

A projekt az Európai Unió támogatásával és az Európai Szociális Alap társfinanszírozásával
valósul meg, a támogatási szerződés száma TÁMOP [4.2.1./B-09/1/KMR-2010-0003](#).

Tartalomjegyzék

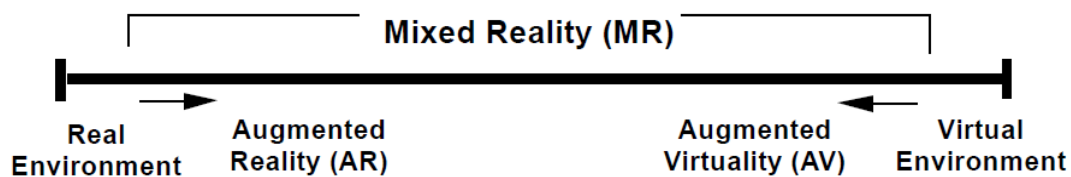
KIVONAT	4
1. BEVEZETÉS.....	5
2. KITERJESZTETT VALÓSÁG	8
2.1. TÖRTÉNETI ÁTTEKINTÉS.....	8
2.2. HÁROM DIMENZIÓBAN VALÓ REGISZTRÁCIÓ	13
2.3. ALKALMAZÁSI TERÜLETEK	20
2.3.1. Egészségügy.....	21
2.3.2. Gyártás és javítás	22
2.3.3. Szórakoztatás.....	23
2.3.4. Oktatás.....	24
2.3.5. Játék.....	25
3. FEJLESZTŐESZKÖZÖK.....	26
3.1. FEJLESZTŐESZKÖZÖK ASZTALI ALKALMAZÁSOKHOZ	26
3.1.1. ARToolKit.....	26
3.1.2. NyARToolKit.....	27
3.1.3. Goblin XNA	28
3.1.4. FLARToolKit és FLARManager	28
3.1.5. IN2AR	30
3.2. FEJLESZTŐESZKÖZÖK MOBIL ALKALMAZÁSOKHOZ	31
3.2.1. AndAR.....	31
3.2.2. Qualcomm SDK	31
3.3. ÖSSZEHASONLÍTÁS.....	32
4. SAJÁT FEJLESZTÉSEK	33
4.1. AR ELTE IK.....	33
4.2. AR PÓLÓ.....	35
4.3. AR KÉMIA.....	36
4.4. MASZK	37
4.5. HANGOSKÖNYV.....	39
5. FEJLESZTŐESZKÖZÖK ÖSSZEHASONLÍTÁSA	40
5.1. FELHASZNÁLT METRIKÁK	40
5.1.1. Méret metrikák.....	41
5.1.2. Komplexitás metrikák	42
5.2. EGYÉB MÉRÉSEK	45
5.2.1. Távolság	46
5.2.2. Markerek bonyolultsága	47
5.2.3. Stabilitás.....	48
5.2.4. Összegzés.....	48
6. JÖVŐBELI TERVEK	50
7. KONKLÚZIÓ	51
IRODALOMJEGYZÉK.....	52

Kivonat

A kiterjesztett valóság (Augmented Reality, AR) az informatika egyik jelenleg is dinamikusan fejlődő ága, mely napjainkban egyre szélesebb körben kerül a hétköznapi életben is alkalmazásra. Segítségével a fizikai világ valós időben kibővíthető számítógép által generált virtuális elemekkel, azt az illúziót keltve, hogy ezek a virtuális elemek (amik lehetnek például 3 dimenziós modellek, videók, vagy animációk) beleolvadnak a valós környezetbe. Az így kapott rendszert, amely a valós és a virtuális világ között helyezkedik el, általában számítógép vagy mobiltelefon kijelzőjén, esetleg fejre illeszthető kijelzőn (Head Mounted Display) tekinthetjük meg. Dolgozatom első részében bemutatom a kiterjesztett valóság történeti hátterét és gyakorlati alkalmazási területeit megvalósított példákon keresztül, valamint a három dimenzióban való regisztráció (a valós és virtuális objektumok egymáshoz való igazítása) matematikai hátterét. A második részben ismertetem az általam feltérképezett kiterjesztett valóság fejlesztőeszközök jellemzőit. A harmadik részben azon saját alkalmazások leírása olvasható, amelyeket korábbi projektjeim során, valamint jelen dolgozathoz készítettem. A negyedik részben a különböző fejlesztőeszközökkel elkészített azonos funkcionalitású programok szoftvermetrikai méréseinek, és egyéb általam kitervelt mérések általi összehasonlítás eredményeit részletezem. A kapott eredményekből megismerhetjük a fejlesztőeszközök erősségeit és hiányosságait, valamint segítenek a programozóknak az elkészítendő alkalmazás függvényében annak eldöntésében, hogy milyen környezetet válasszanak ki a leendő kiterjesztett valóság szoftverük elkészítéséhez.

1. Bevezetés

A kiterjesztett valóság (AR) napjaink egy viszonylag új, egyre jobban elterjedő technológiája, melynek segítségével a valódi fizikai környezet kibővíthető számítógép által generált virtuális elemekkel. Ilyen virtuális elem lehet például egy 3 dimenziós modell, egy animáció, vagy egy videó. Több különböző definíció is létezik, azonban a legáltalánosabban elfogadott Ronald T. Azuma nevéhez köthető, mely szerint a kiterjesztett valóság valós időben, interaktívan ötvözi a valós és virtuális világot, mindezt 3D-ben regisztrálva [2]. Az így kapott rendszer a valós és virtuális világ között helyezkedik el.



1. ábra - A kiterjesztett valóság elhelyezkedése a valós és virtuális környezet között [1].

A kiterjesztett valóság által alkotott világ megjelenítése különféle módon, különféle eszközökkel történhet. A leggyakoribb mód egy monitoron vagy egy telefon kijelzőjén történő megjelenítés, de bizonyos esetekben fején hordható kijelzőt (Head Mounted Display) is használnak erre a célra. A virtuális elemek helyzete szintén többféle feltételhez köthető. Ilyen feltételek lehetnek például az eszköz GPS koordinátái, az iránytűjének helyzete, gyorsulásmérőjének adatai vagy akár kamera képén található jellegzetes képrészletek [3]. A kiterjesztett valóság kétféle típusát különböztethetjük meg a megjelenítéstől és a virtuális elem helyzetét meghatározó eljárástól függően.

Az első ilyen típus a *pozíció és irány alapú AR*, melyet elsősorban mobiltelefonokon alkalmaznak. Célja, hogy a kijelzőn megjelenő valós képet új információkkal terjessze ki. A plusz információkat a POI-k hordozzák. A POI (Point of Interest) különböző helymeghatározó programok által használt kifejezés. A POI olyan pontok helyét adja meg, amely valaki számára érdekes, hasznos lehet¹. Ezen pontok információi közt megtalálhatók az általános információkon túl (utca, házszám, telefonszám stb.) a földrajzi koordinátái is. A valós képet kiterjesztő információ helyének meghatározása iránytű, a GPS pozíció, valamint gyorsulásérzékelő segítségével történik. „A GPS pozíció meghatározza, hogy pontosan hol található a mobil eszköz, tehát ez alapján számolható ki, hogy melyik POI milyen messze van

¹ Forrás: Wikipedia, http://en.wikipedia.org/wiki/Point_of_interest

az aktuális pozíciótól. Az iránytű megmondja, hogy merre néz a telefon, melyből kiszámítható, hogy egy POI benne van-e a kamera által látott képben a földdel párhuzamos síkon található irányvektort tekintve. A gyorsulásérzékelő pedig megmondja, hogy a telefonnak mekkora a dőlésszöge a vízszinteshez képest, tehát a földre merőleges síkban található irányvektor alapján a kamera képe tartalmazza-e a POI-t” [3]. Jelenleg több különböző ilyen alkalmazás létezik, mint például a Wikitude², a Layar³, valamint a Junaio⁴.



2. ábra: A pozíció és irány alapú AR okostelefonon való megjelenése Wikitude segítségével.

A kiterjesztett valóság másik típusa a *marker alapú AR*. Ilyenkor egy speciális képrészletet keresünk, amely kitűnik a környezetéből, így könnyen kereshető. Érdeemes redundáns, hibátűrő kódolást alkalmazni, ugyanis így kisebb valószínűséggel fogja az eljárás hibásan detektálni a markert. Ezen kritériumoknak megfelelően a marker egy fehér alapon fekete négyzet, bizonyos vastagságú szegéllyel, amelyen belül egy egyedi, fekete-fehér képrészlet található. A marker pozíciója és helye meghatározható a kamerához képest, így ráhelyezhetünk tetszőleges virtuális objektumot. A markerek felismeréséhez első lépésként éldetektálásra van szükség. Ezt követően meg kell keresni a kontúrokat, melyek sokszögekké alakíthatóak, majd a kapott sokszögeket szűrni kell a négyszögekre. Miután megvannak a négyszögek, mintát kell venni az élek melletti külső és belső pontokból, hiszen ebből szűrhetőek ki a lehetséges markerek. Az éleken kívül, adott távolságra világos pontoknak, míg az éleken belül, adott távolságra sötét pontoknak kell lenniük. Ilyenkor nagy valószínűséggel már csak a tényleges markerek maradnak meg a lehetséges elemek halmazában. [3]. Ezek a feladatok megoldhatóak az OpenCV⁵ képfeldolgozó könyvtár segítségével. Az újabb típusú kiterjesztett valóság fejlesztőeszközöknek már nincs szükségük fekete keretes markerekre sem a megjelenítendő virtuális elem pozíciójának kiszámításához, hanem tetszőleges kép is alkalmazható erre a feladatra. Minél kontrasztosabb, részletesebb a

² <http://www.wikitude.org>

³ <http://www.layar.com/>

⁴ <http://www.junaio.com/>

⁵ <http://opencv.willowgarage.com/wiki/>

kép, annál jobban felismerhető. Sikeres detektálás után a marker helyére illeszthető a virtuális elem, melyre a marker mozgásából kiszámíthatóak az elvégzendő transzformációk, így ötvözve a képernyőn a virtuális valóságot az igazi valósággal [4].

A kiterjesztett valóság egyre inkább megjelenik hétköznapijainkban, néhány gyakorlati alkalmazás:

- oktatás (http://www.learnar.org/bio_organs_demo.html, AR Kémia [4])
- szerelés (<http://www.youtube.com/watch?v=P9KPJIA5yds>)
- reklám
(http://www.youtube.com/watch?v=RnN6s0xfMvs&feature=player_embedded#at=40)
- sport (<http://www.youtube.com/watch?v=DnmXT6x85p8&feature=fvst>)
- játék (<http://www.youtube.com/watch?v=Lfp8id6bpDU>)
- marketing (<http://www.youtube.com/watch?v=NxQZuo6pFUw>)

A dolgozatom bemutatja a kiterjesztett valóság történetét, fejlődését, és példákon keresztül a hétköznapi életben való alkalmazását, valamint a marker regisztrációjának matematikai hátterét. Különböző fejlesztőkörnyezetekkel ismerkedhetünk meg ezek után, amelyekkel C++, C#, Java vagy ActionScript nyelven készíthetünk el különböző alkalmazásokat, melyek képesek lesznek egyidejűleg több marker kezelésére, reagálhatnak a felhasználói interakciókra. Ezt követően leírom az általam elkészített alkalmazások részleteit. A dolgozat végén bemutatok néhány szoftvermetrikát, és egy –négy kiválasztott fejlesztőeszközzel elkészített– program elemzésén keresztül összehasonlítom ezen eszközök teljesítményét, valamint ismertetem a kapott eredményeimet.

2. Kiterjesztett valóság

2.1. Történeti áttekintés

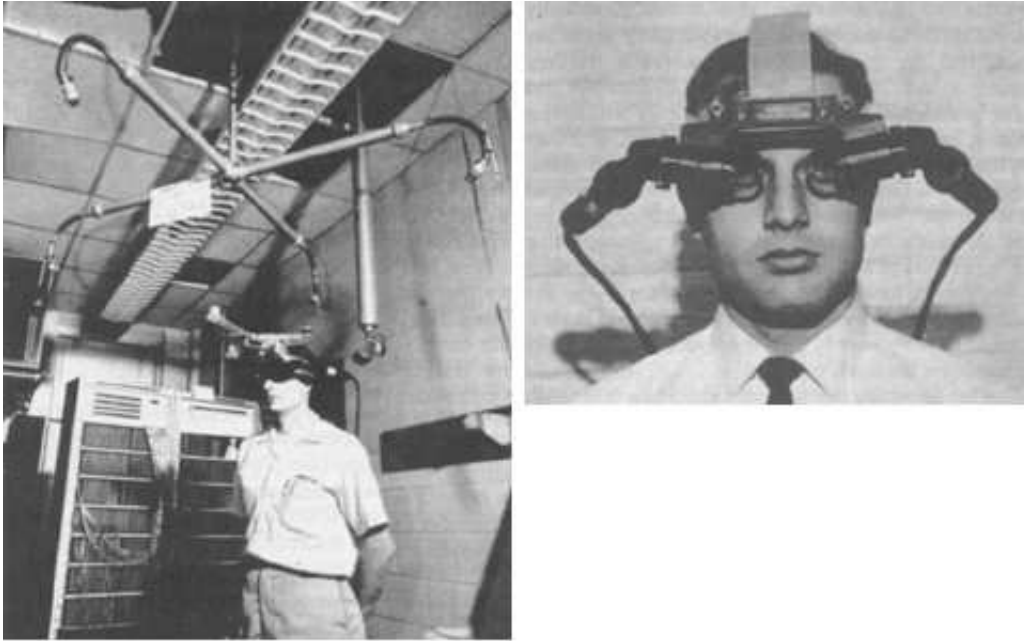
Az első kiterjesztett valósághoz köthető esemény 1957-ben történt, amikor a virtuális valóság atyjaként is emlegetett Morton Heilig amerikai operatőr megalkotta a Sensorama szimulátort⁶. Segítségével a kipróbálója egy motor üléséből járhatta be az 1950-es évekbeli Brooklyn-t, láthatta a tájat és hallhatta a város hangjait. A valóság illúzióját 3D-s mozgóképpel keltette, valamint sztereó hangot és rezgő ülést használt a még élethűbb hatás érdekében. Találmánya 1962-ben kapta meg az amerikai szabadalmat.



3. ábra - A Sensorama szimulátor.

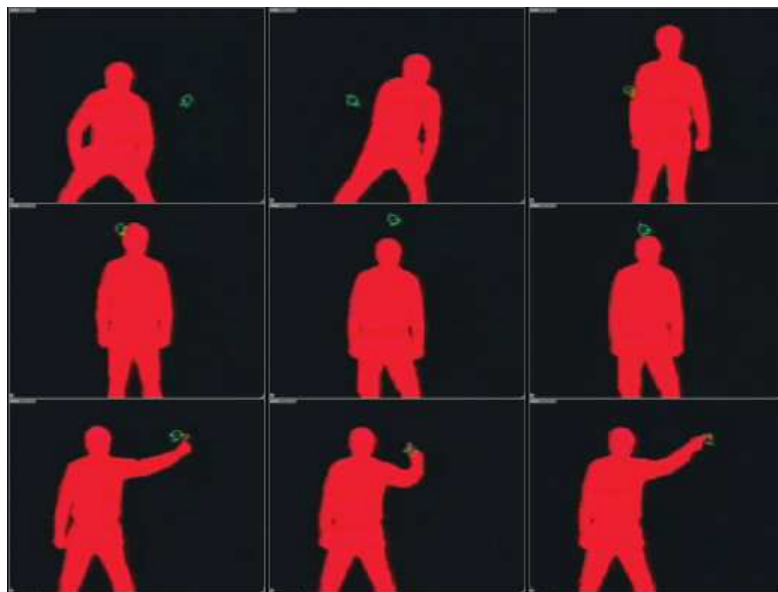
A következő mérföldkőhöz 1966-ban érkezünk el. Ekkor készíti el Ivan Sutherland az első fejre illeszthető kijelzőt, amellyel, mint egy ablakon keresztül tekinthetünk a virtuális valóságba [5]. Az eszköz egy számítógéphez csatlakozott. A fejre illeszthető kijelzőn a felhasználó egy kocka hálóját tekinthette meg a kezdetekben. A sisak fémrudakon lógott a mennyezetről. A rudak több célt is szolgáltak, egyrészt ezekkel határozták meg a fej térbeli helyzetét, másrészt a szerkezet nagy súlyának megtartására is szolgáltak. Később Sutherland továbbfejlesztette találmányát.

