



GÁBOR DÉNES FŐISKOLA

Molnár Zsolt, Zátroch Zoltán,
Tusz Bettina:

HÁROMDIMENZIÓS ROBOTMODELLEZÉS

Konzulens:

Dr. Kopácsi Sándor

Tudományos Diákköri Konferencia

BUDAPEST

2011. november 16.

Tartalomjegyzék

1	Bevezetés.....	2
1.1	Mi az a 3D?	3
1.2	VirCA.....	3
1.3	VirCA köztögető.....	4
2	Hasonló kutatások a 3D internet területén.....	6
2.1	3DInternet	6
2.2	Robologix.....	6
2.3	A megvizsgált rendszerek összehasonlító elemzése	7
3	A modellezés.....	8
3.1	A környezet modellezése	8
3.2	A robot modellezése	8
3.3	A robot részei	9
3.4	Konvertálás ogre formátumba	13
3.5	A szoba berendezése.....	15
3.6	A robot összerakása	22
4	A robot mozgatása.....	25
4.1	Quaternion	25
4.2	A mozgatás lépései.....	26
4.3	A for ciklus működése	26
5	Az életre keltés	29
6	Összefoglalás	31
7	További terveink.....	32
8	Irodalomjegyzék	33
9	Ábrajegyzék	34
10	Rövidítések jegyzéke.....	35

Bevezetés

Munkánkat 2011 szeptemberében kezdtük el a háromdimenziós robotmodellezés témában, Dr. Kopácsi Sándor konzulensünk segítségével. TDK munkánk nagy részét a Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutatóintézetében (MTA SZTAKI) nemrég kialakított robotlaboratóriumban végeztük.

Célunk e dolgozat megírásával az, hogy ismertessük a TDK munkánk során elvégzett feladatainkat és a megvalósításhoz használt eszközöket. Bemutatjuk a VirCA virtuális kollaborációs rendszert, amely lehetőséget nyújt számunkra, hogy internet segítségével, akár földrészekkel távolabbról képesek legyünk irányítani egy adott robotot úgy, mintha az valóban a közelünkben lenne. Majd ismertetjük a VirCA lehetőségeit és előnyeit a többi hasonló rendszerrel szemben, később pedig bemutatjuk eddig elért eredményeinket a háromdimenziós robotmodellezés témában.

Feladatunk az volt, hogy elkészítsük egy Fanuc S 430i F típusú ipari robot és környezetének háromdimenziós modelljét, melyhez elsődlegesen a Google SketchUp 8 programot használtuk. Az elkészült modellt egy ún. VirCA kötözetben összekapcsoltuk más komponensekkel, majd életre keltettük – azaz mozgathatóvá tettük – a VirCA rendszerben. További terveink szerint a virtuális modellt összekötjük a valós robottal, így nem csak szimulációt végzünk, hanem valódi vezérlést is megvalósíthatunk.

TDK munkánkban olyan új eszközökkel és módszerekkel ismerkedtünk meg, majd alkalmaztunk, amelyek segítségével egymástól fizikailag távol levő gépeket összekapcsolhatunk, virtuális valóság alapú háromdimenziós szimulációt és vezérlést valósíthatunk meg az interneten keresztül. Weblapunkon (www.robotmodell3d.hu) megtekinthetők az elért eredmények, azok dokumentációja, ismertető a TDK munkánkról, valamint videók a robotról, környezetéről, a VirCA-ról és annak kötözetéről. Ezeket és más, későbbiekben ismertetett komponenseket összekapcsolva élhetjük át a háromdimenziós élményt a 3D barlangban vagy más, 3D-s megjelenítős segítségével. A weboldal háromdimenziós tartalma, és a dolgozat színes példányának háromdimenziós ábrái speciális, piros-kék lencsájú szemüveggel tekinthetők meg. A működő modellek, ill. a későbbiekben a fenti weboldal ábrái és demói aktív 3D-s kijelzőn, három dimenzióban jelennek meg.

1.1 Mi az a 3D?

A valóságban három dimenzióban látunk (magasság, szélesség, mélység). Ezzel különítjük el egymástól a tárgyakat, és ez így természetes számunkra. Manapság mindenhol árad felénk a „3D”. Három dimenzióban láthatjuk a filmet a mozikban, a televízióban, már mobiltelefonon is átélhetjük a „3D élményt”, az internet segítségével pedig bárhol a világon „körbejárhatjuk” az épületeket a Google Earth program [1] használatával. Ezt jelenti ma egy átlagember számára a „3D”, de tekintsünk ennél tovább. Mit tartogathat még számunkra a háromdimenziós élmény?

1.2 VirCA

Az MTA SZTAKI-ban lehetőségünk nyílt megismerni a VirCA (Virtual Collaboration Arena) nevű eszközt [2], amely egy háromdimenziós, internet alapú, interaktív virtuális valóság-rendszer, amelyben egymástól fizikailag távol lévő eszközöket jeleníthetünk meg, miközben ezen (virtuális és valós) eszközök együttműködését irányíthatjuk. A VirCA használata során virtuális helyiségekben eszközöket helyezhetünk el, amelyeket itt bemutathatunk, kezelhetünk.

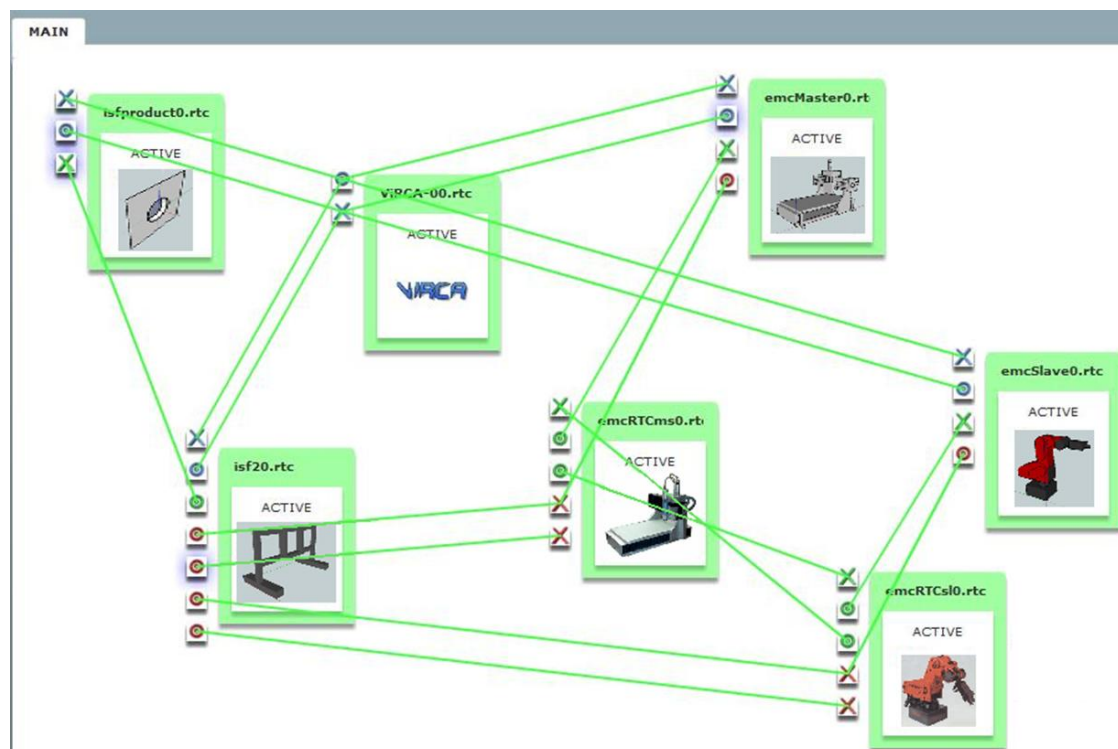
De a VirCA nem csupán játékszer. Ez a szó szerinti virtuális valóság, mely továbbfejleszti a valódi 3D élményt. A VirCA-val összekapcsolhatjuk a virtuális és valós világot. A tervezés során képesek vagyunk egyes alkatrészekből és szoftverekből új robotokat összeállítani, és a virtuális térben körüljárni, tesztelni. Ennek használatával sokkal gyorsabban és olcsóbban tudunk tervezés alatt álló robotokat tesztelni, és így már csak a biztonságosan és jól működő robotot kell elkészíteni. Kipróbálhatunk a virtuális térben hardvereket, eszközöket, algoritmusokat, melyek tesztelése a valóságban körülményes, költséges lenne. A VirCA képes ezt olcsón, veszélyek nélkül, precízen elvégezni. De ez nem csupán szimuláció! A virtuális rendszerben elkészített robotot képesek vagyunk összekapcsolni a valódi eszközzel, így teremtve meg a valós vezérlés lehetőségét. A VirCA használatával a robotot a virtuális modellel egy időben vagyunk képesek mozgatni. Ezzel a lehetőséggel a robotot akár földrészekkel távolabbról is irányíthatjuk. Miközben elvégeztetünk egy feladatot a robottal - például egy tárgyat áthelyezünk -, a dinamikai eseményt ezzel egy időben végigkövethetjük a virtuális modellen keresztül is. Ez nagyon fontos, hiszen a valóságban a robot nem képes minden körülményt figyelembe venni, a virtuális modell viszont képes más eszközökkel

intelligens módon együttműködni is, hiszen a virtuális eszközöket képesek vagyunk ily módon összekapcsolni.