⁶ <http://www.sensorama3d.com/sensorama2.html>



4. ábra - Ivan Sutherland által elkészített fejre illeszthető kijelző.

Az újabb említésre méltó eseményig 1975-ig kellett várni. Ekkor készítette el Myron Krueger a Videoplace interaktív környezetet. A környezet a felhasználók mozdulataival vezérelt interaktív videoinstallációk sorozatából állt. A felhasználók egy hatalmas kijelző előtt álltak, amelyen az árnyékuk jelent meg, és a rendszer a körvonalait használta fel egy kamera segítségével. A környezet által nyújtott Critter művelet egy virtuális lény működését utánozza, amelyen keresztül a felhasználók interakcióba léphettek egy virtuális objektummal. A virtuális lény a felhasználót elkerüli először, majd kiismeri mozgását, majd ha a felhasználó befejezi a mozgást, megközelíti őt, a kontúrjain keresztül megkerüli, majd megsemmisíti magát [6].



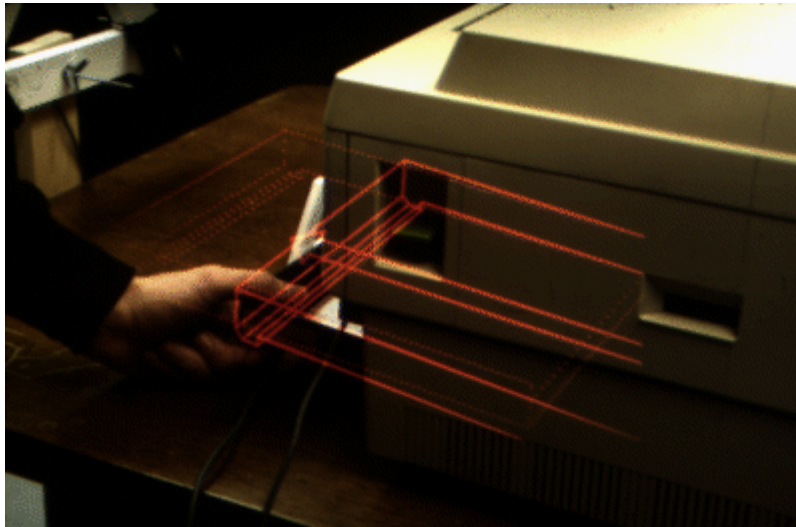
5. ábra - Videoplace használat közben, a felhasználó interakcióba lép a zöld virtuális objektummal.

1989-ben Jaron Lanier bevezette a virtuális valóság fogalmát, majd 1990-ben a köztudatba kerül a kiterjesztett valóság fogalma is Tom Caudell által. Ekkor kérte fel a Boeing Caudell-t és David Mizelt, hogy készítsenek el egy alkalmazást, amely segít a munkásoknak a repülőgépek kábeleinek összeszerelésében. Feladatuk kapták egy költséghatékony alternatíva ajánlását és elkészítését az akkoriban használt összeszerelési útmutatóul szolgáló drága ábrák és jelölési eszközök helyett. A két kutató az egyénileg tervezett utasításokat tartalmazó táblák helyett egy fejre illeszthető kijelző használatát javasolta, amely rávetíti a repülőre az eszközspecifikus utasításokat, a beszerelendő kábeleket virtuálisan a helyükre vetítve. Így nincs szükség minden egyes új folyamatnál egy új használati utasítás elkészítésére, hanem gyorsan és hatékonyan tud a dolgozó egy számítógépes rendszer által adott utasításokat végrehajtani.



6. ábra - A Boeing-nél elkészített prototípus használat közben.

1992-ben három kutató, Steven Feiner, Blair MacIntyre és Dorée Seligmann elkészíti az első jelentős cikket a KARMA-nak (Knowledge-based Augmented Reality for Maintenance Assistance) nevezett kiterjesztett valóság rendszer prototípusáról, amelyet a Graphics Interface konferencián adtak elő. A széles körben idézett publikáció [7] 1993-ban jelent meg az ACM közleményeiben. A KARMA egy fejre illeszthető kijelzőt használó alkalmazás, amely lehetővé teszi az egyszerű végfelhasználónak egy lézernyomtató karbantartását.



7. ábra - A KARMA rendszer jelen esetben megmutatja a felhasználónak a papírtálca eltávolítását.

A kiterjesztett valóság iparon kívüli alkalmazhatóságát bizonyítja az 1994-ben Julie Martin által létrehozott első AR színházi produkció. A *Dancing In Cyberspace* című előadást az ausztráliai kormány finanszírozta. A táncosok és akrobaták műsorszámait ember méretű virtuális objektumok manipulálásával adták elő, amit valós időben vetítettek a fizikai térbe.

1998-ban Ramesh Raskar, Greg Welch és Henry Fuchs kifejleszti az Észak-Karolinai Egyetemen a Spatial Augmented Reality-t. Az új megközelítésük szerint a virtuális elemek közvetlenül a felhasználó fizikai környezetében jelennek meg. A korábbi alkalmazásokhoz HMD-re volt szükség, ehelyett térbeli kijelzők, széles látómező és nagyfelbontású képek segítségével lehet beépíteni a virtuális objektumokat a közvetlen környezetbe. Mivel nincs szükség minden egyes felhasználónak fejre illeszthető kijelzőre, ezért a felhasználók csoportjai között lehetséges az interakció. Hátránya azonban, hogy a kijelzők a fényviszonyok változására érzékenyebbek, valamint szükség van nagyméretű kijelzőkre a számítógép generálta virtuális elemek megjelenítéséhez.



8. ábra - Élő panorámakép a Spatial Augmented Reality alkalmazása közben.

1999-ben Hirokazu Kato és Mark Billinghurst kifejleszti az ARToolKit-et, amely egy hat szabadságfokú, markereket használó nyomkövető könyvtár. Az ARToolKit nyílt forráskódú, és a mai napig népszerű az AR közösségben. A fejlesztőeszközök bemutatásánál részletesebben is olvashatunk róla.

A XX. század első kiterjesztett valósággal kapcsolatos jelentős története a 2000-ben Bruce H. Thomas által kifejlesztett ARQuake, amely az első szabadtéri mobil AR játék, ami a népszerű Quake játékon alapszik. A program az ARToolKit segítségével készült, így markereket használnak a virtuális objektumok megjelenítéséhez. A játékhoz szükség van egy hordozható számítógépes platformra. Thomas a fejlesztését az International Symposium on Wearable Computers konferencián mutatta be.



9. ábra - Az ARQuake hordozható számítógépes platformja és a megjelenített kép.

Mathias Möhring vezetésével 2004-ben kifejlesztettek egy 3D-s markerek nyomkövetésére képes rendszert mobiltelefonokra. Ez volt az első fogyasztói mobiltelefonokra kifejlesztett videó képét kiterjesztő AR alkalmazás. Támogatja a különböző 3 dimenziós markerek nyomkövetését, és a 3D-s grafikák helyes integrálását az élő videó képébe.



10. ábra - 3D-s markerek nyomkövetésére alkalmas szoftver mobiltelefonra.

2008: október 20-án elindul a Wikitude AR Travel Guide, amely a bevezetőben említett pozíció és irány alapú AR egyik alkalmazása. A program a GPS koordináták és egy adatbázis alapján meg tudja mutatni a környék nevezetességeit, kulturális pontjait. Ezen kívül a felhasználók megoszthatják saját megjegyzéseiket a Wikipedia-hoz hasonlóan.

A legutolsó esemény, amelyet megemlétek, 2009-ben következett be. Ekkor egy Saqoosha nevű fejlesztő elkészítette az ARToolKit Adobe Flash-re (FLARToolKit) portolt változatát, amelynek köszönhetően a kiterjesztett valóság elérhetővé vált a webböngészők számára is. A FLARToolKit-ről az ARToolKit-hez hasonlóan részletesebben tájékozódhatunk a fejlesztőeszközök bemutatásánál, valamint a programok leírásánál, mivel több saját programomhoz is ezt az eszközt használtam.

2.2. Három dimenzióban való regisztráció

A kiterjesztett valóság egyik legnagyobb kihívása a 3D-s regisztráció, azaz a virtuális objektum megfelelő módon való leképezése a valós világra. A kezdetleges AR alkalmazások mágneses nyomkövető rendszerrel próbálták megoldani ezt a feladatot, azonban ez a módszer nagy mennyiségű hibát tartalmazott. Egy kalibrálatlan rendszer akár több mint 10 cm-t is tévedhetett, különösen, ha a mágneses mezőt zavaró tényezőknek (például fém-, és elektromos berendezések) volt kitéve. Más rendszerek mechanikus nyomkövetést alkalmaztak, amely pontosabb volt ugyan a mágneses nyomkövetésnél, de alkalmazása jóval nehezebb volt.

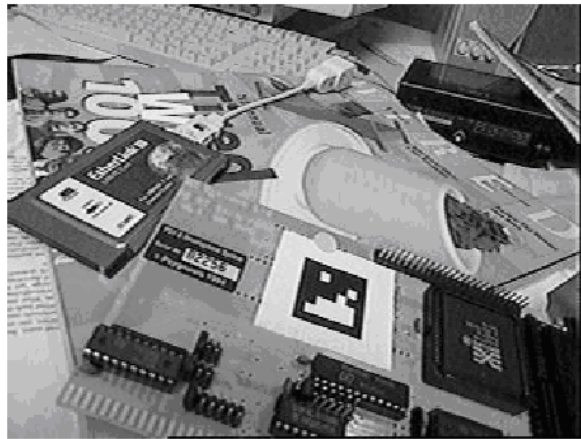
Az 1996-ban State és társai [8] által írt cikk megjelenésekor a meglévő nyomkövető módszerek külön-külön nem voltak kielégítőek: a mágneses nyomkövetés pontatlan, a

mechanikus nehézkes, míg a látásalapú nyomkövetés számításigényes volt. Ezen okok miatt kifejlesztettek egy hibrid megoldást, amely a látás alapú nyomkövetés pontosságával és a mágneses nyomkövetők robusztusságával bír. A látás alapú nyomkövetéshez színkódokkal ellátott tereptárgyakat alkalmaztak, amelyből meg tudták állapítani a kamera pozícióját és irányát. A folyamat segítésére mágneses nyomkövetőket is alkalmaztak.

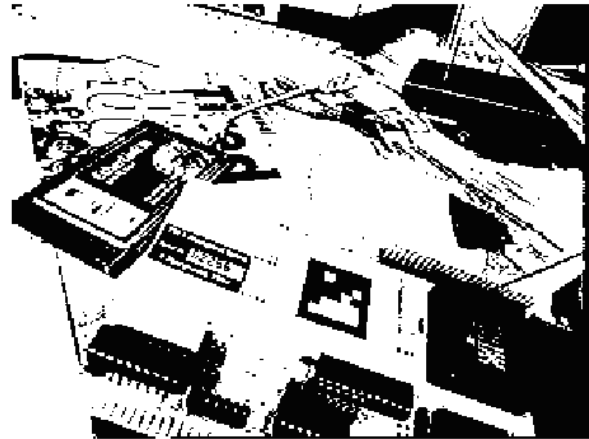


11. ábra - Színkódokkal ellátott tereptárgyak a kamera pozíciójának azonosításához.

Rekimoto 1996-ban kidolgozott megoldása [9] kétdimenziós vonalkódokat használt az alkalmazás regisztrációjához, amely segítségével nagyszámú, 2^{16} darab különböző objektumot képes azonosítani. A rendszer rugalmas és hordozható, ugyanis a vonalkódok egyszerűen kinyomtathatóak. A vonalkódok felismerése öt lépésen keresztül történt. Először is binarizálni kell a kamera képét, amely során fekete-fehérré alakítjuk át a képet. Ezt a legegyszerűbb módon egy küszöbérték választásával érhetjük el, mégpedig úgy, hogy a küszöbérték alatti pixelértékek esetén fehér lesz az adott pixel, ellenkező esetben fekete. A küszöbérték megtalálása bizonyos esetekben nagyon nehéz, jelen esetben azonban egy fix küszöbértéket használnak, amely megfelelően működik a vonalkód magas kontrasztjának köszönhetően. Ezt követően az összefüggő komponensek elemzése során a rendszer megtalálja az összefüggő fekete pixeleket (mivel a binarizálás következtében fekete és fehér pixelek alkotják a képet). Miután ez megtörtént, a vonalkód keretének meghatározása a következő szükséges művelet, majd a dekódolás és hibaellenőrzés után már megtalálható a kamera pozíciója.



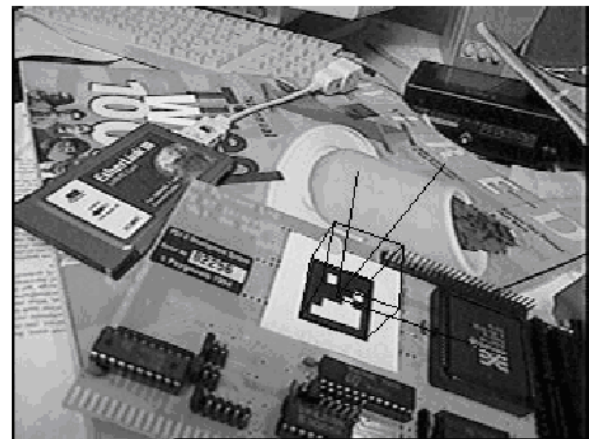
Eredeti kép



Binarizálás



Összefüggő komponensek elemzése

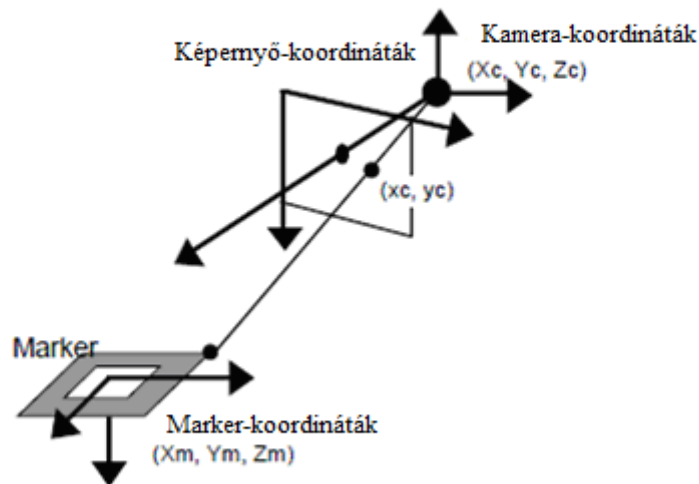


Koordináta-rendszer meghatározása

12. ábra - Rekimoto rendszerének négy lépése.

Neumann és társai [10] két eljárást dolgoztak ki 1999-ben a regisztráció problémájának megoldására. Az első módszer színes régiók detektálásán és szegmentálásán alapul. A felismerés egyszerű, de sok paraméter befolyásolhatja a vonatkoztatási pontok detektálását, mint például a kamerafelbontás, színminőség, valamint a megvilágítás. A másik megközelítés a fuzzy detektálás, amely a szomszédos pixelcsoportok alapján szegmentálja a vonatkoztatási pontokat és a háttérrel. Előnye, hogy nincs szükség küszöbértékre a kalibrálás során, valamint a kapcsolatok stabilak maradnak a fény-, és színváltozások esetén is.

A korábban már említett ARToolKit-et 1999-ben fejlesztette ki Hirokazu Kato és Mark Billinghurst [11], és mivel a későbbiekben még bővebben is lesz róla szó, valamint felhasználom az alkalmazásfejlesztésben, ezért részletesebben ismertetem az általuk kifejlesztett regisztráció matematikai hátterét. Három koordináta-rendszert használtak, a marker-koordinátarendszert, a kamera-koordinátarendszert, valamint a képernyő-koordinátarendszert.



13. ábra - Az ARToolKit által használt koordináta-rendszerek.

A feladat az ismert méretű négyzet alakú markerek koordináta-rendszerét áttranszformálni a kamera-koordináta-rendszerbe. A marker-koordináták kamera-koordinátákba átvivő T_{cm} transzformációs mátrixát, amelyet az 1. egyenleten láthatunk, a képelemzés során közelítő algoritmussal határozták meg.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & V_{13} & W_x \\ V_{21} & V_{22} & V_{23} & W_y \\ V_{31} & V_{32} & V_{33} & W_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} V_{3 \times 3} & W_{3 \times 1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = T_{cm} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}$$

1. Egyenlet

A bemeneti kép binarizálása után kinyerték azon területeket, amelyeket négy vonalszegmensen négyzetbe tudtak foglalni. A vonalszegmensek paramétereit, és a négy metszéspont koordinátáit tárolták későbbi folyamatokhoz. A területeket normalizálták, és a területen belüli részképet összehasonlították sablonillesztéssel a rendszerben tárolt mintákkal, mielőtt azonosították volna az egyedi felhasználói markert. A felhasználó neve, vagy egy fénykép lehet például egy azonosítható minta.