1.3 VirCA kötöztető

Az előbbieken olvashatóak voltak a VirCA használatának főbb jellemzői, de hogyan is valósulnak meg ezek?

A VirCA ún. kötöztető moduljának (lásd 1. ábra) segítségével képesek vagyunk összekapcsolni a robot modelljét más komponensekkel. A képen látható módon a komponensek interfészeinek összekapcsolásával tudjuk egymáshoz rendelni a különböző elemeket. A beillesztett modellhez rendelhető például Wii kontrollert, melyről később lesz szó -, labdák, melyekkel tesztelhetünk, és a beszédfelismerés is. Abban, hogy a robot a beszédfelismerés hatására a valóságban is életre kel, rengeteg lehetőség nyílik. Nem csupán a későbbiekben ismertetett Fanuc robothoz hasonló eszközök esetében válhat hasznossá, hanem bármely más gép irányítását praktikussá teheti.



1. ábra: VirCA kötöztető

De térjünk vissza az ipari használatra, és az általunk modellezett robot lehetőségeire. A VirCA rendszer segítségével bárki képes vezérelni a modellezett eszközt, nem szükséges megtanulni a robot irányítását, ismerni a kezelőpanelt, amely egy hozzá nem értő számára nagy kihívást jelent. A beszéddel való vezérlés során parancsokkal szólíthatjuk fel, kérdezhetünk tőle, melyekre reagál. Az eszközök megosztják velünk képességeiket, így nyújtva segítséget a kezelő számára. Ilyen lehetőség például, hogy megkérdezzük a robotot: „Milyen funkcióid vannak?”. Erre válaszként felsorolja nekünk a lehetőségeket, melyek közül választhatunk. A szobán belül lévő robotok ismerik egymás lehetőségeit, ezáltal intelligens módon képesek együtt dolgozni. Ennyire egyszerűvé válik a robotok kezelése és eszközök, lehetőségek összehangolása a VirCA használatával.

2 Hasonló kutatások a 3D internet területén

A 3D internet témakörben kétféle kutatásról szereztünk tudomást TDK munkánk során, ezeket a külföldi fejlesztésű internet alapú virtuális rendszereket mutatjuk be ebben a fejezetben.

2.1 3DInternet

Manapság, az e-learning módszer elfogadottá válásával, rengeteg lehetőség nyílik tananyagok fejlesztésére. Ezt a lehetőséget aknázza ki a 3Dinternet nevű amerikai érdekeltségű cég [3].

Céljuk, hogy a kurzusokon használt PowerPoint előadásokat felváltsák interaktív tananyagokkal. Nem csupán látványossá és érdekessé válik így a tanulás, hanem animációkkal, valószerű grafikával élethűen ábrázolható bármely helyzet, melyet később a gyakorlatban lehet alkalmazni. Kis- és nagyvállalatok számára készítenek efféle segédeszközöket az oktatáshoz. Programtervezőik felmérik az adott cég igényeit, majd a programot elkészítve korszerű tananyagot készítenek.

A 3DInternet díjnyertes a tréning-szimuláció területén. Készít szimulációkat az elektronika, a hadászat, a gyártás és az oktatás témakörében is. Piacvezetőként új technikákat, a legkorszerűbb technológiát képesek alkalmazni, bár eltér az általunk ismertetett VirCA-tól. Mindkét találmány az internet segítségével működtethető, gépek, programok, algoritmusok tesztelhetők benne. Mégis más, hiszen a 3DInternet esetében nem beszélhetünk valós vezérlésről, csupán szimuláció történik.

2.2 Robologix

A kanadai Logic Design Inc. vezető szoftvergyártó cég a robot-szimuláció területén. Termékük, a RoboLogix [4] az egyik legkorszerűbb szimulációs program, melyet használhatunk tanításra, tesztelésre és egyéb funkciókra, amire az adott ipari robot képes.

Az alkalmazás tartalmaz olyan eszközöket, amelyekkel testre szabhatjuk a tesztelésre szánt robot környezetét, így készíthető el a valós szoba pontos mása. A RoboLogix futtatásakor megvizsgálhatjuk a robotot, körbejárhatjuk azt, majd tesztelhetjük a vezérlőalgoritmusokat. A kanadai cég programja ideális mérnökök, robottervezők, és diákok számára is, hiszen árkatóriájában a legjobb szimulációs

program, melyhez tartozik olyan modul, amivel a tesztelni kívánt teret élethűen ábrázolhatjuk. A szimulációs szoftver figyelembe veszi a mozgó robot hatókörzetét, így könnyen kiküszöbölhetőek az ütközések, vagy más efféle problémák. Ezzel a lehetőséggel segíthető a tervezési eljárás és a programfejlesztés biztonságossága. A RoboLogix képessé teszi a programozókat arra, hogy saját robotprogramokat írjanak és módosítsák a környezetet. A szimulátor képes tanulni is. A felhasználó megparancsolhatja a robotnak, hogy emeljen meg egy bizonyos tárgyat, majd térjen vissza eredeti helyére, vagy más, már előre programozott pozícióba.

Figyelemre méltó a RoboLogix sokrétűsége. Nem csupán tanulhatunk, taníthatunk a segítségével, hanem segítheti a programozók, tervezők munkáját is. A RoboLogix használatával időt és pénzt spórolhatunk, a hibák lehetősége pedig kiküszöbölhető.

2.3 A megvizsgált rendszerek összehasonlító elemzése

A megismert kutatások megkönnyítik életünket az oktatás, a programozás, a tervezés területén. A 3DInternet eszközének segítségével valós környezetben tanulhatunk. A RoboLogix-szal robotokat, algoritmusokat tesztelhetünk. A 3DInternettel ellentétben a programozók számára is alkalmazható a RoboLogix, hiszen ezzel a programmal képesek vagyunk módosítani, újraírni a környezetet és a parancsokat.

Mindkét program hasznunkra válhat, mégis más az általunk közelebbről is megismert eszközhöz képest. A VirCA-val is oktathatunk, tesztelhetünk, de ennek a rendszernek lényeges előnye az itt bemutatottakkal szemben, hogy nem csak szimulációra vagyunk képesek, hanem segítségével irányítani is tudjuk a modellezett eszközöket.

3 A modellezés

3.1 A környezet modellezése

TDK munkánk elsődleges tárgyát képező, modellezésre szánt robot az MTA-SZTAKI CIM Kutatólaborjában kialakított robotlaborban található. Első feladatunk az volt, hogy ezt a környezetet modellezzük. Lehetőségeinkhez mérten igyekeztünk mindent a lehető legnagyobb pontossággal elkészíteni. Erre a feladatra a Google SketchUp 8 programot választottuk [5]. A választás azért esett erre a programra, mert bár nem lehet benne olyan nagy pontossággal modellezni, mint más profi CAD programmal, viszont a kezelése egyszerű és nagyon gyors [6]. A rendelkezésre álló kevés idő miatt ez volt a legjobb megoldás. A modellezés pontossága még így is a mi céljainknak megfelelő, milliméter pontosságú volt.