A markerhez tartozó négyzet két párhuzamos vonalszegmensének egyenletét a következőképp írhatjuk fel:

$$a_1x + b_1y + c_1 = 0, \quad a_2x + b_2y + c_2 = 0$$

2. Egyenlet

Minden egyes markerre e paraméterek értékeit már a vonalillesztés során megszerezték. A kamerakalibráció során megszerzett 3. egyenleten látható P perspektív projekciós mátrixot⁷ és a 2. egyenletet felhasználva a megfelelő helyettesítések és átrendezések után megkaphatjuk a 4. egyenleten látható, a vonalszegmenseket tartalmazó síkok egyenletét.

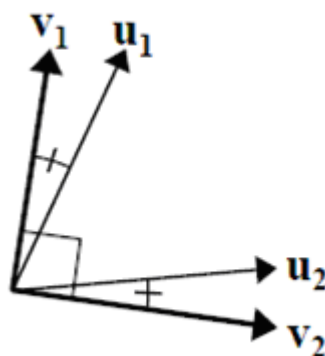
$$\mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & 0 \\ 0 & P_{22} & P_{23} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} hx_c \\ hy_c \\ h \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

3. Egyenlet

$$\begin{aligned} a_1P_{11}X_c + (a_1P_{12} + b_1P_{22})Y_c + (a_1P_{13} + b_1P_{23} + c_1)Z_c &= 0 \\ a_2P_{11}X_c + (a_2P_{12} + b_2P_{22})Y_c + (a_2P_{13} + b_2P_{23} + c_2)Z_c &= 0 \end{aligned}$$

4. Egyenlet

A négyzet két párhuzamos oldalából megkapható a két egységvektor, az u_1 és u_2 , amelyeknek merőlegeseknek kell lenniük egymásra. Azonban a képfeldolgozási hibáknak köszönhetően ezek a vektorok nem lesznek merőlegesek egymásra. Ennek kompenzálására definiálták a v_1 és v_2 egymásra merőleges vektorokat a síkban, amelyek befoglalják az u_1 , u_2 vektorokat, amint az a 14. ábrán látható.



14. ábra - Az u_1 és u_2 vektorból számított merőleges egységvektorok.

Tekintettel arra, hogy a v_3 egységvektor az, amely merőleges mind a v_1 -re és v_2 -re, ezért az 1. egyenletben szereplő T_{cm} transzformációs mátrix $V_{3 \times 3}$ -as forgatási komponense ezek

⁷ A 3D-s koordináták 2D-s koordinátákká alakító dimenziócsökkentő műveletet végzi el.

alapján meghatározható. Mivel a $V_{3 \times 3}$ már adott, a fent szereplő egyenletrendszerek segítségével a W_x , W_y , W_z eltolási komponens értéke is meghatározható, amely után már ismertté válik a T_{cm} mátrix.

Martin-Gutiérrez és társai 2010-es cikkükben szintén leírják a három dimenzióban való regisztráció lépéseit [12]. A legelső teendő a kamera relatív pozíciójának meghatározása a valós világhoz képest, amit a marker jelez. A kamera belső K mátrixa valamint a torzítás paraméterei adottak a γ vektorban.

$$K = \begin{bmatrix} f_x & \sigma & c_x \\ 0 & f_y & c_y \\ 0 & 1 & 1 \end{bmatrix}, \quad \gamma = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

5. Egyenlet – Kamera belső mátrixa és a torzítás paraméterei.

Az 5. egyenlet a következő paramétereket használja:

- f_x , f_y fókusz távolság az x és y irányban,
- (c_x, c_y) a kamera optikai középpontja,
- α_1 , α_2 a sugárirányú torzítás paraméterei,
- β_1 , β_2 az érintő irányú torzítás paraméterei,
- σ a ferdeség az x és y tengely között.

A kamera kalibrálására többféle módszer létezik, Zhan 1998-ban kidolgozott eljárását napjainkban is használják [13].

Amennyiben a belső paraméterek ismertek, akkor a kamera pozíciója kiszámítható egy síkbeli struktúrából. A tér és annak síkbeli vetülete egy 3×3 -as homográfia mátrixszal kifejezhető. A $Z=0$ síkot tekintve a H homográfia mátrix egy pontot jelöl ki a térben, a neki megfelelő kétdimenziós m pont kifejezhető a $P=K[R/t]$ perspektíva alatt.

$$\hat{m} = P\hat{M} = K(R^1 R^2 R^3 t) \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K(R^1 R^2 t) \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

6. Egyenlet - Az m pont meghatározása.

A másik irányban, amennyiben H és K ismert, akkor a kamera pozíciója meghatározható. A H mátrix négy $M_i \leftrightarrow m_i$ összefüggésből és egy direkt lineáris transzformáció (DLT) algoritmus segítségével kiszámítható. Mivel $H_w^t = K(R^1 R^2 t)$, a kamerapozíció t eltolásvektora és az R forgatásmátrix első két sora előállítható a $K^{-1}H_w^t$ kifejezésből. Az R mátrix harmadik sorát (R^3) az $R^1 \times R^2$ adja az R mátrix ortonormált sorai miatt. A kamera pozíciója az általánosságban tovább finomítható egy nemlineáris

minimalizálással. Mivel az eljárás zajérzékeny, ezért csökkenhet a precizitás, amely remegés jelenséget eredményezhet (mintha a virtuális elem remegne a képen). Ha zajos a számítás, akkor a kamerapozíció előállítása a projekciós hibák minimalizálásával történik. Ez a kivetített háromdimenziós pontok és kiszámított kétdimenziós koordináták közti négyzetes távolságot jelenti. Ennek köszönhetően felírható a 7. egyenlet, amely megoldható a Levenberg-Marquardt (LM) algoritmussal, amely egy numerikus megoldást ad a problémára.

$$[R/t] = \operatorname{argmin}_{[R/t]} \sum_i \|P\hat{M}_i - m_i\|$$

7. Egyenlet

Következő lépésként az ARToolKit-nél ismeretett módszerhez hasonlóan kinyerik a bemeneti képből a markert. Ezután a minta normalizációja történik meg a négyzet szögeire alkalmazott perspektivikus transzformáció segítségével. Ennek célja a belső területen lévő minta azonosítása. Az azonosítás a marker típusától függően eltérő megközelítés szerint történik. Kétféle markert különböztetünk meg, az ID-alapút és a sablon-alapút. Az ID-alapú markerek jellemzői, hogy könnyen azonosíthatóak, egyediek, így egymástól könnyen megkülönböztethetőek. A sablon alapú markerek esetén a marker belsejében lévő kép minden ismert minta-sablonnal összehasonlításra kerül. Az összehasonlítás során az elforgatásokat is figyelembe veszi. Az összehasonlításhoz használt hasonlóság vizsgálati módszer az átlagos négyzetes hiba (MSE), amelynek nagy előnye, hogy már egészen kis értékekből is megállapítható a hasonlóság.

$$Mse(\hat{\theta}) = E(\hat{\theta} - \theta)^2 = Var(\hat{\theta}) + Bs^2(\hat{\theta})$$

8. Egyenlet - Átlagos négyzetes hiba egyenlete.

A módszer hátránya a fényviszonyokra való érzékenység. Az MSE-nél egy előnyösebb megközelítés a normalizált kollerációs együttható, ahol a képhez tartozó átlagos (μ_X , μ_Y) és standard (σ_X , σ_Y) szórás már ki van számolva.

$$\operatorname{corr}(X, Y) = \frac{\operatorname{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

9. Egyenlet - Normalizált kollerációs együttható egyenlete.

Ha az együttható egy adott küszöbérték mellett maximális, akkor a kép elfogadott [14].

Az ARToolKit-nak hátránya, hogy a marker felismeréséhez szükség van a kép körül egy keretre, valamint a teljes markernek a kamera látószögében kell lennie. Azonban az évek folyamán kifejlesztettek olyan eszközöket is már, amelyek tetszőleges képet felismernek anélkül, hogy keretet kellene köré tennie, valamint nem szükséges a teljes képnek a kamera

látószögében lennie. Ilyen például az IN2AR⁸ Adobe Flash alapú kiterjesztett valóság engine. Minél részletesebb és kontrasztosabb a nyomkövetésül használt kép, annál könnyebb azt felismernie a rendszernek.

A fent bemutatott módszerek mind valamilyen képrészlet vagy marker felismerésén alapultak. A következő eljárás, amit Lee és Höllerer [15] dolgozott ki 2007-ben, szakít ezzel az elvvel, és marker helyett az emberi kéz jellegzetes mintáit használja az azonosításhoz. Az általuk Handy AR-nek keresztelt rendszer a felhasználó kinyújtott kezén lévő ujjbegyeket használja a nyomkövetéshez. A kéz kamerához képesti helyzetéből képesek stabilizálni a kéz tetején megjelenő 3D-s grafikákat, aminek köszönhetően a felhasználó különböző látószögekből is kényelmesen megtekintheti a virtuális objektumokat. Az ujjbegyeket a felhasználó kezének körvonala alapján detektálják, majd a hüvelykujjtól számított pozíció alapján sorba rendezik, ami után felhasználják azokat a kamera helyzetmeghatározó algoritmusához. Amíg az öt ujjbegy nyomkövetése sikeres, addig ez az algoritmus megfelelő pontossággal meg tudja határozni a kamera helyzetét a kézhez képest, aminek köszönhetően a felhasználó szabadon mozgathatja vagy forgathatja a kezét, a virtuális objektum akkor is látszani fog.



15. ábra - Egy virtuális nyúl a Handy AR felhasználó kezén, különböző látószögekből szemlélve.

2.3. Alkalmazási területek

A kiterjesztett valósággal a hétköznapi életben először sportközvetítések során találkozhattunk. Labdarúgó mérkőzéseken a középkezdéskor pályára vetített klubcímerek, vagy úszás, kajak-kenu esetén a célba érés előtti utolsó métereken az első versenyző előtt húzott pályára vetített vízszintes vonal is ezzel a technológiával készült. A marketingben is jól használható, például a felhasználó felpróbálhatja a ruhákat virtuálisan a vásárlás előtt, így nincs szükség a tényleges felpróbálásra és időigényes átöltözködéssre. A turizmusban a pozíció

⁸ <http://www.in2ar.com/>

és irány alapú AR nyújtotta lehetőségeket lehet kitűnően kiaknázni, például városnézés során a mobilunk kameráját használva megtalálhatjuk a környezetünkben lévő múzeumokat, amelyek adatai meg is jelennek a kijelzőn, holott lehet, hogy még a látómezőnkben sincsenek. A továbbiakban különböző alkalmazási területeket mutatok be konkrét megvalósított példákon keresztül.

2.3.1. Egészségügy

A kiterjesztett valóság nyújtotta előnyöket az orvosok műtétek vizualizációjának támogatására használhatják. 3D-s adathalmazokat nyerhetnek ki a páciensről valós időben nem invazív⁹ szenzorokat használva (például MRI, CT és ultrahangos képek). Ezeket az adatokat jelenítenék meg valós időben a páciensen. Ez nagyon hasznos lehet minimális-invazív műtéteknél, ahol csökkentheti a műtéti traumát azáltal, hogy kevesebb bemetszést kell végrehajtani, vagy egyáltalán nem lenne szükség bemetszésekre. A probléma a minimális-invazív technikákkal az, hogy ezek redukálják az orvos páciensbe való betekintésének lehetőségét, így a műtét nehezebb. Az AR technológia lehetővé tenné belső nézetek elérését nagyobb bemetszések nélkül.

Rosenthal és társai [16] 2001-es cikkéből egy melldaganat biopsziát megkönnyítő alkalmazást ismerhetünk meg. A virtuális objektumok azonosítják a tumor helyét, és segítik a tűt célba érni. A hagyományos ultrahang vezérelte biopszia pontosságát hasonlították össze a 3D-s AR technológia vezérelte biopszia pontosságával fantomokon¹⁰. Céljuk egy olyan rendszer hatékonyságának bemutatása, amely egyszerű, és nem veszélyes a betegre. Az általuk kapott eredmények szerint a fejre illeszthető kijelző használatával a kívánt céltól statisztikailag átlagosan kisebb eltérést kaptak, mint a hagyományos módszerrel.

⁹ Az az orvosi eljárás, melynek során a testbe vágás vagy szúrás által behatolnak.

¹⁰ Orvosi kifejezés, jelentése: az emberi testnek vagy valamely részének átlátszó modellje.



16. ábra - Mind az ultrahangos szonda, mind a tű nyomkövetve van. A rendszer jelzi a tű síkbeli vetületét az ultrahangos szeletbe (kék vonalak), valamint megjeleníti a tű tervezett pályáját az adott pillanat szerint.

2.3.2. Gyártás és javítás

A bonyolult szerkezetek karbantartásához és javításához is jól használható a kiterjesztett valóság. Az utasításokat könnyebb lehet megérteni, ha nem kézikönyvek formájában kapjuk meg, hanem 3D-s alakzatokat helyezünk el a tényleges berendezésen, amelyek bemutatják lépcsről-lépésre a feladat végrehajtását. Ezek a 3D-s alakzatok lehetnek animáltak is, így még hatékonyabbá tehető az útmutatás. Ehhez a területhez láthattunk már példákat az AR történetét taglaló 2.1. alfejezetben is.

Henderson és Feiner 2009-es cikkében [17] egy olyan kiterjesztett valóság alkalmazás tervezését, implementálását és tesztelését írja le, amely segítségével páncélozott járművek tornyának belsejében végrehajtott rutin karbantartási feladatokat lehet megkönnyébbíteni a szerelők számára. A prototípus fejre illeszthető kijelzőt használ, amely kibővíti a szerelő látómezejét szövegekkel, címkékkel, nyilakkal és animált utasítássorozatokkal, amely megkönnyíti a feladat megértését és végrehajtását. Az esettanulmányok során profi katonai szerelők tesztelték a rendszert 18 gyakori feladat gyakorlati körülmények közötti végrehajtásával. Ezek a feladatok magukba foglalták a kötőelemek és jelzőfények, valamint a csatlakozó vezetékek telepítését és eltávolítását, és hasonló egyéb teendőket. A fejre illeszthető kijelzőn kívül még egy fix lapos kijelzőjű megjelenítőeszközt is kipróbáltak a jelenleg alkalmazott laptop alapú dokumentálás javított változataként. Az AR alkalmazás

lehetővé tette a szerelőknek, hogy gyorsabban oldják meg a feladatokat a korábbiaknál, valamint sokkal kevesebb fejmozgásra volt szükségük. A kvantitatív felmérések alapján a szerelők könnyen felhasználhatónak és kielégítőnek találták a rendszert.



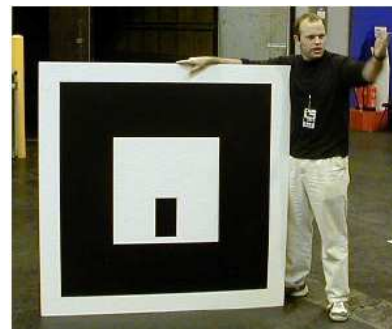
17. ábra - A prototípus használat közben.

2.3.3. Szórakoztatás

A Duran Duran zenekar 2000-es élő koncertkörútjára különféle valós idejű vizuális effekteket fejlesztettek ki, amihez az ARToolKit-et használták fel. A szoftver segítségével a rendszer az élő videón valós időben összekombinálta a közönséget vagy a zenekart a 3D-s animációkkal. Az eredményt egy nagy képernyőre vetítették, amit megtekinthetett a közönség. Az animációk előre be voltak programozva, és a koncert közben az énekes kérésére jelentek meg, így lehetővé vált, hogy az animált karakterek együtt éljenek a koncerttel a kivetítőn. A koncert közben nem használták a markereket a nyomkövetéshez, mivel a véletlenszerű fényviszonyok óriási mértékben megnehezítették volna az egyenletes eredményt. Azonban rendkívül hasznosnak találták az ARToolKit-et a 3D-s helyzet lokalizációhoz a színpadi környezetben. A koncert előtt az ARToolKit markereit használták, hogy rögzítsék a 3D-s koordinátákat, ahol az animált karakterek megjelennek a koncert alatt [18].