A szoba modellezéséhez rendelkezésünkre állt egy alaprajz, ami nagyban megkönnyítette a munkát, mert nem kellett külön a helyszínen méréseket végezni. A robotlabor modelljét egy külön fájlban készítettük el, amely a room nevet kapta, de ez nem tartalmazza a berendezéseket, csupán a falakat az ajtót, az ablakokat és a falon található kapcsoló szekrényeket valamint a tűzjelző készüléket. A többi berendezés modelljét külön-külön fájlokba mentettük azért, hogyha a későbbiekben a robotlaboron belül egy bútor esetleg máshová kerül, akkor azon a modellben is könnyedén, gyorsan lehessen változtatni.

A laborban található asztalok, szekrények, lámpák a kontroller és a robot alapjának modellezéséhez azonban már a helyszínen kellett méréseket végeznünk. Mindezek szintén milliméter pontossággal történtek. A környezethez tartozó két asztalt desk_1, desk_2 névvel láttuk el. A robot irányító-panelet a controller nevet kapta. A három szekrényt a case_1 case_2 illetve case_3 nevet kapta. Ezen kívül a környezet részeként bekerült a modellbe a robot alapjául szolgáló megerősített talpazat, amely a fanuc_base nevet kapta. Az itt említett nevekre a *3.4 Konvertálás ogre formátumba* c. részben is hivatkozunk.

3.2 A robot modellezése

A modellezett robot egy Fanuc S-430i típusú ipari robot, amely 1300 kg tömegű, és 130 kilót képes megmozgatni. A robot háromdimenziós modellje, ahogyan a bútorok

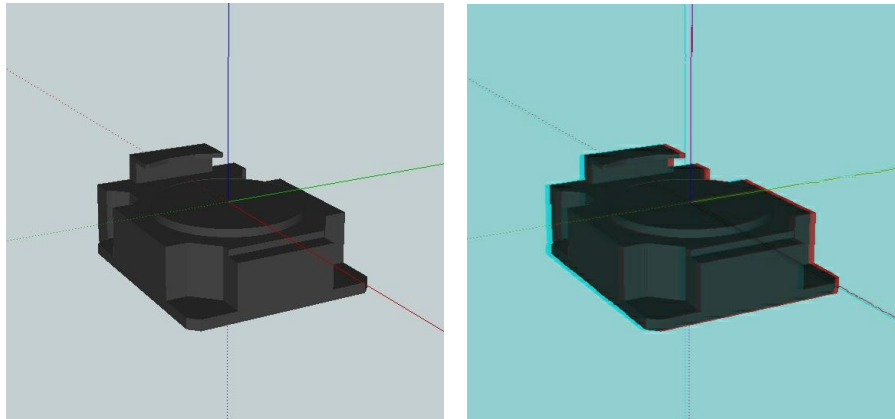
is, helyszíni mérések után készültek. Ez azonban a Solid Works [7] programmal készült, mert ezzel az alkalmazással a bonyolultabb, összetettebb robot egyszerűbben megrajzolható a legapróbb részletekig. Miután a robotot összeállítottuk a Solid Works-ben fel kellett azt darabolunk. Hogy miért volt erre szükség, azt a későbbiekben részletesen kifejtjük. A robotot hét darabra osztottuk fel aszerint, hogy a karok hány helyen forognak el, illetve ide tartozik még a robot alapja is, amely nem mozog, de a későbbiekben látszani fog, hogy ezt is szükséges volt külön darabként kezelni. Ezeket a darabokat fanuc_0 (a robot alapja), fanuc_1, fanuc_2, fanuc_3, fanuc_4, fanuc_5 és fanuc_6 névvel láttuk el. Ám ez a megoldás a későbbiekben több problémát is generált, amelyre a dolgozat következő fejezetében részletesen kitérünk.

3.3 A robot részei

Ebben a fejezetben bemutatjuk a robot részeit és azt, hogy a modellezés során a koordináta-tengelyeket hogyan állítottuk be. Ez fontos lesz a későbbiekben, amikor a robotot összeállítjuk a fanuc_test program segítségével. Azt, hogy a robot részeit miért így mentettük el, a dolgozat 3.6 *A robot összerakása* című részében magyarázzuk meg. Az 1. táblázat foglalja össze, hogy a robot hány helyen és milyen szögben képes elfordulni, valamint, erre milyen szögsebességgel képes.

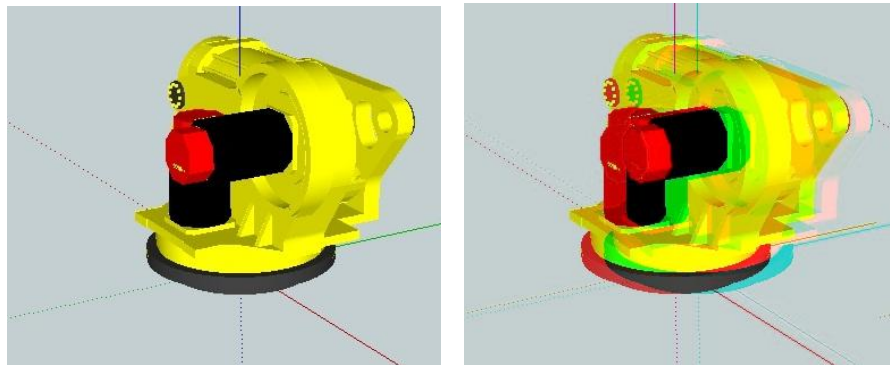
Az elfordulás szöge		Elfordulás szögsebessége	
fanuc_0	$\pm 0^\circ$	0 $^\circ/\text{s}$	0 rad/s
fanuc_1	$\pm 360^\circ$	210 $^\circ/\text{s}$	3.67 rad/s
fanuc_2	$\pm 125^\circ$	150 $^\circ/\text{s}$	2.62 rad/s
fanuc_3	$\pm 360^\circ$	210 $^\circ/\text{s}$	3.67 rad/s
fanuc_4	$+229^\circ, -133^\circ$	100 $^\circ/\text{s}$	1.75 rad/s
fanuc_5	$+75^\circ, -59^\circ$	110 $^\circ/\text{s}$	1.92 rad/s
fanuc_6	$\pm 180^\circ$	100 $^\circ/\text{s}$	1.75 rad/s

1. táblázat: A robot forgatási szögtartományai és szögsebessége



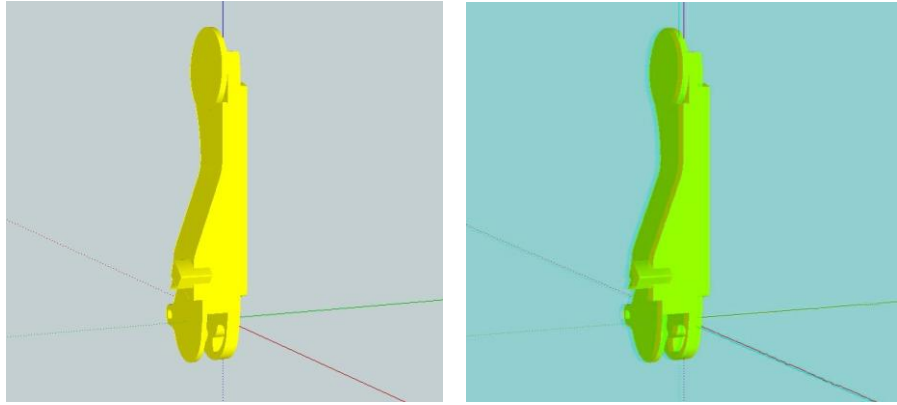
2. ábra: fanuc_0 elem (2 és 3 dimenzióban)

A 2. ábrán látható a robot talpa, amely a fanuc_0 nevet kapta. Ez az a része a robotnak, amely nem mozog. Az origót annak a körfelületnek a közepére állítottuk, amelyen a robot következő része fog illeszkedni.