18. ábra - Animált karakter a zenekar mögött.



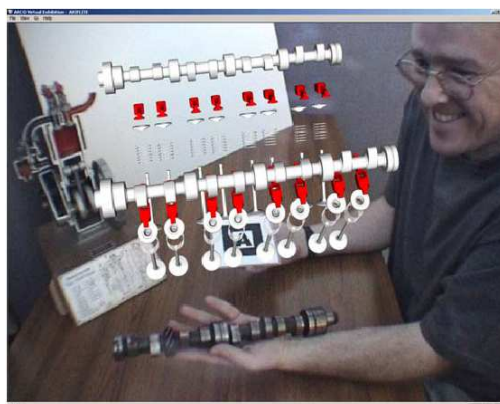
19. ábra - Marker a 3D koordináták megállapításához.

2.3.4. Oktatás

Az AR technológia az oktatásban elsősorban a szemléltetés újfajta alternatívájaként jelenhet meg. Kémia oktatásában a molekulák térbeli szerkezetének ábrázolásához használható, erre mutat példát az AR Kémia, amelyről a későbbiekben részletesen tájékozódhatunk majd. Biológia tanításánál a csontokat és belső szerveket az emberre vetítve megkönnyebbíthetnénk ezek elhelyezkedésének megtanulását a diákok számára. A matematika oktatásában is jól alkalmazható a térbeli testek szemléltetésére, ugyanis ezeket síkban nehéz jól ábrázolni [19].

Liarokapis és társai [20] cikkükben egy olyan oktatási alkalmazást mutatnak be, amely lehetővé teszi a felhasználók számára, hogy kölcsönhatásba léphessenek 3D-s webtartalmakkal (Web3D¹¹) virtuális és kiterjesztett valóságot alkalmazva. Ez lehetővé teszi a Web3D és az AR előnyeinek vizsgálatát a mérnökoktatásban. A hagyományos anyagot multimédiás tartalommal gazdagítják, ami megtekinthető az interneten keresztül vagy lokálisan, vagy egy asztali AR környezetben. A szerzők a rendszer életképességét négy gépészmérnöki téma oktatásán keresztül mutatják be.

A bemutatott oktatási rendszer segíti a tananyag megértését a hallgatóknak a Web3D és AR használatával. A mérnöki és tervezőalkalmazás kísérleti célja, hogy segítse a gépészeti fogalmak gyors megértését. Meg kell jegyezni, hogy a legnagyobb hangsúlyt a 3D-s megjelenítés kapta a rendszerben, mert a 3D azonnal növeli a tanulási folyamat eredményét. Például egy tanár el tudja magyarázni, hogy mi a vezérműtengely diagramok, képek és szöveg segítségével. Azonban még így is nehéz megérteni, mi is az a vezérműtengely. Az aktuális rendszerben a Web 3D képek, szöveg és 3D-s modellek (amelyek lehetnek animáltak is) segítségével vizualizálják a vezérműtengelyt, amelyet manipulálhatnak a hallgatók



20. ábra - Dugattyú vizualizálása az oktatószoftver segítségével.

¹¹ <http://www.web3d.org/realtime-3d/>

2.3.5. Játék

Számos kiterjesztett valóság játék látott már napvilágot, ilyen többek között a korábban már említett ARQuake, a 2007-ben megjelent első kereskedelmi AR játék, a The Eye of Judgment¹², vagy az ARhrrrr¹³.

Bergig és társai [21] 2011-es cikkükben bemutatnak egy új játékplatformot, az AR Rubik-kockát. A kocka egyszerre működik kontrollerként és játéktáblaként. Az interakció keveréssel és döntéssel hajtható végre. A Rubik kocka AR játékplatformként való használatához két játékot készítettek el. Az 21. (a) ábrán egy pillanatkép látható a puzzle játékból. A játékban virtuális falvak találhatók a kockán. Ezek a falvak fejlesztési erőforrásokat kapnak. A cél az erőforrások egyensúlyi elosztása a falvak között. Ahhoz, hogy ezt elérjük, a játékos megkeveri a virtuális erőforrásokat a falvak között. Az 21. (b) ábra egy pillanatkép a második játékból, amely egy ügyesség-alapú labirintusjáték. Ebben a játékban a játékosnak segítenie kell a karakternek elérni a célt, ehhez a kockát döntenie kell, és óvatosan végigvezetni a figurát egy keskeny ösvényen. Ha nem vigyázunk, szabadesésben lezuhanhat a kockáról a földre.



21. ábra - AR Rubik-kocka. (a) puzzle játék és (b) labirintus játék.

Az architektúra, amely szétválasztja a platformot és a játék implementációt, lehetővé teszi különböző játékok kifejlesztését ugyanazon az eszközön. A rendszer két rétegbe szerveződik. A magréteg elemzi az élő videóképet és detektálja a kocka helyzetét minden egyes frame esetén. A következő réteg tárolja az előzményeket és leképezi a magrétegre a játék interakciókat, valamint stabilizálja a regisztrációt. A magréteget az OpenCV és az ARToolKit segítségével készítették el.

¹² <http://www.eyeofjudgment.com/>

¹³ <http://ael.gatech.edu/lab/research/handheld-ar/arhrrrr/>

3. Fejlesztőeszközök

A következő fejezetben kiterjesztett valóság alkalmazások elkészítésére szolgáló könyvtárcsomagokról olvashatunk. A könyvtárakat két nagy kategóriába osztottam aszerint, hogy asztali AR alkalmazás készítésére alkalmasak, vagy pedig okostelefonokra szánt programokat készíthetünk a segítségükkel.

Az asztali fejlesztőeszközökkel C++, C#, Java és ActionScript nyelvű programok készíthetők el, a háromdimenziós tartalmak megjelenítésére pedig OpenGL, DirectX, XNA valamint az ismert 3D-s Flash motorok használhatóak fel.

A mobiltelefonokhoz szánt könyvtárcsomagok markeres és markerless¹⁴ Java nyelvű programok elkészítését teszik lehetővé.

3.1. Fejlesztőeszközök asztali alkalmazásokhoz

3.1.1. ARToolKit

Az ARToolKit az egyik legelső publikus könyvtár kiterjesztett valóság alkalmazások C vagy C++ nyelven való fejlesztéséhez. Hirokazu Kato és Mark Billinghurst fejlesztette ki 1999-ben, és a Washingtoni Egyetemhez tartozó Human Interface Technology Laboratory adta ki. Nagyon széles körben elterjedt AR fejlesztőeszköz, 2004 óta több mint 500000-szer töltötték le¹⁵. Segítségével a programozónak egyáltalán nem kell a képfeldolgozással foglalkoznia. Az ARToolKit a markerek követésére számítógépes látási algoritmusokat használ. A nyomkövető könyvtár valós időben számolja ki a kamera pozícióját a markerekhez képest. Az alkalmazás elkészítésekor meg kell adni, hogy milyen mintákat (pattern) keresünk a marker fekete négyzetének közepén belül. Ez a minta egy 16×16-os mátrix, amelyet minden lehetséges markerre kipróbál. Ennek következtében a felhasznált markerek számában nő a markerfelismerés ideje. Amennyiben sikeres találatot ad a rendszer, kiszámítható, hogy milyen transzformációs mátrixot kell megadni a 3D-s megjelenítésért szolgáló OpenGL-nek. Alapértelmezett kamerakalibrálás tartozik hozzá, amely során a kamerák különböző mértékű képmezőelhajlása korrigálható [3]. Az ARToolKit nagy hangsúlyt fektet a platformfüggetlenségre, ezért különböző operációs rendszeren futtatható (Windows, Linux, MacOS).

¹⁴ Amikor nincs szükség a nyomkövetésül szolgáló kép köré keretet helyezni.

¹⁵ <http://sourceforge.net/projects/artoolkit/files/stats/timeline>

A saját markerek elkészítésének lehetősége is adott, ehhez a letöltött könyvtárban található marker-sablon felhasználására van szükség. A sablon segítségével elkészített, használni kívánt markerhez az ARToolKit által nyújtott mk_patt programmal állítható elő a hozzá tartozó pattern fájl. A pattern fájl elkészítése után a marker szabadon felhasználható a programokban.

A könyvtárcsomag segítségével több virtuális objektumot kezelhetünk szimultán, reagálhatunk a felhasználói interakciókra, az OpenGL segítségével 3D-s modelleket, valamint VRML animációkat jeleníthetünk meg. Egy ARToolKit alkalmazás az alábbi hat lépésből tevődik össze:

1. videó inicializálása, pattern fájlok és kamera paramétereinek beolvasása
2. videó képének lekérése
3. markerek detektálása és minták felismerése a videó képen
4. kamera-transzformáció kiszámítása a detektált mintához képest
5. virtuális objektum kirajzolása a detektált mintára
6. videó bezárása.

Az 1. és 6. lépés inicializáláskor, illetve az alkalmazás befejezésekor történik, míg a 2-5. lépés az alkalmazás futása során folyamatosan ismétlődik.

Az ARToolKit kétféle licenz alatt áll. A nem kereskedelmi felhasználás esetén a GNU GPL¹⁶ licenz érvényes az elkészített programokra. Üzleti célra való felhasználáshoz fel kell venni a kapcsolatot az ARToolworks-szel¹⁷ a kereskedelmi engedély megszerzésének érdekében.

3.1.2. NyARToolKit

Az NyARToolKit az ARToolKit Java nyelvre portolt változata. Ez az eredeti verziónál lassabbá teszi a végrehajtást, azonban teljesen architektúra független. Az ARToolKit-hez hasonlóan az NyARToolKit is egy könyvtárcsomag a virtuális valóság fizikai környezetbe való integrálására, beleértve ebbe a valós idejű kamerakezelést, a virtuális objektumok 3D-s renderelését és ezek integrálását a kimeneten megjelenő képbe.

A könyvtárt egy Nyatla néven ismert japán fejlesztő készítette 2008-ban, ezért került a könyvtárcsomag nevében az Ny betű az ARToolKit elé.

¹⁶ <http://www.gnu.org/licenses/gpl.html#SEC1>

¹⁷ <http://www.artoolworks.com/>

Mivel Java-ban írták, ezért együttműködik az Android operációs rendszerrel is, de portolták még különböző platformokra is, mint például a C#, ActionScript, Silverlight. A 3D-s tartalmak megjelenítését C# esetén a DirectX segítségével oldja meg.

Az üzleti licenszet az ARToolworks nevű cég birtokolja, azonban nem üzleti célú felhasználásra ingyenesen használható a technológia, akárcsak az ARToolKit esetében.

Saját marker készítésére a 3.1.1. alfejezetben leírtak szerint van mód. Szintén több marker kezelésére képes párhuzamosan, reagál a felhasználói interakciókra, és kihasználja a DirectX nyújtotta lehetőségeket.

A könyvtárcsomagot folyamatosan fejlesztik, a legutolsó kiadás 2011 májusában történt.

3.1.3. Goblin XNA

A Goblin XNA egy nyílt forráskódú platform 3D-s felhasználói felületek készítéséhez, de természetesen támogatja a hagyományos kétdimenziós komponenseket is. Elsősorban AR játékok készítéséhez ajánlják, támogatja a hang és videó lejátszást is. Ingyenesen letölthető a <http://goblinxna.codeplex.com> linkről, a BSD¹⁸ licenz vonatkozik rá. A fejlesztőeszközt C# nyelven készítették el a Microsoft XNA Game Studio 3.1.-et¹⁹ használva. Mivel az XNA Game Studio-t használja, ezért a Goblin XNA-val írt programokhoz Microsoft Visual Studio használandó.

A hat szabadságfokú marker-alapú nyomkövetéshez az ALVAR²⁰ nyomkövető csomagot használja fel. A képfeldolgozáshoz a korábban már említett OpenCV könyvtárat használja. Képes egyszerre több markert is párhuzamosan feldolgozni.

Bár a számítógépes alkalmazások elkészítésére alkalmas fejlesztőeszközök közé soroltam, a besorolás nem teljesen helytálló, ugyanis Windows Phone-ra is készíthetők programok a segítségével. A platform folyamatos fejlesztés alatt áll, a 4.0-ás béta verzió 2011. október 28-án jelent meg. A Goblin XNA fejlesztését a Microsoft finanszírozta.

3.1.4. FLARToolKit és FLARManager

A FLARToolKit²¹ egy 2009-ben megjelent Flash nyelvű open source kódkönyvtár, melynek segítségével kiterjesztett valóság alkalmazásokat fejleszthetünk. Az 3.1.2.

¹⁸ <http://www.freebsd.org/copyright/license.html>

¹⁹ <http://www.microsoft.com/download/en/details.aspx?id=39>

²⁰ <http://virtual.vtt.fi/virtual/proj2/multimedia/alvar.html>

²¹ <http://www.libspark.org/wiki/saqoosha/FLARToolKit/en>

alfejezetben ismertetett NyARToolkit-en alapul. A FLARToolKit felismeri a bemeneti képből a markert, és kiszámítja a pozícióját a 3D-s világban, ehhez számítógépes látási algoritmusokat használ fel. Erre illeszti rá a modellt, videót, vagy képet. A 3D-s grafikai elemek kezeléséhez különböző segédosztályokon keresztül rendelkezésre bocsátja az ismert 3D-s Flash motorokat (Papervision3D²², Away3D²³, Sandy3D²⁴, Alternativa3D²⁵). Alapértelmezésként a Papervision3D motort használja. Nem kereskedelmi felhasználása ingyenes, a forráskódot elérhetővé kell tenni (GNU General Public License). Üzleti célra való felhasználáshoz fel kell venni a kapcsolatot az ARToolWorks-el a kereskedelmi engedély érdekében.

A szintén 2009-ben megjelent FLARManager²⁶ egy könnyűsúlyú keretrendszer, amely megkönnyíti a kiterjesztett valóság alkalmazások készítését Flash-ben. Kompatibilis a különböző nyomkövető könyvtárakkal, és 3D-s keretrendszerekkel, valamint egy robusztus esemény-alapú rendszert nyújt a markerek kezelésére (hozzáadás, frissítés, eltávolítás). Támogatja és kezeli a szimultán megjelenő több markert is. A licensze a FLARToolKit licenszével azonos.

Támogatott nyomkövető könyvtárak:

- FLARToolKit
- flare*tracker²⁷
- flare*NFT²⁷.

Támogatott 3D keretrendszerek:

- Alternativa3D
- Away3D
- Away3D Lite²³
- Papervision3D
- Sandy3D.

Ez a két ActionScript nyelvű keretrendszer böngészőből futtatható, honlapba ágyazható, több markert párhuzamosan kezelni képes interaktív webes alkalmazások készítésére alkalmas. A segítségükkel készített programok képesek az eddig megszokott 3D-s modelleken és animációkon kívül videók és hangok lejátszására is.

²² <http://code.google.com/p/papervision3d/>

²³ <http://away3d.com/>

²⁴ <http://www.flashesandy.org/>

²⁵ <http://alternativaplatform.com/en/alternativa3d/>

²⁶ <http://words.transmote.com/wp/flaremanager/>

²⁷ [http://www.imagination.at/en/?Products:Augmented Reality for Flash](http://www.imagination.at/en/?Products:Augmented+Reality+for+Flash)

3.1.5. IN2AR

A 2011-ben megjelent IN2AR egy Adobe Flash ActionScript könyvtár, amely lehetővé teszi bármilyen kép detektálását és pozíciójának meghatározását szabványos webkamera segítségével. Az IN2AR robusztus markerless nyomkövetésre képes extrém szögből is (több kép párhuzamos követésére is van lehetőség). A markerless azt jelenti, hogy az eddig ismertett fejlesztőeszközökkel ellentétben nincs szüksége semmilyen keretre a kép körül. A korábbi eszközök hátránya az is, hogy a markeren belül négyzet alapú mintát lehetett csak elhelyezni, azonban az IN2AR képes téglalap alapú képek kezelésére is. Az egyetlen megkötés, hogy 640 pixelnél szélesebb legyen a használni kívánt kép. Minél kontrasztosabb, részletesebb a felhasználni kívánt kép, annál jobban fog sikerülni a nyomkövetés. A kép felismerésének sebessége a felismerendő képek számának függvényében nő.

A fentebb bemutatott fejlesztőeszközöknél a marker detektálásához szükséges a teljes markernek a kamera látószögében lennie. Ezzel szemben az IN2AR-nek erre sincs szüksége, a kép egy nagyobb részét látva már képes a nyomkövetésre. Ezen kívül könnyen felhasználhatóak az ismert 3D-s Flash motorok is. A letölthető SDK-ban jól használható példaprojekteket is találunk.

Nem üzleti célra való használata ingyenes, viszont hátránya, hogy a pattern fájlokat nem lehet önkezűleg elkészíteni az ingyenes licenz keretein belül, valamint hogy az elkészített projekten megjelenik az IN2AR logója. Amennyiben saját képeket szeretnénk használni, el kell küldeni azokat a keretrendszer fejlesztőinek (egyszerre maximum tízet lehetséges), akik néhány napon belül visszaküldik az elkészített pattern fájlokat. A kereskedelmi licenz ára borsos, 2950 euró projektenként. Cserébe nem fog szerepelni a programban az IN2AR logó, és a képekhez tartozó pattern fájlokat szintén elkészítik.

A könyvtárcsomagot folyamatosan fejlesztik, a legfrissebb kiadás 2011. október 12-én jelent meg. Jelenleg még a mobiltelefonokat nem támogatja, azonban már tesztelik az Android-hoz való könyvtárakat, és terveik szerint azt átírják majd iOS-re is.

3.2. Fejlesztőeszközök mobil alkalmazásokhoz

3.2.1. AndAR

Az AndAR²⁸ az ARToolKit Androidra portolt változata. Objektumorientált Java API-t nyújt, amely elrejt a natív könyvtári hívásokat. Mivel az ARToolKit-en alapszik, ezért örökli annak tulajdonságait is (marker alapú, OpenGL-t használ a háromdimenziós tartalmak megjelenítéséhez, kétféle licenz vonatkozik rá). A projekt honlapján jól kidolgozott mintaalkalmazások érhetőek el, így ezek, valamint a dokumentáció segítségével könnyen elkezdhető a fejlesztés. A legutolsó frissítés 2010 májusában történt.

3.2.2. Qualcomm SDK

A 2010-ben Qualcomm által elkészített SDK²⁹ segítségével Android és iPhone készülékekre lehet kiterjesztett valóság alkalmazásokat készíteni. Windows, Mac OS X és Linux operációs rendszerek alatt futtatható. Az AndAR-el ellentétben nincs szükség fekete keretes markerekre, hanem tetszőleges kép megadható neki a nyomkövetésre. Kifinomult algoritmusokkal detektálja a képet a rajta megtalálható természetes jellemzők alapján. Felismeréskor ezeket a jellemzőket hasonlítja össze az adatbázisban tárolt erőforrásokkal. Ha egy kép egyszer detektálva lett, akkor az SDK addig nyomon fogja követni, amíg legalább részben a kamera látóterében van.

A képekhez tartozó pattern fájlt az online Target Management System³⁰ segítségével hozhatunk létre, amihez előzetes regisztrációra van szükség. JPG vagy PNG formátumú képből hoztató létre egy pattern fájl. A feldolgozott képek jellemzőit letárolja az SDK egy adatbázisban és a képfelismerés esetén futásidejű összehasonlításra használja. Szimultán több kép felismerésére és kezelésére is alkalmas a fejlesztőeszköz. A háromdimenziós megjelenítéshez az AndAR-hez hasonlóan OpenGL-t használ.

Az SDK ingyenesen letölthető és felhasználható, a segítségével elkészített alkalmazások kereskedelmi forgalomba hozatalához el kell fogadni a Qualcomm által kínált licenst, fizetni azonban nem kell érte.

A legutolsó frissítés az Androidos verzió esetében 2011 augusztusában, míg az iPhone verziójának esetében 2011 szeptemberében történt.

²⁸ <http://code.google.com/p/andar/>

²⁹ <https://ar.qualcomm.at/qdevnet/>

³⁰ <https://ar.qualcomm.at/qdevnet/projects>

3.3. Összehasonlítás

A fejlesztőeszközök tulajdonságait egy táblázatban gyűjtöm össze, így könnyen áttekinthető, hogy melyik könyvtárnak mi a sajátossága, milyen esetben használatos. Nyolc szempont alapján osztályozom az eszközöket. Ezek a megjelenés éve, a programozási nyelv, a licenz (ahol szétválasztják a kereskedelmi és nem kereskedelmi célú licenst, ott azt is feltüntetem), a velük készíthető alkalmazás markerless-e, mobil vagy asztali alkalmazások készítésére szolgál, tartalmaz-e valami extrát az alap megjelenítéseken (kép, 2 vagy 3 dimenziós grafika, modell, animáció) kívül, bővíthető-e egyedi markerekkel/képekkel, valamint hogy a nyomkövetéshez szükséges-e a teljes markernek/képnek a kamera látószögében lennie. Ez utóbbit hely híján nem tüntetem fel a táblázatban, ugyanis értékeit megkaphatjuk a markerless oszlop értékeiből: amelyik eszköznek nincs szüksége markerre az azonosításhoz, annak a teljes képet se kell látnia a kép felismeréséhez. A licenz oszlopban egy kis jelmagyarázatra van szükség. Ahol két érték szerepel, ott az első a nem kereskedelmi használatra készített alkalmazásra vonatkozó licenz, míg a második a kereskedelmire vonatkozó. Az I betű az ingyenes rövidítése, míg az S a saját rövidítése, amely arra is utal, hogy a licenst meg kell vásárolni.

	Megjelenés	Nyelv	Licenz	Markerless	Mobil/asztali alkalmazás	Extra	Egyedi marker
ARToolKit	1999	C++	GNU/S	Nem	Asztali	-	Igen
NyARToolKit	2008	Java, C#	GNU/S	Nem	Asztali	-	Igen
Goblin XNA	2009	C#	BSD	Nem	Mindkettő	Hang, videó	Igen
FLARToolKit	2009	ActionScript	GNU/S	Nem	Asztali	Hang, videó, web	Igen
FLARManager	2009	ActionScript	GNU/S	Nem	Asztali	Hang, videó, web	Igen
IN2AR	2011	ActionScript	I/S	Igen	Asztali	Hang, videó, web	Fejlesztők adják
AndAR	2010	Java	GNU/S	Nem	Mobil	-	Igen
Qualcomm SDK	2010	Java, Objective-C	I	Igen	Mobil	-	Igen

1. Táblázat - A fejlesztőeszközök összehasonlító táblázata.

4. Saját fejlesztések

A fejezet tartalmazza az általam elkészített kiterjesztett valóság alkalmazások bemutatását, amelyeket a témavezetőm, Turcsányiné Szabó Márta vezetésével tartott szoftvertechnológia labor és virtuális valóság projekt³¹ keretein belül hoztam létre. Öt programot fogok ismertetni, amelyek a FLARToolKit, FLARManager, NyARToolKit és az IN2AR könyvtárak segítségével készültek. Egyes alkalmazások markert használnak a nyomkövetéshez, míg mások hagyományos képeket. A programok háromdimenziós modelleket, videókat jelenítenek meg, valamint hanghatásokat játszanak le, egyes esetekben egyszerre több markert is felhasználva.

4.1. AR ELTE IK

A programot – amely az Informatikai Kar honlapjáról is elérhető³² – a FLARToolKit segítségével ActionScript nyelven készítettem el. Kihasználva a fejlesztőkörnyezet nyújtotta webes megjeleníthetőséget, az alkalmazás internetről is kipróbálható. A program használatához csak webkamerára és a kinyomtatott markerre van szükség. A markert a webkamera elé helyezve kezünkben tarthatjuk az ELTE Lágymányosi Campus Déli tömbjének háromdimenziós modelljét. A marker mozgatásával párhuzamosan valós időben transzformálódik a modell, így azt akár körbe is forgathatjuk. A háromdimenziós regisztrációhoz használt marker különlegessége, hogy az AR marker egyben egy QR kód is, amely tartalmazza az Informatikai Kar honlapjának a címét. A QR kód³³ egy kétdimenziós vonalkód, amelyet különböző információk tárolására alkalmazhatunk (webcím, üzenet, SMS, névjegy, GPS koordináták stb.). A kialakítás újdonságnak tekinthető, amelynek köszönhetően a marker két technológia előnyeit ötvözi. A programot az ELTE IK nyílt napján, valamint a Kutatók Éjszakáján tartott bemutató órán prezentáltam.

³¹ <http://matchsz.inf.elte.hu/VVprojekt/>

³²

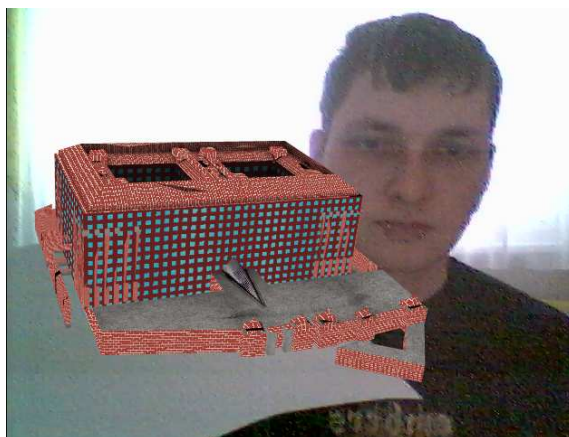
<http://www.inf.elte.hu/karunkrol/hirek/Lapok/ELTED%C3%A9li%C3%A9p%C3%BClet%C3%A9nekmegtekint%C3%A9se3D-benARk%C3%B3dseg%C3%ADts%C3%A9g%C3%A9vel.aspx>

³³ http://en.wikipedia.org/wiki/QR_code



22. ábra - A programhoz használt marker, amely egyben QR kód is. A háttérben a Déli tömb látható.

Az alkalmazás elkészítéséhez Eclipse IDE-t használtam, amelyhez telepíteni kellett az Adobe Flash Builder Plugin-t. Ez utóbbi eszköz nem ingyenes, azonban a hatvan napos próbaverziója térítésmentesen letölthető. A telepítés elvégzése után könnyen birtokba vehető a FLARToolKit. Az alkalmazás a következő főbb elemekből épül fel. Elsőként meg kell adni a webkamerát reprezentáló objektumnak a kamerakalibrációs fájlt, amelyet a könyvtár csomag tartalmaz. Ezt követően be kell állítani a markert reprezentáló objektumot, hogy használja fel a markerhez tartozó pattern fájlt, ami a háttérben egy 16×16-os mátrixként tárolódik. Ennek segítségével tudja majd elvégezni a program a háromdimenziós modell transzformálását. Miután ez megtörtént, be kell állítani a FLARToolKit környezetet. Ez teszi lehetővé a marker nyomkövetését és végzi el helyettünk a képfeldolgozás feladatát. Az alkalmazás a háromdimenziós megjelenítéshez a Papervision3D motort használja, így szükségünk van az ehhez tartozó objektumok létrehozására is. Rendereléskor a FLARToolKit megvizsgálja, hogy detektál-e markert, és amennyiben ez megtörtént, elvégzi a transzformációkat és megjeleníti helyén a 3D-s modellt. Az épület modelljét Jámbori András készítette el a Maya modellezőeszköz segítségével, és a programom .dae³⁴ formátumban használja fel.



23. ábra - A program képe futás közben a webkamerán keresztül.

³⁴ <http://en.wikipedia.org/wiki/COLLADA>

4.2. AR Póló

A következő projekt az ELTE IK nyílt napjára készült. Az alapkonceptió az volt, hogy rugaszkodjunk el a marker hagyományos, papírra nyomtatott használatától, és egy újfajta módon próbáljuk felhasználni azt. Ennek szellemében elkészült egy póló, amire egy marker lett nyomtatva. A marker helyén a programban egy Flash animáció jelenik meg, amely felsorolja az ELTE Informatikai Karán megtalálható képzések és szakirányok neveit. Az alkalmazás a FLARManager könyvtárcsomag használatával készült, az előző programhoz hasonlóan Eclipse IDE és Adobe Flash Builder Plugin használatával. A marker nyomkövetésére a FLARToolkit-et használja fel, ugyanakkor kényelmesebbé teszi azok kezelését különböző eseményekkel. Külön esemény tartozik a marker képernyőn való megjelenéséhez, eltűnéséhez, valamint frissítéséhez, amelynek köszönhetően egyszerűbbé válik a virtuális elemek kezelése a programozó számára.

A FLARManager-nek köszönhetően gyorsabban fejleszthetünk AR alkalmazást, mintha a FLARToolkit-et használnánk magában. A konfigurációs beállítások egy XML fájlban tárolhatóak, amelyeket felhasznál az alkalmazás a helyes működés érdekében. Így például több marker használata esetén nincs szükség – a FLARToolkit-tel ellentétben – a kódban annyi markert reprezentáló objektumra, ahány mintát használni akarunk, helyette elegendő egy ilyen, amely a konfigurációs fájlból a pattern azonosítója alapján tudni fogja, hogy melyik marker jelent meg a képernyőn.

A programot működés közben az érdeklődők egy projektorra kivetítve nézhették meg.



24. ábra - A program által a pólón megjelenő animáció.

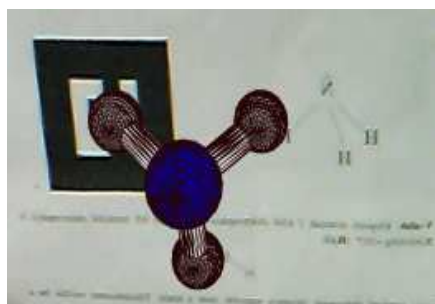
4.3. AR Kémia

A kiterjesztett valóság jól alkalmazható az oktatásban AR könyvek használatán keresztül. AR könyv alatt egy markerekkel ellátott könyvet értünk, amely megfelel a könyvek hagyományos funkciójának, azonban új tartalommal is fel van ruházva.

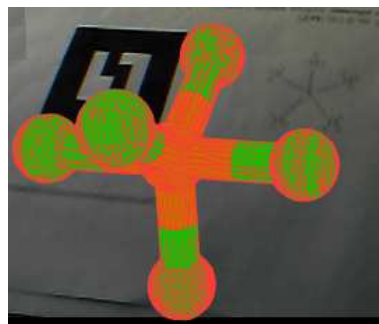
Az ilyen könyvek oktatás hatékonyságának növelésére gyakorolt hatását egy kísérlettel kívántuk igazolni. A kísérlet a budapesti Neumann János Számítástechnikai Szakközépiskolában és a II. Rákóczi Ferenc Gimnáziumban zajlott, és a kémia – azon belül is a molekulaszervezetek – tanítását hivatott segíteni. Előzetes felmérések után kiderült, hogy az osztályokban a diákok fele rendelkezik az új oktatóanyagból való tanuláshoz szükséges webkamerával, míg a másik fele nem. Ez megfelelt a várakozásnak, ugyanis azt szeretttük volna látni, hogy van-e különbség az új típusú és a hagyományos formájú tananyagból készült tanulók között.

A kísérlet kezdeti lépéseiként elkészültek a 3D-s modellek, valamint az oktatóanyag. A markerekkel ellátott tananyag³⁵ öt darab háromdimenziós modellt és két videót tartalmaz. Az alkalmazást szintén a FLARManager segítségével készítettem, amelyet fel kellett készítenem a hét különböző marker és az eltérő formátumú virtuális elemek kezelésére.

Felkészítésként a kémiatanár bevitt néhány kinyomtatott AR markert és bemutatta a tanulóknak az órán a program használatát, de az oktatófüzetet nem nézhették végig. A felkészítés két tanóra alatt történt meg, majd a harmadik órán egy számonkérés következett.



25. ábra - AR Kémia, NH₃ megjelenítése.



26. ábra - AR Kémia, PCl₅ megjelenítése.

A kapott eredmények tükrében a kísérlet sikeresnek tekinthető: az AR csoport tagjainak jelentős része, 60%-a a félévi jegyénél jobbat kapott a dolgozat során, míg a kontrollcsoportban ez az arány csak 27% volt [4].