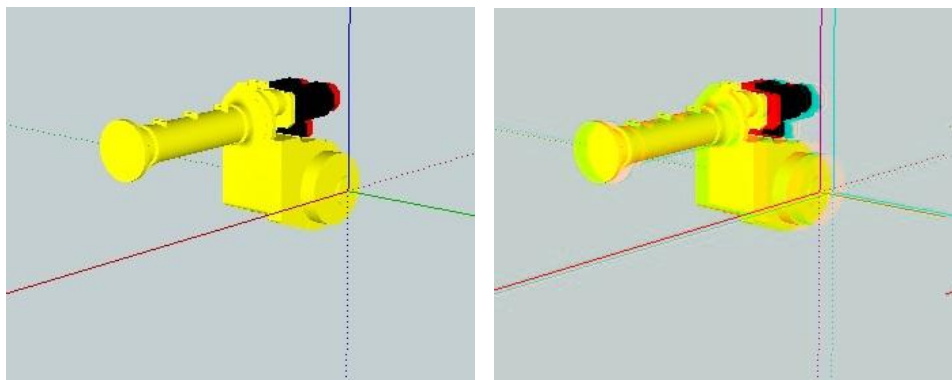
3. ábra: fanuc_1 elem (2 és 3 dimenzióban)

A robot első része, amely mozog, a fanuc_1 nevet kapta. Ezen az elemen két motor látható, fekete és piros színnel ábrázolva. A függőlegesen álló motor szabályozza az elem vízszintes elfordulását, a függőlegesen álló motor pedig a következő elem függőleges elmozdulását biztosítja. Az origó az elem alján található körfelület középpontjába lett beállítva.



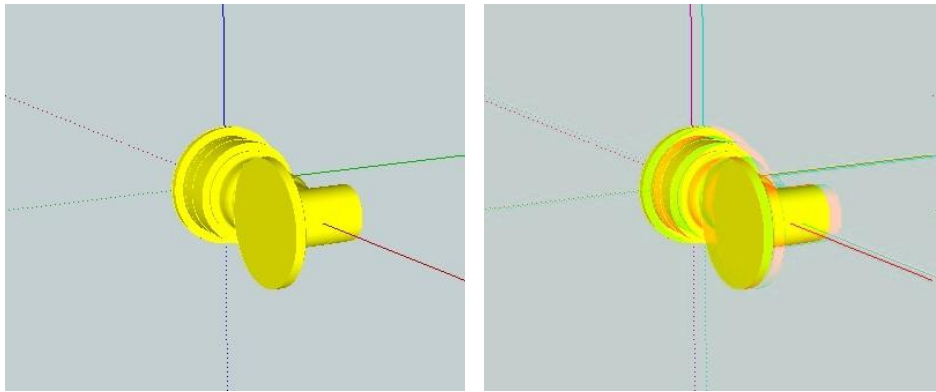
4. ábra: fanuc_2 elem (2 és 3 dimenzióban)

A következő mozgó elem a fanuc_2 nevet kapta (4. ábra). Az ábrán látható, hogy az elem origója az alsó tengely forgásközéppontjába lett elhelyezve. Az elem az x tengely körül fog forogni.



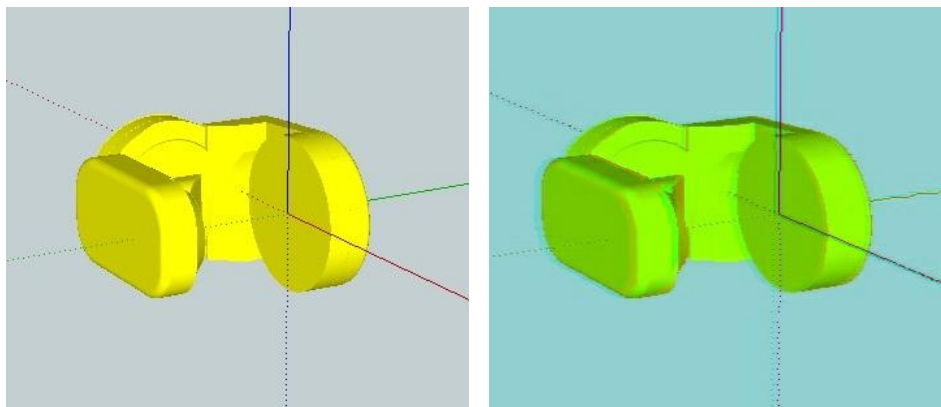
5. ábra: fanuc_3 elem (2 és 3 dimenzióban)

Ezt követi a fanuc_3 nevű elem (5. ábra), amely négy motorral van felszerelve. Egy motor szabályozza az elem függőleges irányú fordulását, egy a fanuc_4 forgását a z tengelyen, egy a fanuc_5 forgását az x tengelyen, végül pedig egy a fanuc_6 forgását szabályozza a z tengelyen.



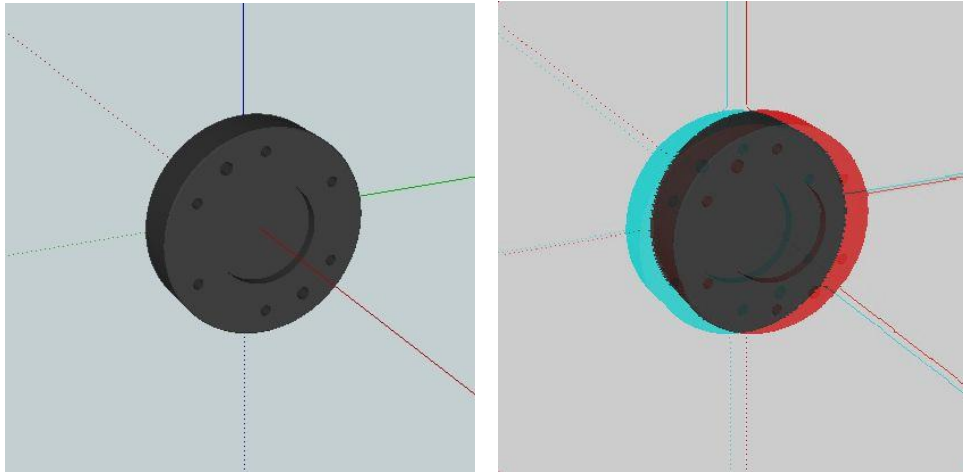
6. ábra: fanuc_4 elem (2 és 3 dimenzióban)

A következő elem a fanuc_4 (6. ábra), amely a **z** tengely körül képes elfordulni. Az origó ennél az elemnél azon körfelület közepére lett elhelyezve, amely a fanuc_3 elem végére illeszkedik.



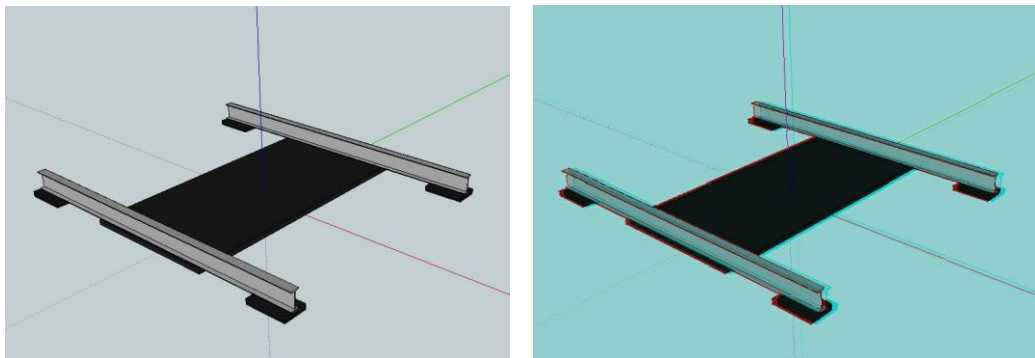
7. ábra: fanuc_5 elem (2 és 3 dimenzióban)

Az utolsó előtti elem a fanuc_5 (7. ábra). Ez az elem az **x** tengely körül képes forogni. Az origója annak a körfelületnek a közepére lett pozícionálva, amely illeszkedik a fanuc_4 elemhez.



8. ábra: fanuc_6 elem (2 és 3 dimenzióban)

Az robot utolsó eleme a fanuc_6 (8. ábra). Az origó az elem hátoldalának közepéhez lett pozícionálva. Ez az elem képes a **z** tengely körül forogni.



9. ábra: fanuc_base elem (2 és 3 dimenzióban)

A fanuc_base valójában nem a robot része, de szorosan kapcsolódik hozzá mivel a fanuc_0 elem erre van rögzítve.