³⁵<http://arportal.elte.hu/arkemia/bin-debug/>

4.4. Maszk

A következő projekt összetettebb az eddig bemutatottaknál. Az alkalmazás az ELTE és a MOME (Moholy-Nagy Művészeti Egyetem) koprodukciójában szervezett Serious Games pályázatra készült. A komoly játék projektötlete valamilyen módon elősegíti a tanulást, pl. helyszínekhez kötött játék, szabadtéri vagy beltéri múzeumi játék, együttműködést serkentő játék, a szórakozva tanulást elősegítő játék, az aktuális hangulatunkat figyelembevevő játék, 3D virtuális teret az élő térrel összekötő játék.³⁶

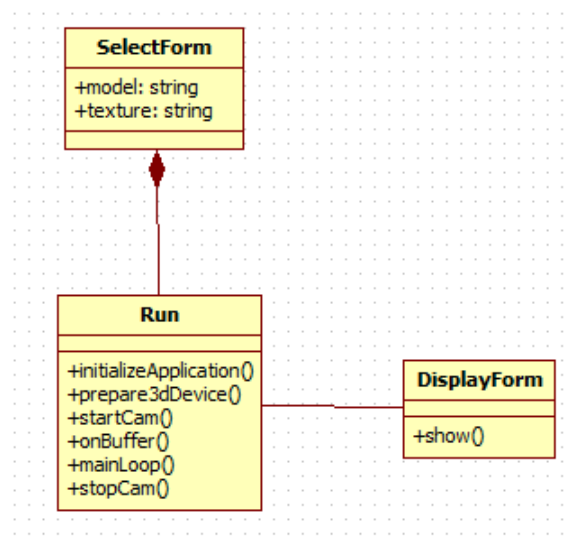
Az általam elkészített alkalmazást gyerekek használhatják múzeumban, vagy akár otthon is. Egy markerekkel ellátott fejpántra és webkamerára van szükség használatához. A fejpánt három markert tartalmaz, a középső helyére vetődik a kamera képén magának a maszknak a háromdimenziós modellje, míg a másik kettővel a maszk méretét lehet szabályozni. Mivel a különböző gyerekek különböző méretű lehet a fejük, ezért szükség van a maszk méretének dinamikus változtatására. Ehhez találtam ki a markeres megoldást, amelynek segítségével nincs szüksége a programot használónak a hagyományos interakciót biztosító eszközökre (egér, billentyűzet) a maszk méretezéséhez. A megoldás a következő. Amennyiben a felhasználó csökkenteni kívánja a maszk 3D-s modelljének méretét, letakarja kezével a fejpánton lévő bal markert, ennek hatására a program végrehajtja a megfelelő transzformációt a modellen. A modell méretének növelése esetén ugyanezt kell megtenni, csak a jobb oldali markerrel. Természetesen a modell transzformálása csak akkor történik meg, ha a középső marker látható, azaz látatlanban nem végzünk el felesleges változtatásokat a modellen. A fejpántnak köszönhetően a gyerekek arca helyett a kiválasztott maszk háromdimenziós modellje jelenik meg a monitoron, így felölthetik magukra a kiválasztott figura szerepét.

A programot Microsoft Visual Studio 2008 IDE segítségével C# nyelven készítettem, az NyARToolKit könyvtárcsomagot és DirectX-et felhasználva. Az alkalmazás elindításakor egy képnézegető segítségével kiválasztható a használni kívánt modell (a képeken a modellekről készült képek láthatóak). Fontosnak éreztem a program egyszerű bővíthetőségét újabb modellekkel, ezért a kódban dinamikusan, előre definiált könyvtárból olvasom be a megjelenítendő képeket. Egy új modell felhasználásához mindössze annyit kell tennie a program felhasználójának, hogy a megjelenítendő képet bemásolja a projekt Images mappájába, a modellt leíró .x kiterjesztésű fájlt a Models mappába, míg a hozzá tartozó textúrát – ha van – a Textures mappába. Természetesen textúra megadása nélkül is működik az alkalmazás.

³⁶ <http://tet.inf.elte.hu/momelte/>

Az eddigi műveleteket egy Form kezelte. Miután megtörtént a modell kiválasztása, el kell indítani az AR alkalmazást – amely műveleteit a Run osztály végzi el –, amelynek több markert kell egyszerre kezelnie szimultán módon. Ehhez három tömbre van szükség, amelyek annyi elemből állnak, ahány markert egyszerre használunk (felhasználónként három markerrel kell számolni, egyszerre több felhasználó is lehet). Ezek a tömbök tárolják az egyes markereket detektáló objektumokat (az NyarToolkit NyARSingleDetectMarker osztály példányai), a markerek adott pillanatban való láthatóságát, valamint a hozzájuk tartozó transzformációs mátrixot.

A programnak első lépésként inicializálnia kell a DirectX környezet objektumait (device beállítása, az előző Form-tól paraméterül kapott modellek betöltése, stb.) és az NyARToolkit objektumait (detektáló objektumok, markereket reprezentáló objektumok, stb.). Ezt követően el kell indítani a webkamerát. Az NyARToolkit onBuffer() eseményében (amely a puffer és kép frissítését végzi el) egy ciklussal végigmegyünk a detektáló objektumokon, és ha az megtalálja az adott képen a hozzá tartozó markert, akkor kiszámítja a megfelelő transzformációs mátrixot, valamint feljegyzi az adott marker láthatóságát. A mainLoop() metódus minden egyes képkockára megkapja az előző eseménytől a markerek állapotát tároló tömböket, és azok láthatóságától függően (ha a modellhez és a kicsinyítéshez vagy nagyításhoz tartozó markerek látszanak) elvégzi az elemi affin transzformációt, majd az eredmények függvényében megjeleníti a maszk megfelelő méretű, háromdimenziós modelljét. Ha csak a középső marker látszódik, akkor nem hajt végre semmilyen skálázó transzformációt. Végül az alkalmazás befejezésekor leállítódik a webkamera.



27. ábra - Az alkalmazás vázlatos osztálydiagramja.

A 27. ábrán látható az alkalmazás vázlatos osztálydiagramja. A kiválasztó ablakban a felhasználó megadja a modellt, majd az ablak létrehozza a megfelelő paraméterekkel a Run osztályt, amely a `DisplayForm` osztályt használja a webkamera képének és a 3D-s modellek megjelenítésére.

4.5. Hangoskönyv

Szintén az ELTE és a MOME által közösen szervezett Serious Games pályázatra készült a Hangoskönyv projekt is. A Bajor Gizi Színészmúzeumban is bemutatásra került, az odalátogató gyerekek is kipróbálták. A projektről készült egy cikk is, amely a Newcastle-ben rendezendő Designing Interactive Systems konferenciára lett elküldve. Az eddig ismertetett alkalmazásoktól eltérően nem keretes markert használ a három dimenzióban való regisztrációhoz, hanem tetszőleges képeket. A program a múzeumban és otthon is használható. Az alkalmazás célja, hogy a gyerekek illusztráció alapján beazonosítva tudjanak dalokat meghallgatni, illetve azokat szövegesen követni, a múzeumi kiállítás részeként, valamint a múzeumon kívülre magukkal vihessenek egyes dalokat.

A projekt keretein belül egy olyan könyv készült el, amelynek bal oldali lapjain egy illusztráció található, míg a jobb oldali lapokon a szöveg olvasható. A könyv webkamera elé kerülése esetén képes azonosítani az illusztrációt, és ezután elindítani az adott képhez rendelt dalt. Lapozás után a következő illusztrációhoz tartozó dalt indítja el. Mivel nincs szükség a könyvből kirívó keretes markerek alkalmazására, ezért kinézetre nincs különbség egy hagyományos és a mostani AR tartalommal bővített könyv között. A hordozhatóságot úgy sikerül elérni, hogy weben keresztül futtatható programot készítettem, aminek a könyvre és egy webkamerára van szükség az elindításához.

A programot ActionScript nyelven készítettem az IN2AR fejlesztőeszköz segítségével, ami lehetővé teszi tetszőleges kép beazonosítását és nyomkövetését. Az ActionScript nyelven való megvalósításnak köszönhetően triviálisan adódik a weben keresztüli alkalmazás is, valamint könnyen kezelhetőek a különböző hangfájlok. A könyv kialakításának köszönhetően egyszerre csak egy illusztráció fog a webkamera képe elé kerülni, ezért az alkalmazás logikája is egyszerűsödik, nincs szükség egyszerre több kép párhuzamos kezelésére, amely lassítaná a feldolgozás idejét.

5. Fejlesztőeszközök összehasonlítása

A 3. fejezetben a különböző fejlesztőeszközökről, míg a 4. fejezetben az általam elkészített alkalmazásokról olvashattunk. Jelen fejezet a 3. fejezetben ismertetett eszközök közül négyet fog részletesen, különböző szoftvermetrikák segítségével összehasonlítani.

A négy kiválasztott könyvtárcsomag az ARToolKit, az NyARToolKit, a FLARToolKit és a FLARManager. A választásom okai a következőek. Az ARToolKit az egyik legelső publikus könyvtár, nagyon sok alkalmazásban felhasználták és a mai napig nagyon elterjedt fejlesztőeszköznek számít. Az NyARToolkit-et azért választottam, mivel .NET keretrendszer használ, és fontosnak éreztem egy ilyen környezet használatát. A FLARToolKit-re a webes felhasználhatósága miatt esett a választásom. A FLARManager-re szintén emiatt, valamint az is szerepet játszott, hogy cserélhető a nyomkövető könyvtára, így többféle módon is használható. Figyelembe vettem, hogy ez a négy fejlesztőeszköz több különböző grafikus környezetet használjon (OpenGL, DirectX, ismert 3D-s Flash motorok), így később ezek alapján is választhasson a programozó. Fontosnak éreztem még, hogy ezen fejlesztőeszközök segítségével olyan programokat írhatunk, amelyek használatához nincs szükség semmilyen drága eszközre (pl.: okostelefon, tablet), hanem egyszerűen a markereket és a webkamerát használják.

Mind a négy eszközzel egy azonos funkcionalitású programot készítettem el, amelyen különféle méréseket végeztem el. A program egy markert kezel, amely helyén egy földgömböt jelenít meg, amit a billentyűzet segítségével forgathatunk, valamint megváltoztathatjuk a sugarát, ezzel biztosítva a felhasználói interakciót. Mindegyik program ugyanazt a markert használta fel működése során.

5.1. Felhasznált metrikák

Ahhoz, hogy a programokat meg tudjuk valahogy mérni, különböző szoftvermetrikákra van szükség. A metrikák a programokhoz egy számértéket rendelnek, amely jelzi a program bonyolultságát. A jelen esetben alkalmazott metrikákat két csoportra bonthatjuk. A méret metrikák a szoftver bonyolultságát valamilyen fizikai tulajdonsága alapján mérik. Tipikusan a forráskód méretét (pl. hosszát, forrásfájlok számát) határozzák meg. Ezek a metrikák nem vizsgálják a programok szemantikáját, csak szintaktikus szempontokat vesznek figyelembe. Emiatt ezek függetlenek a használt programozási paradigmától. Viszont előnyük, hogy maga

a számítás nagyon egyszerű, ezért gyakran használják őket [22]. A másik csoport a komplexitás metrikák, amelyek a program bonyolultságához próbálnak mérőszámot rendelni.

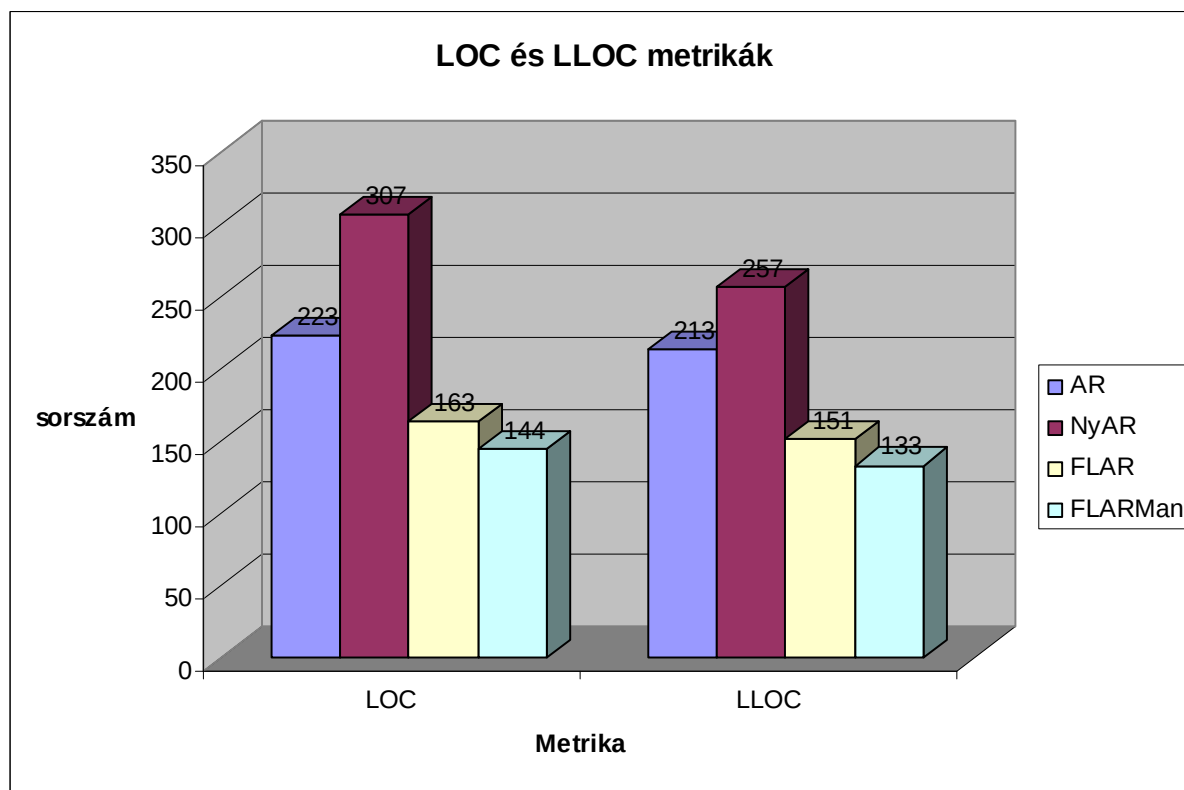
5.1.1. Méret metrikák

A méret metrikák közül négyet használtam a programok mérésére. A LOC (Lines of Code) metrika a program kódsorainak számát méri, az LLOC metrika már csak a nem üres és nem megjegyzés sorokat. Az NA (Number of Attributes) metrika az attribútumok számát, míg az NM (Number of Methods) metrika a metódusok számát méri. A C# nyelvű program több osztályból áll a grafikus felület miatt, ezért ott csak azokat a sorokat vettem bele az LLOC metrikába, amelyeket nem a Visual Studio generált, ugyanígy tettem a metódusok számával is.

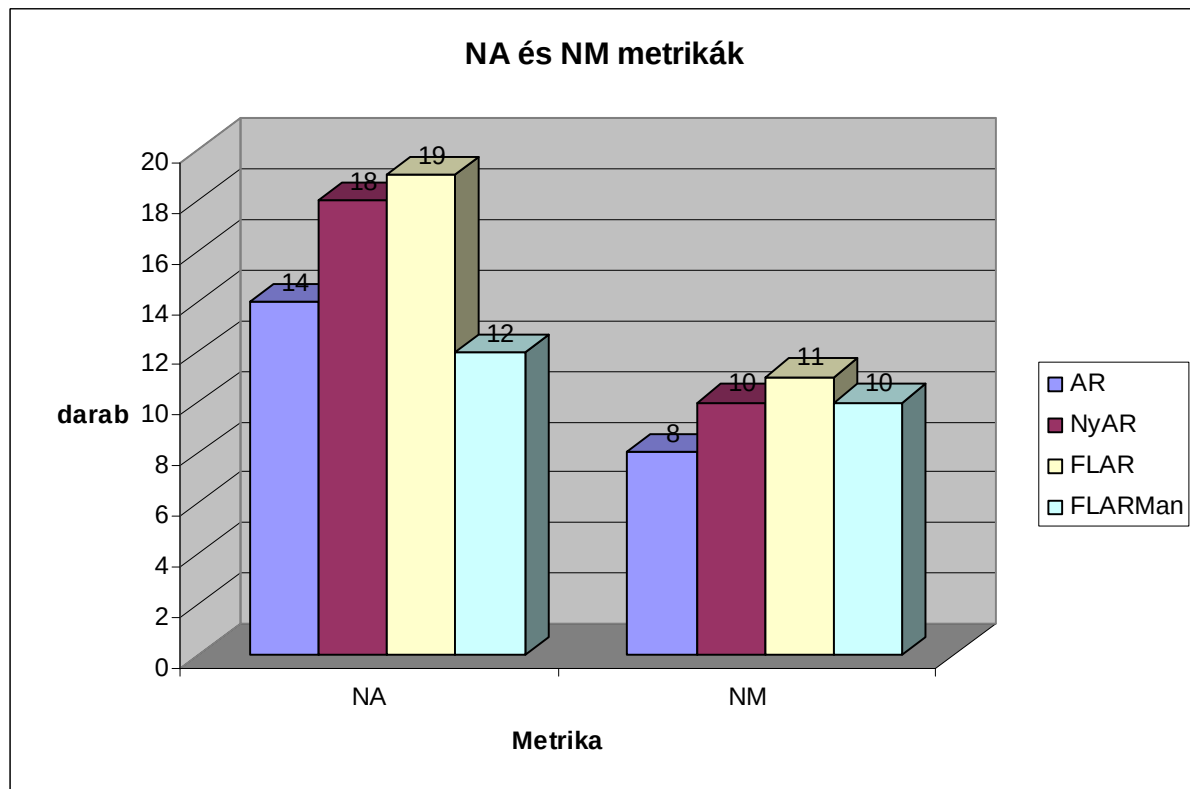
A következő eredményeket kaptam a mérések során.

	LOC	LLOC	NA	NM
ARToolKit	223	213	14	8
NyARToolKit	307	257	18	10
FLARToolKit	163	151	19	11
FLARManager	144	133	12	10

2. Táblázat - Méret metrikák a programokra.



28. ábra



29. ábra

A LOC és LLOC metrikák eredményeiből leolvasható, hogy a legrövidebben a FLARManager segítségével írhatunk meg egy, a fentebb leírt funkcionalitású programot, ezt követi a FLARToolKit, ami el is várható, hiszen a FLARManager azzal a céllal készült el, hogy a FLARToolKit-nál könnyebben tudjunk AR alkalmazást készíteni ActionScript nyelven. Az ARToolKit és a NyARToolKit magasabb értékkel szerepelnek, ez betudható annak, hogy míg az ActionScript Papervision3D könyvtárában van beépített, textúrázható gömb típus, addig az OpenGL-ben és DirectX-ben nincs. Megfigyelhető még az NyARToolKit LOC és LLOC metrikák közti nagyobb különbség, ez annak köszönhető, hogy a Visual Studio által generált kódot nem számítottam bele a hasznos kódsorok számába.

A legkevesebb attribútummal szintén a FLARManager rendelkezik, viszont megfigyelhető, hogy itt a FLARToolKit rosszul teljesít. A metódusok számában az ARToolKit áll a legjobb helyen, míg a második helyen holtversenyben az NyARToolKit és a FLARManager áll.

5.1.2. Komplexitás metrikák

A komplexitás metrikák közül elsőként a McCabe metrikát (ciklomatikus bonyolultság) használtam a programok strukturális bonyolultságának összehasonlítására. E szerint a metrika szerint a programhoz hozzárendelünk egy irányított gráfot. A gráf minden csomópontjának

egy szekvenciális blokkot feleltetünk meg (nem tartalmaz elágazást, vagy ciklust), míg az éleknek a vezérlésátadásokat feleltetjük meg. Az n csúcsú, e élő, p komponensű G gráf ciklomatikus számát a következő összefüggéssel kaphatjuk meg:

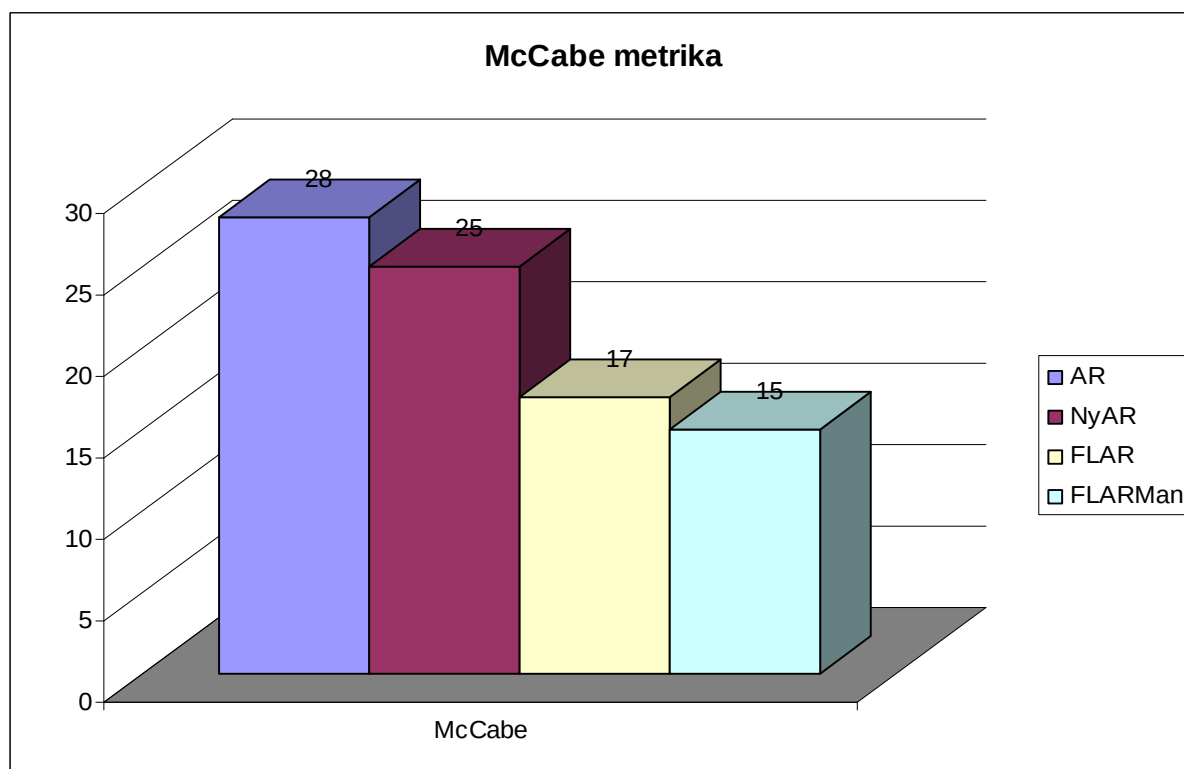
$$V(G) = e - n + 2p \text{ [23].}$$

Ezt a számot a [24] hivatkozás alapján számoltam ki a programok metódusaira. Egy program ciklomatikus számát a metódusai ciklomatikus számának összegzésével kaptam. A McCabe-féle metrikáról részletesen a [23] hivatkozásban olvashatunk.

A következő eredmény született a ciklomatikus bonyolultságra:

	McCabe
ARToolkit	28
NyARToolKit	25
FLARToolKit	17
FLARManager	15

3. Táblázat - McCabe-metrika.



30. ábra - McCabe-féle ciklomatikus bonyolultság a programokra.

A McCabe-féle ciklomatikus bonyolultság szerint strukturálisan a legbonyolultabb a C++ nyelvű ARToolKit, ezt követi a C# nyelvű NyARToolKit. A legegyszerűbb bonyolultságú a FLARManager. Ez megfelel az általam az implementálás során személyesen tapasztaltaknak.

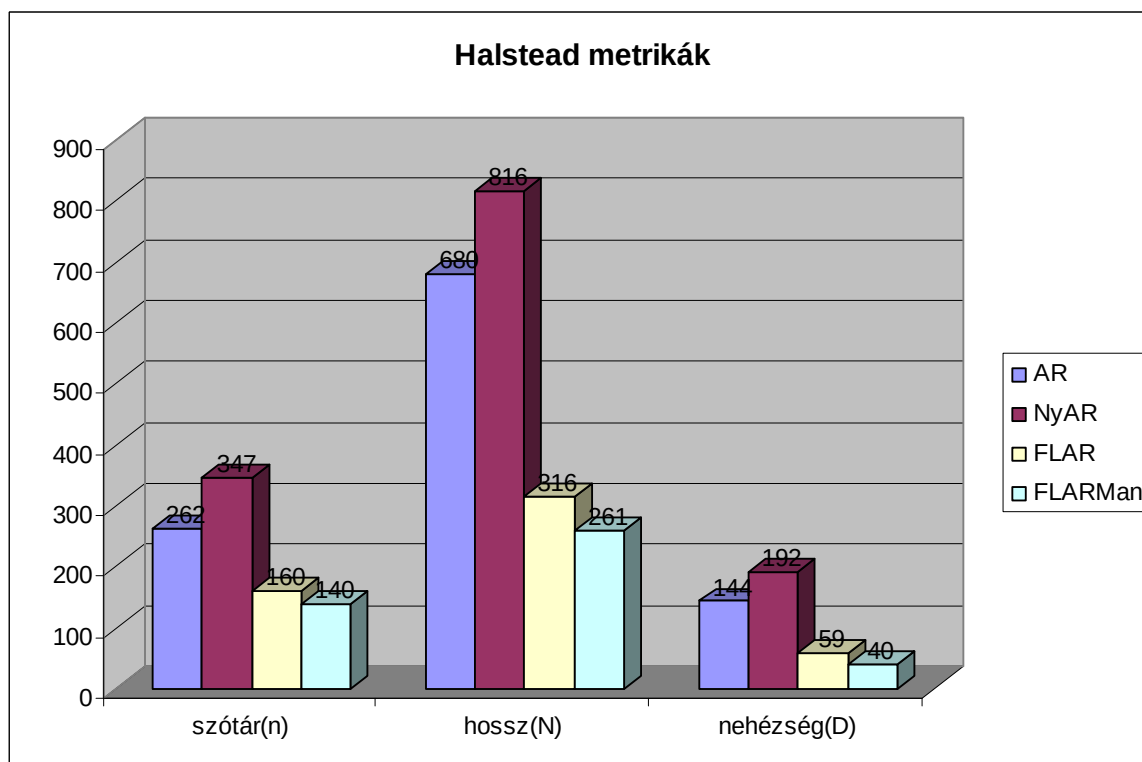
A másik komplexitás metrika, amit kiszámoltam a programokra, a Halstead-féle metrika, mely a program több tulajdonságát együttesen jellemzi. Ezek a tulajdonságok a programszótár, a programhossz, a tárhely, és a nehézség. A programszótár a programot tokenek sorozataként értelmezi. Egy token vagy operátor, vagy operandus. A programszótár mérete a különböző tokenek számával egyezik meg ($n = n_1 + n_2$). A programhossz a (nem feltétlenül különböző) operandusok (N_1) és operátorok (N_2) összegeként kapható meg ($N = N_1 + N_2$). A tárhelyet a következőképp számolhatjuk ki: $V = N \cdot \log_2 n$. A nehézséget pedig az $(n_1/2) \cdot (N_2/n_2)$ képletből kaphatjuk meg. A tárhely (V) az elkészült program bonyolultságát is jelzi, minél magasabb ez az érték, annál nagyobb szellemi ráfordításra van szüksége a programozónak [22, 25, 26].

A program értékeit a McCabe-féle metrikához hasonlóan a programok metódusaira kiszámított értékek összegzésével kaptam.

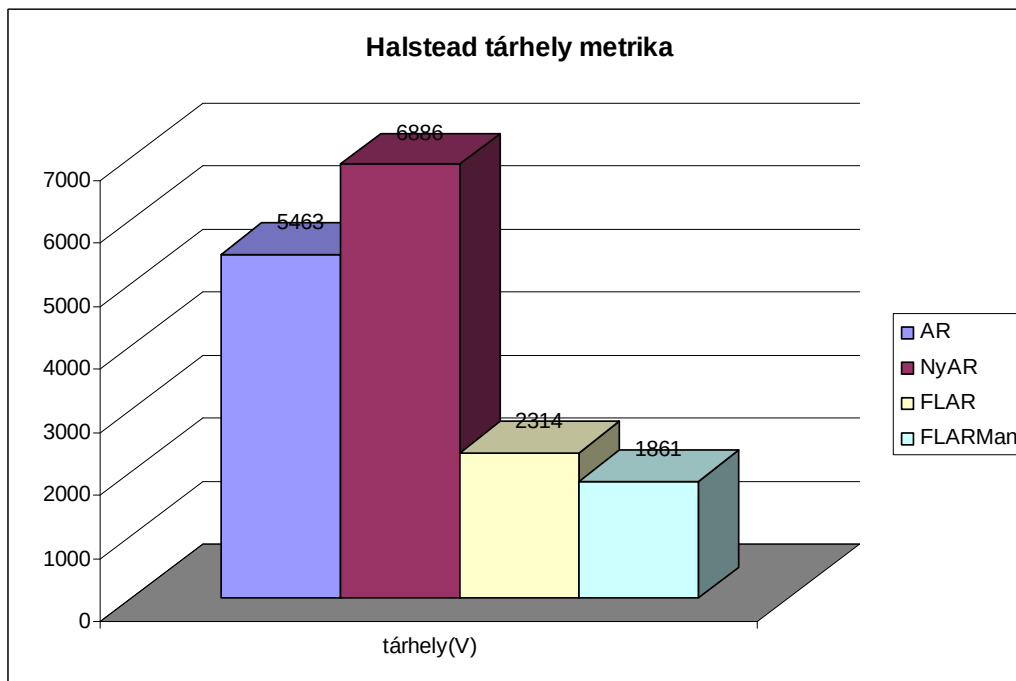
A számítások után a következő eredményre jutottam:

	szótár (n)	hossz (N)	tárhely (V)	nehézség (D)
ARToolKit	262	680	5463	144
NyARToolKit	347	816	6886	192
FLARToolKit	160	316	2314	59
FLARMananager	140	261	1861	40

4. Táblázat - Halstead-metrikák.



31. ábra

**32. ábra**

Mind a négy jellemzőnél megfigyelhető, hogy a legalacsonyabb – legjobb – értéket a FLARManager érte el, valamint a legrosszabbat az NyARToolKit. A kettő között második és harmadik helyen a FLARToolKit és ARToolKit szerepel. Az ARToolKit és NyARToolKit rendre jóval magasabb értékeket szereztek az ActionScript nyelvű eszközöknél, ez valószínűleg annak köszönhető, hogy a forráskódok hossza azokban az esetekben jóval nagyobb.

A kapott eredmények nagyjából megfelelnek a fejlesztés során általam tapasztaltaknak, bár a tárhely és nehézség jellemzőnél az NyARToolKit számértéke felülmúlja az ARToolKit-ét, ami szerint jóval több erőfeszítésre lenne szükség az elkészítéséhez, azonban személy szerint én nem éreztem ekkora különbséget az elkészítés nehézségében.

5.2. Egyéb mérések

A szoftvermetrikákkal kapott eredményeken kívül még egyéb tulajdonságok összehasonlítását is fontosnak éreztem. Ezen tulajdonságok próbálják jellemezni az adott fejlesztőkészlettel elkészített program felhasználói élményét, hasznosíthatóságát. Nem olyan szabványos, szigorú mérési eszközzel kapott eredmények, mint amiket a metrikák felhasználásával kaptam, de eredményük sokat mond a program használhatóságáról. Ezek az egyéb tulajdonságok a következők:

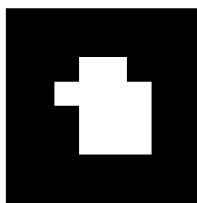
- az a távolság, ahonnan a program a markert még stabilan képes felismerni

- a felhasznált markerek bonyolultsága (szín, marker képének összetettsége, stb.)
- a marker mozgására való érzékenység.

Az első két tulajdonságot ezek közül pontosan, valamilyen mértékegység segítségével le tudtam mérni (cm), vagy egyértelműen eldönthető volt (adott markert felismer-e a program vagy sem), azonban a harmadik tulajdonságot csak körülbelül lehet egymással összehasonlítani, konkrét értéket nem tudtam hozzárendelni.