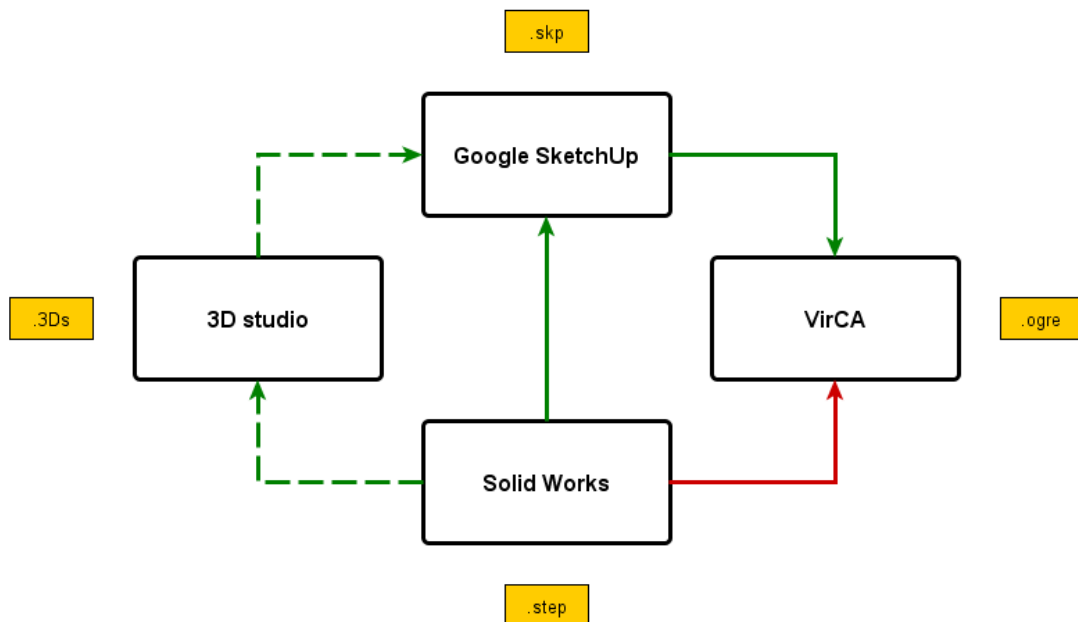
3.4 Konvertálás ogre formátumba

Az „ogre” [8] az a formátum, amelyet a VirCA a modellek megjelenítésére használ, tehát a konvertálás elengedhetetlen ahhoz, hogy a modellt összeállítsuk és működésre bírjuk. A konvertálás során keletkezik egy „mesh”, egy „material”, valamint melléktermékként egy „xml” fájl. A konvertálás során ebből az „xml” fájlból jön létre a „mesh” fájl. A három formátum közül a legfontosabb a mesh fájl, ez tartalmazza a modell összes tulajdonságát és objektumát, magát a testhálót, ezt

használja a VirCA a későbbiekben a modell megjelenítésére. Látható, hogy tényleges adat, amelyre szükségünk lesz a „mesh” és „material” fájlokban van. A „material” fájl adatokat tartalmaz a modell színéről, arról, hogy ez a modell milyen színű fényeket ver vissza, mennyire tükröződik a felülete, illetve ő maga bocsájt-e ki fényt.

A szoba és a berendezések modelljeinek konvertálása „skp” formátumból, amely a Google SketchUp program sajátja, egy lépésből megoldható egy ingyenesen letölthető beépülő modul [9] segítségével. Azért ezt az utat választottuk, mert a konvertálás során fontos szempont volt az is, hogy ha lehet, akkor egy olcsó megoldást találjunk erre, amelyet több gépen is, akár otthon is használni tudunk.

A robotmodell „step”-ből „ogre” formátummá alakítása nem volt megoldható, mert egy lépésből nem létezett olyan beépülő modul vagy program Solid Works-höz, amellyel ezt meg lehetett volna oldani. Több lehetőséget is számba vettünk, ezek közül a leggyorsabbnak és legegyszerűbbnek az a megoldás bizonyult, ha a „step” fájlt „skp”-ba konvertáljuk, majd annak egy megfelelően megválasztott pontját az origo-hoz pozícionáljuk. Ezek után a Google SketchUp beépülő modulja segítségével már könnyen „ogre” formátumba alakíthatjuk. Ezt mutatja a 10. ábra.

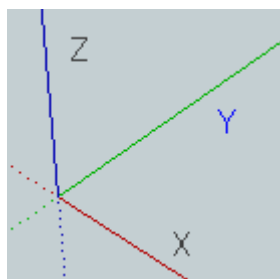


10. ábra: A konvertálás lehetséges lépései

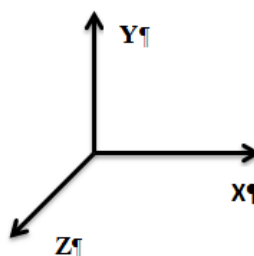
Mindeközben felmerült még egy probléma, a konvertálás nem működött tökéletesen, mert a „skp”-ba konvertált „step” fájlban a robot méretei egy ismeretlen

hiba miatt elvesztek, így a robot nem volt méretarányos a szobához képest. Ezt a problémát, mivel a robotot előzőleg lemértük és annak méreteiről dokumentáció is a rendelkezésünkre állt, Google SketchUp programban úgy oldottuk meg, hogy átméreteztük a darabokat, majd az így már megfelelő méretű modellt exportálhattuk „ogre” formátumba. A későbbiekben is lett volna lehetőség arra, hogy a robotot átméretezzük, azonban ezt a megoldást láttuk gyorsabbnak és egyszerűbbnek.

A konvertálás során figyelniünk kellett még egy nagyon fontos dologra, a tengelyekre. Az ogre koordináta-rendszere ugyanis nem egyezik meg a Google SketchUp koordináta-rendszerével. Ezt mutatja a 11. ábra és a 12. ábra. A különbség az, hogy az egyes tengelyek máshogy állnak, de a két rendszer közötti konverzió az x tengely körül történő 90° -os forgatással egyszerűen elvégezhető. Ennek a későbbiekben lesz nagy jelentősége, amikor a labor berendezéseit helyezzük el a VirCA rendszerben. Ezért amikor a szobához tartozó bútorokat elmentettünk és még mielőtt azt ogre formátumba konvertáltuk volna, körültekintően meg kellett választanunk, hogy a modell mely pontja kerüljön a koordináta-rendszer origójába. A későbbiekben ugyanis ez az origó lesz az a pont, amelyet majd tologatnunk kell a különböző tengelyeken ahhoz, hogy a berendezéseket elmozgassuk a számunkra megfelelő helyekre a szobán belül.



11. ábra: SketchUp tengelyek



12. ábra: Ogre tengelyek

3.5 A szoba berendezése

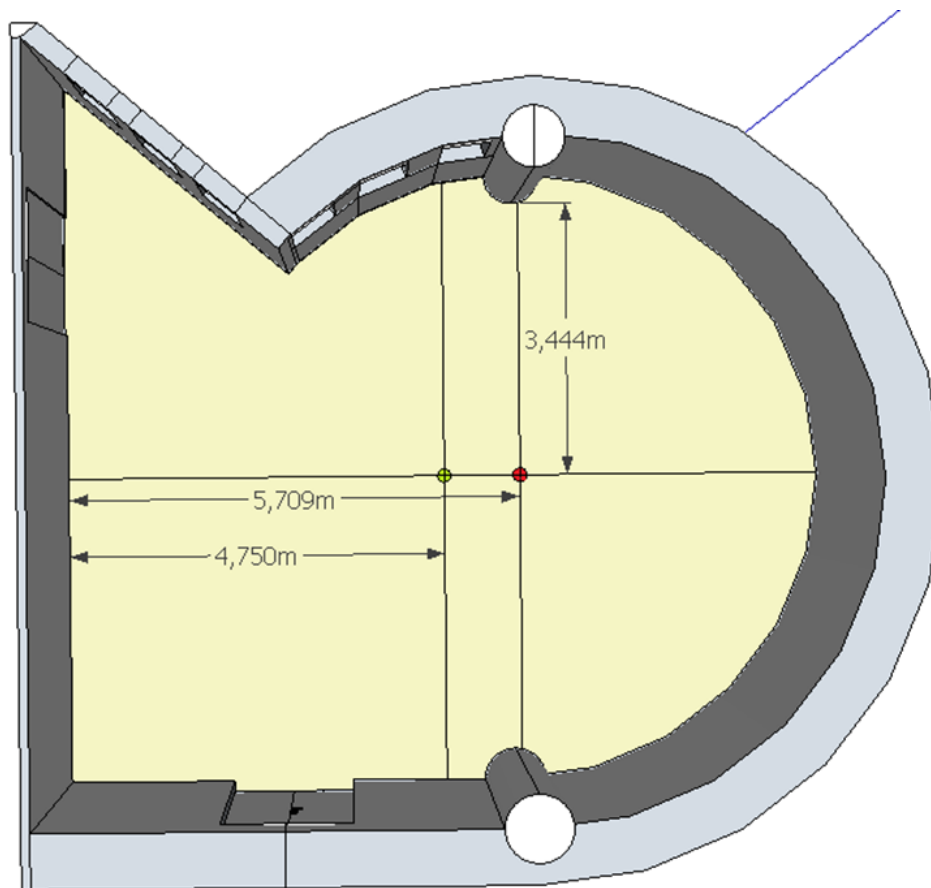
A VirCA-ban a szoba egy 3D-s környezetet jelent, amely tartalmazza a már említett tárgyakat, textúrákat és egy ún. dotScene „scene” fájlt, amely leírja a tárgyak elrendezését. A dotScene csak a tárgyak elhelyezkedéséről tartalmaz információkat, a tényleges adatokat a testhálóról és annak színéről a „mesh”, illetve a „material” fájlok tartalmazzák. Így a „scene” konfiguráció azokat az adatokat foglalja magába,

amelyekből kiderül majd, hogy melyik tárgy hová kerül a szobában. A roomname.scene fájl szintaktikája többé-kevésbé megegyezik az XML szintaktikájával. A létrehozott „roomname”.scene fájlt a Media/rooms mappából tölti be a VirCA, és ennek az eredményét fogjuk látni a számítógép képernyőjén.

A szoba összerakásában először a szoba origóját kellett meghatározni, ami nagyon lényeges, mert ehhez az origóhoz képest fogunk elhelyezni minden további elemet a szobában. A mi estünkben egy olyan helyzet állt elő, hogy a robotlabor falának egy bizonyos szakasza félkör alakú volt. Ilyen falszakaszt a VirCA roomname.scene fájljában nem lehet előállítani, ugyanis ott csak síkfelületek hozhatók létre. Ezért azt a módszert választottuk, hogy létrehozunk egy megfelelő méretű, téglalapok által határolt szobát, amelyet akkorára méreteztünk, hogy a Google SketchUp-ban modellezett robotlabor beférjen úgy, hogy annak origója és az általunk téglalapokból létrehozott szoba origója egy pontba essen. A robotlabor origójának azonban nem a labor valódi origóját választottuk, hanem azt a pontot, amely a robot talpának a középpontja. A 13. ábra szemlélteti a labor valódi origója (bal oldali, zöld pont) és az általunk választott origó (jobb oldali, piros pont) közötti különbséget.

Azért, hogy a szoba valóban megjelenjen a VirCA rendszerben az ogre formátumba konvertált berendezésekkel a következő lépéseket kellett megtennünk:

1. A.mesh fájlt bemásoltuk a VirCA mesh könyvtárába.
2. A.materials fájlt bemásoltuk a VirCA materials könyvtárába.
3. A „scene” fájlban létrehozunk egy új node-ot, és megadjuk a pozícióit az **x**, **y**, **z** tengelyeken, valamint amelyik elemnél szükséges, ott forgatást is beállíthatunk.

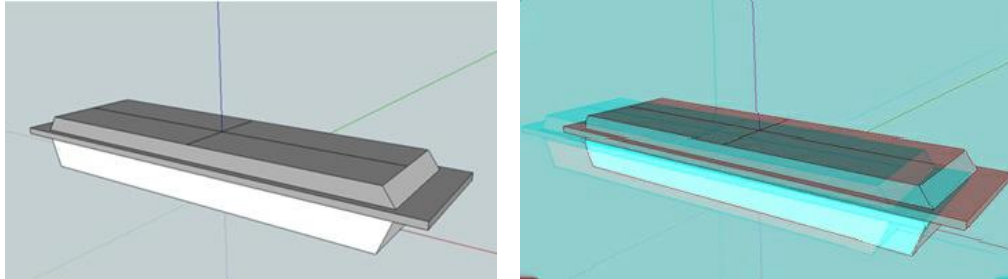


13. ábra: A labor modellje felülnézetből

Miután ezeket a lépéseket megtettük, már csak el kell mentenünk a változásokat a .scene kiterjesztésű fájlban, és el kell indítanunk a VirCA rendszert, amelyben ezek után már megjelenik az üres robotlabor.

A következő feladatunk 12 darab lámpa (lásd 14. ábra) elhelyezése volt. A lámpák helyzetét helyszíni mérésekkel határoztuk meg. A pontos pozíciókat azonban számolással kaptuk meg úgy, hogy tudtuk a szoba magasságát, tehát (ogre-ben) az y koordinátát, amely állandó marad. A szoba szélességét és hosszát is tudtuk, így már csak fel kellett osztani megfelelő részekre, mivel a laborban, ahol a robot található, három sorban helyezkednek el a lámpák. A szoba hosszát három egyenlő részre osztottuk, így meg kaptuk az x koordinátát, mert egy lámpasorban négy darab lámpa van. Már csak a z koordinátát kellett meghatározni, vagyis a szoba szélességét. Itt a helyszíni mérési adatok alapján a két lámpa sor közti távolságot használtuk fel. A lámpasorok közötti távolságot elosztottuk kettővel és így kaptuk meg a szakasz felét, amelyet az origóra helyezve egyenlő távolságra kerültek a lámpák. A lámpa

modellezése és konvertálása után azt is pozícionálnunk kellett, az **x, y, z** koordináták mentén a megfelelő helyre tolva. Ezt szintén a már előzőleg említett `.scene` fájlban tettük meg.



14. ábra: A lámpa (2 és 3 dimenzióban)

Van azonban a lámpáknak még egy nagyon fontos tulajdonsága, ez pedig a fényük. Azzal, hogy a node-okat létrehozuk a „`scene`” fájlban, csak a grafikus modelljüket helyezzük el, ettől azok nem fognak világítani. Ahhoz, hogy világítsanak, magának a fényforrásnak is létre kell hozni egy külön node-ot, amelynél a pozícionálás az előbbieket szerint történik. A fénnel kapcsolatos beállításokat a következőképpen tesszük meg:

1. Megadjuk a lámpa típusát. A mi esetünkben ez a „`spot`” lesz.
2. Beállítjuk a fény színét.
3. Beállítjuk, hogy a fény mennyire tükröződjön.
4. Ezt követően a fénycsillapítás mértékét állítjuk be.
5. Végül pedig a kibocsájtott fény világítási tartományát adjuk meg.

A „`scene`” fájlban ez a következőképp néz ki:

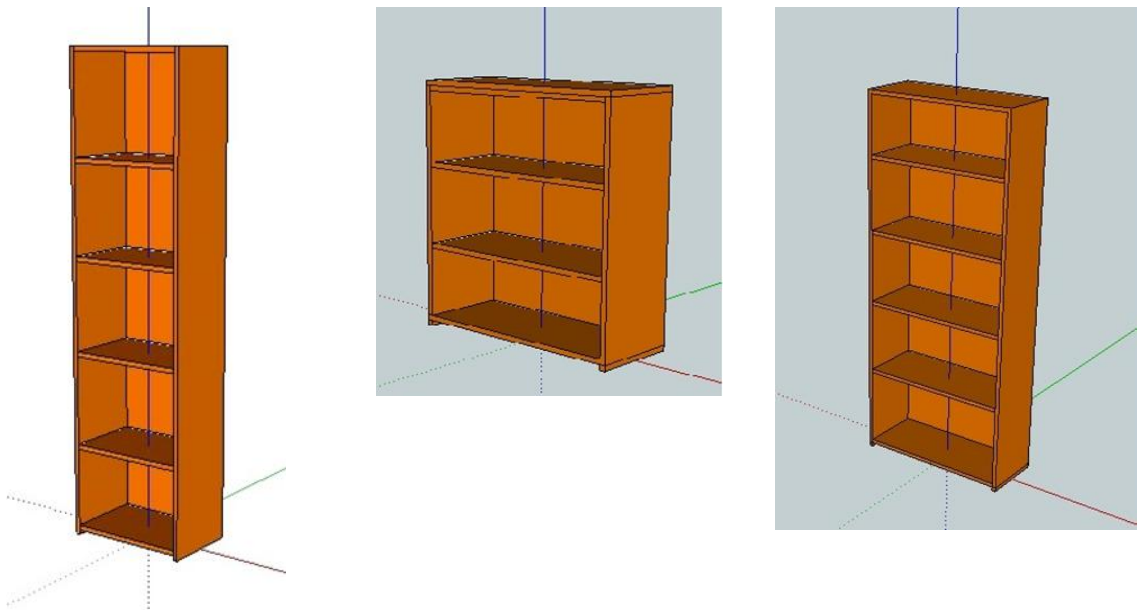
```
<node name="lamp0_light" >
  <position x="0" y="700" z="0" />
  <scale x="1" y="1" z="1" />
  <rotation w="1" x="0" y="1" z="0" />
  <light name="lamp" visible="true" type="spot"
castShadows="false" power="1">
  <colourDiffuse r="0.8" g="0.8" b="0.8" />
  <colourSpecular r="0.5" g="0.5" b="0.5" />
```

```

        <lightAttenuation      range="2000"      constant="1.5"
linear="0" quadric="0"/>
        <lightRange inner="30" outer="90" falloff="10"/>
    </light>
</node>