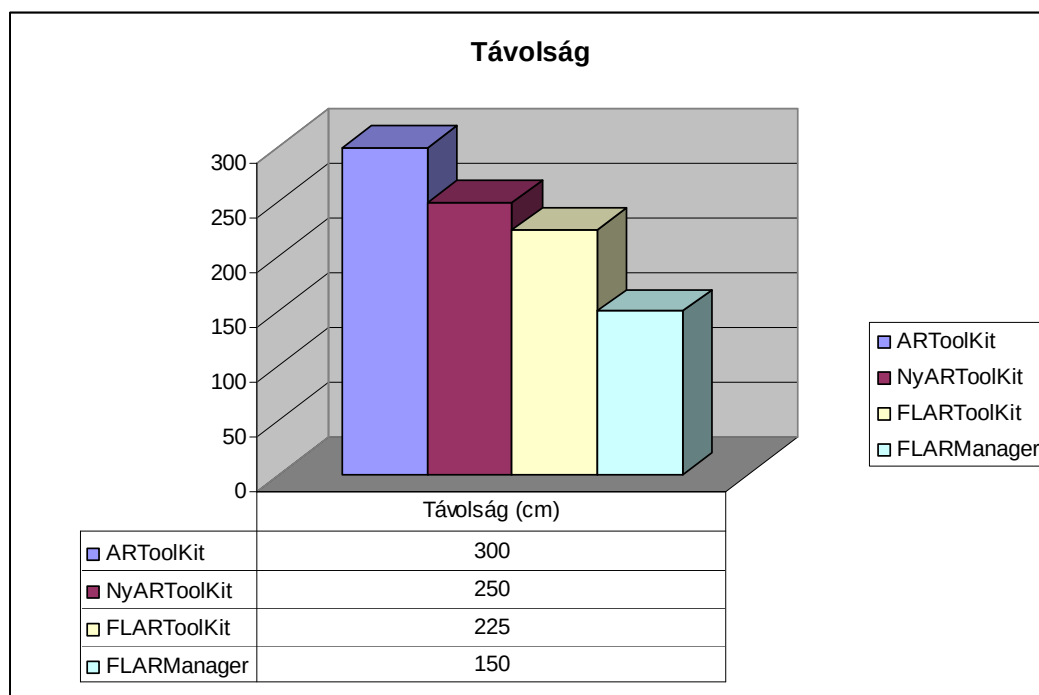
5.2.1. Távolság

Elsőként arra voltam kíváncsi, a programok milyen távolságból képesek felismerni a markert. Ez fontos tényező lehet például egy markerekkel irányítható rendszer esetén, ahol az egér helyett markert használunk, így távolról is irányítható a program. A fentebb leírtak szerint mindegyik program megegyező markert használ, ehhez egy igen egyszerű markert választottam, amely a 33. ábrán látható.



33. ábra - A programok által felhasznált marker.

A mérések – melyek azonos fényviszonynál, egymás után készültek – után a következő eredményeket kaptam:

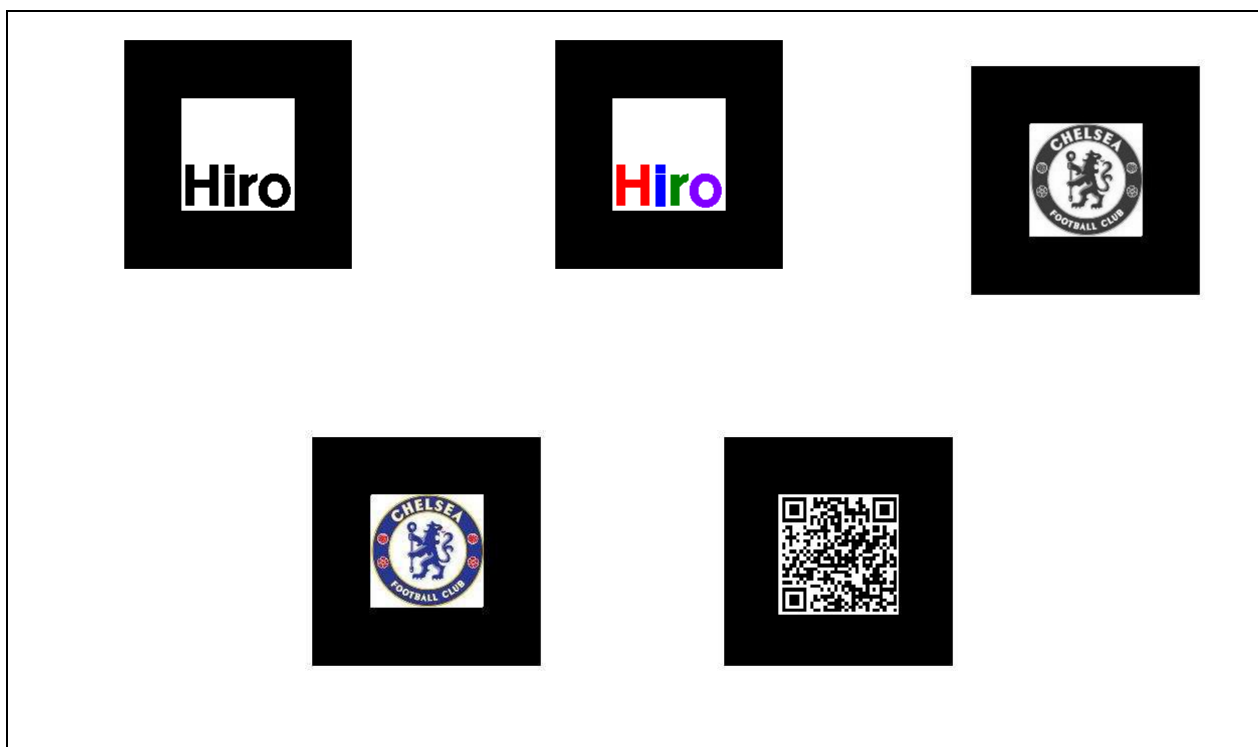


34. ábra - Markerfelismerés távolsága.

Az eredményekből látható, hogy ugyanazt a markert legtávolabbról, stabilan, mozgatóskor a modellt a markerről nem eltüntetve az ARToolKit ismeri fel. Három méteres távolságból is képes követni a földgömb valós időben a markert, akár annak mozgásakor is. Természetesen ilyenkor a távolsággal arányosan csökken a markeren megjelenített földgömb mérete is, azonban a programban a földgömb sugarát növelhetjük, így ilyen távolságból is a kellő méretűre nagyítható. Az ARToolKit-et követi fél méterrel lemaradva az NyARToolKit, és attól egy picit gyengébben teljesít a FLARToolKit (225 cm). Végül jócskán elmaradva a többi eszköztől, a FLARManager csak 150 cm-es távolságról képes stabilan felismerni a markert.

5.2.2. Markerek bonyolultsága

Mint arról a korábbi fejezetekben a saját markerek készítésénél szó volt, lehetőségünk van egyedi mintázatú markerek elkészítésére. Ennek köszönhetően különlegessé tehetjük a saját AR alkalmazásainkat. Ez ihlette a következő tulajdonság mérését, amelyben arra a kérdésre kapunk választ, hogy melyik fejlesztőeszköz milyen bonyolultságú markerekkel képes megbirkózni. Öt, különböző tulajdonságú markert készítettem el, amelyekkel ezt a jellemzőt teszteltem. Két-két marker a színezéstől eltekintve megegyezik, míg az ötödik egy QR kód.



35. ábra – A tesztelés során felhasznált markerek.

Egy fejlesztőeszköztől eltekintve (FLARManager) mindegyik környezet képes volt kezelni az összes markert, ami bízató jel azoknak, akik egyedi, akár színes markerekkel szeretnék feldobni alkalmazásukat. Az ötödik marker, az AR kódba ágyazott QR kód a két technológia ötvözésével az eddigieknél még hatékonyabb alkalmazást tesz lehetővé (ugyanis kihasználhatjuk a QR kódban kódolt tartalmat is egy-egy alkalmazásnál, mint például a korábban bemutatott AR ELTE IK esetében). A FLARManager az első két markert hibátlanul, stabilan felismerte, azonban a maradék hármat vagy egyáltalán nem ismerte fel, vagy csak rövid pillanatokra, huzamosabb használatra alkalmas módon nem.

5.2.3. Stabilitás

Az utolsó tulajdonság – amit nem tudtam számszerűsíteni – azt próbálja meg jellemezni, hogy milyen stabil, mennyire érzékeny a mozgásra az elkészített program. Az alkalmazás élvezhetőségének szempontjából igen fontos, hogy viszonylag szabadon, könnyeden mozgathatja-e a felhasználó a markert annak megkockáztatása nélkül, hogy eltűnjön a megjelenítendő modell a monitor képernyőjéről. Kísérletezésem során arra a tapasztalatra jutottam, hogy az ARToolkit és NyARToolkit nagyon stabil, messziről is igen gyors, hirtelen mozdulatokat tehetünk meg anélkül, hogy megkockáztatnánk a megjelenítendő virtuális elem eltűnését a marker helyéről. Hasonló tapasztalatra jutottam a FLARToolkit esetén is, bár ott egy kicsivel érzékenyebb már a mozgásra a program, de még mindig jól megfelel a célnak. A FLARManager esetében azonban már nem ennyire kedvező a helyzet, hajlamos hirtelen mozdulatra eltüntetni a virtuális elemet, különösen akkor, ha nincsen elég közel a kamerához a marker. A másik három eszköz esetén nem állt fent ez az eset kísérleteim során.

5.2.4. Összegzés

A mérések eredményeinek alapján elmondható, hogy mindegyik fejlesztőkörnyezetnek megvan a maga előnye, valamint a hátránya is. Az ActionScript nyelvű eszközök, különösen a FLARManager segítségével egyszerűen, gyorsan, röviden készíthetünk el látványos alkalmazásokat. Amennyiben valaki kevésbé jártas a programozásban, esetleg most tanulja azt, akkor az ő esetükben megfelelő eszköz egyszerű AR alkalmazások készítésére a FLARToolkit vagy a FLARManager. A gyorsaságért és egyszerűségért cserébe kevésbé stabil alkalmazást kapunk. Nagy előnye viszont, hogy képes videó-, és hangkezelésre is, míg a másik két eszközzel nekem eddig nem sikerült ilyen alkalmazást készítenem, valamint az interneten keresztül is beágyazható egy weboldalba az swf kiterjesztésű alkalmazás.

Az ARToolKit és NyARToolKit segítségével kissé hosszabb, bonyolultabb programokat készíthetünk, cserébe viszont nagy távolságról is stabilan, gyorsan fog működni, valamint az OpenGL és DirectX grafikus motorok jól dokumentáltak rengeteg példával, míg az ActionScript nyelvű eszközök által használt Papervision3D kevésbé ismert. Komolyabb, nagyobb lélegzetvételű projektekhez ezen eszközök valamelyikét ajánlanám.

6. Jövőbeli tervek

Dolgozatomat többféle módon is lehet a jövőben továbbfejleszteni. Az egyik ilyen lehetséges lépés nemcsak az 5. fejezetben leírt négy könyvtárcsomag segítségével elkészített program metrikákkal való mérése, hanem a többi, a Fejlesztőeszközök fejezetben ismertetett könyvtárra is elvégezni ugyanezen méréseket.

Egy másik lehetséges fejlesztés a kiterjesztett valóság mobiltechnológiával foglalkozó részét érintené. Olyan, a bevezetőben leírt pozíció és irány alapú AR alkalmazást lehetne készíteni, amely a jelenleg használt programoktól eltérően a webkamera képén megjelenítendő új információt nem egy helyen tárolt adatbázisból nyerné, hanem szemantikus webtechnológiát használna erre a célra.

7. Konklúzió

Dolgozatomban a szükséges irodalom feltérképezése és feldolgozása után bemutattam a kiterjesztett valóság történetét, fajtáit, valamint a szakirodalomból megismert, a hétköznapi élet különböző területein felhasznált gyakorlati alkalmazásokat. Ismertettem az AR egyik legnagyobb kihívását jelentő három dimenzióban való regisztráció fejlődését, bizonyos esetben a matematikai hátteret, valamint a regisztráció különböző fajtáit. Összegyűjtöttem és összehasonlítottam az általam jelentősebbnek tartott kiterjesztett valóság alkalmazások fejlesztéséhez való könyvtárcsomagokat, valamint leírtam a segítségükkel elkészített saját fejlesztéseimet. Négy kiválasztott könyvtárcsomaggal kifejlesztett alkalmazásokra különböző méréseket végeztem el. A kapott eredmények segítséget nyújtanak a programozóknak a megfelelő könyvtárcsomag kiválasztásában az elkészítendő projekt sajátosságainak függvényében.

Irodalomjegyzék

- [1] MILGRAM, P., TAKEMURA, H., UTSUMI, A. (1994). Augmented Reality: A class of displays on the reality-virtuality continuum. *Proceedings of Telem manipulator and Telepresence Technologies*
- [2] AZUMA, R. T. (1997): A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*
- [3] GERENCSÉR, P. (2010): Kiterjesztett valóság alkalmazások Android platformon. *TDK dolgozat*
- [4] PASARÉTI, O., HAJDÚ, H., MATUSZKA, T., JÁMBORI, A., MOLNÁR, I., TURCSÁNYI-SZABÓ, M. (2011): Augmented Reality in education. *INFODIDACT 2011 Informatika Szakmódszertani Konferencia*
- [5] SUTHERLAND, I. E. (1968): A head-mounted three dimensional display. *AFIPS '68 (Fall, part I) Proceedings of the December 9-11, 1968, fall joint computer conference, part I*
- [6] KOVÁCS, B. (2010): Érzékszervek párbeszéde - Az interaktív szonifikáció esztétikuma. *Doktori disszertáció*
- [7] FEINER, S., MACINTYRE, B., SELIGMANN, D. (1993): Knowledge-based augmented reality. *Communications of the ACM*, 36(7), July 1993, 52-62
- [8] STATE, A., HIROTA, G., CHEN, D. T., GARRET, W. F., LIVINGSTON, M. A. (1996): Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking. *Proceedings of SIGGRAPH96*, pp.429-446
- [9] REKIMOTO, J. (1998): Matrix: A Realtime Object Identification and Registration Method for Augmented Reality. *Proceedings of Asia Pacific Computer Human Interaction*
- [10] NEUMANN, U., YOU, S., CHO, Y., LEE, J., PARK, J (1999): Augmented Reality Tracking in Natural Environments. *Ohmsha and Springer-Verlag*, pp.101-130
- [11] KATO, H., BILLINGHURST, M. (1999): Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System. *In Proceedings of the 2nd International Workshop on Augmented Reality*
- [12] MARTÍN-GUTIÉRREZ, J., SAORÍN, J.L., CONTERO, M., ALCANIZ, M., PÉREZ-LÓPEZ, D.C., ORTEGA, M. (2010): Design and validation of an augmented book for spatial abilities development in engineering students. *Computers & Graphics*
- [13] ZHANG, Z. (1998): A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*
- [14] ERDŐDI, P. (2012): Portál fejlesztése kiterjesztett valóság befogadására. *Diplomamunka*

- [15] LEE, T., HÖLLERER, T. (2007): Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking. *11th IEEE International Symposium on Wearable Computers*
- [16] ROSENTHAL, M., STATE, A., LEE, J., HIROTA, G., ACKERMAN, J., KELLER, K., PISANO, E. D., JIROUTEK, M., MULLER, K., FUCHS, H. (2001): Augmented Reality Guidance for Needle Biopsies: A Randomized, Controlled Trial in Phantoms. *Lecture Notes in Computer Science Volume 2208/2001*, 240-248
- [17] HENDERSON, S. J., FEINER, S. (2009): Evaluating the Benefits of Augmented Reality for Task Localization in Maintenance of an Armored Personnel Carrier Turret. *Proceeding of IEEE International Symposium on Mixed and Augmented Reality 2009*, pp. 135-144
- [18] PAIR, J., WILSON, J, CHASTINE, J., GANDY, M. (2002): The Duran Duran project: the augmented reality toolkit in live performance. *Augmented Reality Toolkit, The First IEEE International Workshop*
- [19] MATUSZKA, T., ERDŐDI, P., CSEH, P. (2011): AR portál. *III. Oktatás-Informatikai konferencia tanulmánykötet*, 197-201 old. *ELTE PPK*, ISBN 9789633120378.
- [20] LIAROKAPIS, F., MOURKOUSSIS, N., WHITE, M., DARCY, J., SIFNIOTIS, M., PETRIDIS, P., BASU, A., LISTER, P. F. (2004): Web3D and augmented reality to support engineering education. *World Transactions on Engineering and Technology Education*. ISSN: 1446-2257
- [21] BERGIG, O., SOREQ, E., HAGBI, N., PEVZNER, K., LEVI, N., BLAU, S., SMELANSKY, Y., EL-SANA, J. (2011): Out of the Cube: Augmented Rubik's Cube. *International Journal of Computer Games Technology Volume 2011*
- [22] PORKOLÁB, Z., PATAKI, N., SIPOS., Á. (2006): Jelenleg használatos szoftvermetrikák összehasonlító elemzése.
- [23] PORKOLÁB, Z. (2002): Programok strukturális bonyolultsági mérőszámai, *doktori értekezés*
- [24] TEMPERO, E.: McCabe's Cyclomatic Complexity Number. *COMPSCI 702: Software Measurement*
- [25] SEREBRENIK, A. (2011): Software metrics.
<http://www.win.tue.nl/~aserebre/2IS55/2010-2011/10.pdf>
- [26] SZIRAY, J. (2009): Bevezetés a szoftvertechnológiába.