```

A lámpák elhelyezése után a következő lépés a szekrények pozícionálása volt. Ezek modellezése során - szintén előre gondolkodva - meghatároztuk azt, hogy melyik pontjuk kerüljön a koordináta rendszer origójához. Minden szekrénynél már arra is gondolnunk kellett, hogy nem csak pozícionálni, hanem forgatni is kell majd őket, hiszen ezek a szekrények a labor íves falfelülete mentén helyezkednek el. Ezért mindhármát célszerű volt úgy elmenteni, majd konvertálni, hogy a **z** tengely keresztülhaladjon a középpontjukon, így átláthatóbbá válik a forgatás (lásd 15. ábra). A szekrények elforgatása során a forgatás mértékét nem fokban kellett megadni, hanem egy 1-nél kisebb számot, ugyanis az 1 jelenti a teljes forgatást, vagyis 360°-ot.



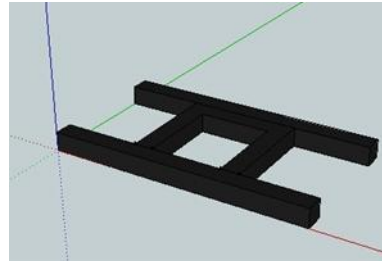
15. ábra: A szekrények és tengelyeik pozícionálása

A szekrények elhelyezése után a keret következett. A keret azt a célt szolgálja, hogy a Fanuc robot innen vegyen fel, illetve ide helyezzen bizonyos tárgyakat. A keret két részből áll frame_0 és frame_1 (lásd 16. ábra és 17. ábra), amelyek egymáson helyezkednek el. Az alsó keretet a modellezés során úgy pozícionáltuk és mentettük el, hogy annak bal hátsó lába, és a láb bal hátsó sarka helyezkedjen el az origóban. A

kisebb felső keretrészt pedig úgy, hogy a bal hátsó sarka legyen az origóban. Ezek forgatására nem volt szükség, csupán az x és a z tengelyeken kellett elmozgatni őket, illetve a keret felső részét feljebb kellett hoznunk az y tengely mentén, pontosan a keret alsó részének magasságával megegyező mértékben.

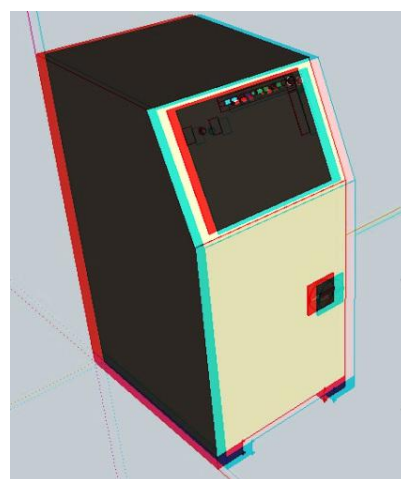


16. ábra: frame_0



17. ábra: frame_1

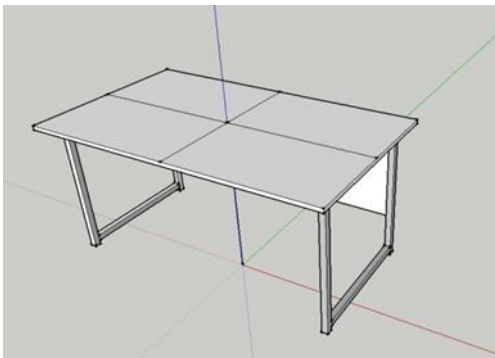
Mielőtt az asztalokat elhelyeznék, előtte a robot irányítóegységét, a controllert (lásd 18. ábra) helyeztük el. Ezt a modellelemet szintén úgy pozicionáltunk, mint ahogyan az előbb említett keret alsó részét (frame_0-t), vagyis a bal hátsó lábának bal hátsó sarkát helyeztük el az origóban majd így mentettük el és konvertáltuk ogre formátumba.



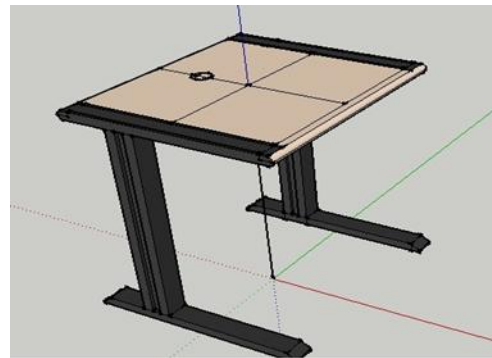
18. ábra: Robot kontrolller (2 és 3 dimenzióban)

Ezek után a szobában elhelyezés itt is az előzőekben leírt módon történt: miután elmozgattuk az x és z tengelyek mentén a kontrollert a megfelelő helyre, még el is kellett azt forgatunk.

A következő lépés az asztalok (19. ábra és 20. ábra) elhelyezése a laborban. Ezeket úgy pozícionáltuk, és mentettük el, majd konvertáltuk ogre formátumba, hogy az asztallapok közepét véve levetítettük azt a talajszintig, és ebbe a pontba helyeztük el az origót. Mindezek után az asztalok könnyen elforgathatóvá váltak, és el tudtuk tolni azokat a vízszintes tengelyek mentén, ahogyan azokat a többi node-nál is megtettük.

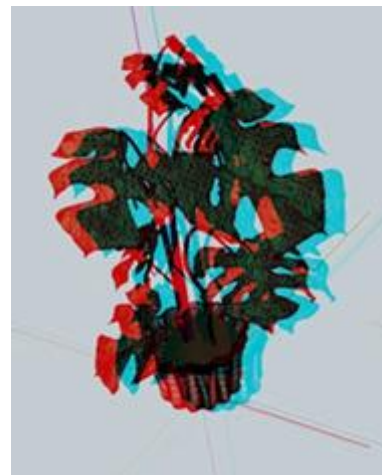


19. ábra: desk_1



20. ábra: desk_2

Ezek után már csak a növény elhelyezése volt hátra, amelyet a modellezés során úgy mentettünk el és konvertáltunk ogre formátumba, hogy az origó a virágcserep aljának középpontja legyen.



21. ábra: plant (2 és 3 dimenzióban)

E lépések elvégzése után a szoba készen állt, minden berendezés a helyére került, egyedül már csak a robot hiányzott belőle. Ezt azonban nem a szoba részeként kellett elhelyeznünk, így nem is egyszerűen új node-ként illesztettük be, ahogyan az eddig leírt modelleket, hanem a Microsoft Visual Studio 2008 fejlesztői környezetet használva C++ programkód segítségével, hogy az vezérelhetővé válhasson. A következő fejezetben erről lesz szó.

3.6 A robot összerakása

A robot összerakása Microsoft Visual Studio 2008 fejlesztői környezetben történt egy C++ nyelven írt program segítségével, amelyet `fanuc_test`-nek neveztünk. Ez a program állítja össze a robotot külön részenként, és ez a program képes mozgatni is egyben a már összeállított robotot.

Mivel a robot darabjait különálló részekként kezeljük, ezért minden egyes részt egy új ún. `CyberDevice`-ként kell regisztrálni. A `CyberDevice` olyan eszköz, amely csatlakozik a fő alkalmazáshoz, és a virtuális térben tudjuk bővíteni a tulajdonságokat, például hangvezérléssel. Így léteznek olyan eszközök, amelyek képesek hangfelismerésre, és ilyen módon irányíthatóak. A regisztráció folyamán az eszköz megjelenik a virtuális térben a megfelelő tulajdonságokkal.

A robot egyes darabjait ún. hierarchikus objektumként regisztráltuk a `fanuc_test` programba a `fanuc_base` elem kivételével, amely azért nem ilyen módon lett a modell része, mert a valóságban is ehhez az elemhez van rögzítve a `fanuc_0`. Ezért gondoltuk, hogy ez az elem is a robot részeként kerül bele a modellbe, mivel az szorosan kapcsolódik hozzá. A hierarchikus objektumok létrehozására azért volt szükség, mert így a létrehozott szülő objektum tulajdonságait (pl. pozícióját, orientációját) örökölni fogják a leszármazottai, azaz a robot egyes darabjai. A mi esetünkben itt arról van szó, hogy mozgás közben a robot egyes darabjai együtt fognak mozogni. Szülő objektumnak a `fanuc_0` (lásd 2. ábra) objektumot választottuk, mégpedig azért, mert ez a robot alapja, erre épülnek a további elemek. A következő elem a `fanuc_1` (lásd 3. ábra), amelyet szintén hierarchikus objektumként regisztráltunk, és örökli a `fanuc_0` tulajdonságait. Ezt követi a második mozgó darab, a `fanuc_2` (lásd 4. ábra), amely szintén örökli az előtte álló `fanuc_1` tulajdonságait, így haladva egészen az utolsó `fanuc_6` darabig. Ha esetleg a

későbbiekben a robotot át szeretnénk helyezni a szobán belül, akkor elég csak a `fanuc_0`-t áthelyezni, mert a robot többi része így vele együtt fog elmozdulni a `fanuc_base` kivételével, mivel az nem hierarchikus objektumként van regisztrálva. Az objektumok (`fanuc_0`, `fanuc_1`, `fanuc_2`, `fanuc_3`, `fanuc_4`, `fanuc_5`, `fanuc_6`) ezzel együtt el fognak mozdulni. Ezeket a lépéseket követve állítottuk össze a robotot.

A robot részeit egy eljárásban raktuk össze. Először, ahogyan a szobának is, úgy a programnak is meg kellett határozni az origóját. Erre azért volt szükség, mert a program erre a megadott koordinátára helyezi a `CyberDevice`-okat. Ezt a pontot mi úgy választottuk meg, hogy egybe essen a szoba origójával, amely egyben a robot talpának középpontja is. E pont koordinátái az $x = 0$; $y = 0$; $z = 0$. Így az első két elem beillesztése nagyon egyszerű lesz, hiszen csak a z tengelyen kell majd elmozgatni őket.

Legelsőnek a `fanuc_base` elemet raktuk a programba, mert ahogyan azt már említettük, erre épülnek rá a következő elemek, így ezt a robot részeként kezeltük. A `fanuc_base` origója egybe esett a szoba középpontjával. Ez nagyon szerencsés választás, mert a fent említett origóban lett elhelyezve, és csupán annyi módosításra volt szükség, hogy azt feljebb emeltük a y tengely mentén. A `fanuc_base` pozíciója így a következő lett $x = 0$, $y = 4$, $z = 0$ (mértékegység cm-ben). Az y értéke azért nőtt meg négygyel, mert a `fanuc_base` elem alján található egy 4cm-es talp (lásd 9. ábra). Természetesen a `fanuc_test` programban is hasonló lépések vannak, mint a „scene” fájlban a szoba összerakásánál. Ezek a lépések a következők:

1. Első lépésként meg kell adni a robotalap pozícióját:
 $x = 0$; $y = 4$; $z = 0$.
2. Második lépésként, be kell állítani a forgatás mértékét vagy is a Quaternion-t (lásd később, ebben az esetben nem kell forgatni.)
3. Harmadik lépésként létre kell hozni egy új `CyberDevice`-t
4. Az elkészült, és átkonvertált modell „mesh” és „material” fájljait a megfelelő helyre kell másolni.
5. Utoljára a `fanuc_base.mesh` fájlt be kell importálni a `fanuc_test` programba.

A fanuc_base elem után a fanuc_0 elem elhelyezése következett. A robot tényleges alapja a fanuc_base elemre van csavarozva. Nagyon fontos, hogy a fanuc_0 már egy hierarchikus objektumként lett létrehozva. Így ő lett a szülőobjektum, azaz a fanuc_1 őse. A fanuc_0 origója a tárgy tetején lévő körfelület közepére került, így ki kellett emelni a robotalapból, tehát csak az **y** koordináta lett megváltoztatva. A fanuc_0-nál is a fent említett lépéseket kellett végrehajtani, a következő változásokkal.

1. Első lépésként is meg kell adni a fanuc_0 pozícióját:
 $x = 0$; $y = 23,4$; $z = 0$. A fanuc_base-hez képest 23,4 centiméterrel van feljebb tolva ez az elem.
2. Második lépésként, be kell állítani a forgatás mértékét, vagyis a Quaternion-t =0.
3. Harmadik lépésként, létre kell hozni egy új CyberDevice-t.
4. Az elkészült és átkonvertált modell „.mesh” és „.material” fájljait a megfelelő helyre kell másolni.
5. Utoljára a fanuc_0.mesh fájlt importálni kell a fanuc_test programba.

A következő elemekkel (fanuc_2 fanuc_3 fanuc_4 fanuc_5 fanuc_6) végrehajtjuk ugyanezeket a lépéseket így összeállítva a robotot. Az elemek elhelyezését, a beállítandó paramétereket a következő, 2. táblázat tartalmazza.

név	mesh és material név	szülő- objektum	x [cm]	y [cm]	z [cm]
fanuc_base	fanuc_base.mesh fanuc_base.material	-	0	4	0
fanuc_0	fanuc_0.mesh fanuc_0.material	-	0	23,4	0
fanuc_1	fanuc_1.mesh fanuc_1.material	fanuc_0	0	0	0
fanuc_2	fanuc_2.mesh fanuc_2.material	fanuc_1	22,2	35,6	-12
fanuc_3	fanuc_3.mesh fanuc_3.material	fanuc_2	0	75	-10
fanuc_4	fanuc_4.mesh fanuc_4.material	fanuc_3	75,8	20,8	18,6
fanuc_5	fanuc_5.mesh fanuc_5.material	fanuc_4	19,2	0	-4
fanuc_6	fanuc_6.mesh fanuc_6.material	fanuc_5	20	0	5

2. táblázat: A robotösszerakás paraméterei

4 A robot mozgatása

Ebben a részben a Fanuc robot mozgatását fogjuk tárgyalni. Amint a `fanuc_test` program össze lett kötve a VirCA-val (lásd 1.2 fejezet) a kötőzgető (lásd 1.3 fejezet) segítségével, a robot megjelent a szobában, de még nem tud mozogni. Ahhoz, hogy a robot mozogni tudjon, még egy pár lépést végre kell hajtani.

Ahogy azt a 3.6 fejezetben leírtuk, a robotot azért kellett külön álló részekre szétszedni, hogy a `fanuc_test` programba mozgatni tudjuk. Elsőként a robot elemeit hoztuk létre a `fanuc_test` programban, és ezt kétféleképpen tettük: objektumként és hierarchikus objektumként hoztuk létre. A hierarchikus objektum azért fontos, mert az adott objektum csak az őiséhez képest tud elfordulni. A forgatások nagy részét úgynevezett Quaternion segítségével végeztük el.

4.1 Quaternion

A Quaternion egy számrendszer, amely kiterjeszti a komplex számokat. Jellemzője, hogy két Quaternion nem kommutatív. Leggyakoribb felhasználási területe az elméleti és az alkalmazott matematika, különösen a háromdimenziós számítógépes grafikában a háromdimenziós forgatások. A forgatás a következő képlet (lásd 1. egyenlet) segítségével működik.

$$[w, \ x, \ y, \ z] = \left[\cos \frac{\alpha}{2}, \ n_x \sin \frac{\alpha}{2}, \ n_y \sin \frac{\alpha}{2}, \ n_z \sin \frac{\alpha}{2} \right]$$

1. egyenlet: Forgatás Quaternion segítségével

A képletben α az elfordulás szöge. Az $\mathbf{n}_x$, $\mathbf{n}_y$, $\mathbf{n}_z$ a forgástengely, ehhez a tengelyhez képest van elforgatva a tárgy.

4.2 A mozgítás lépései

A hat szabadságfokú robot mozgása meglehetősen összetett. A `fanuc_0`-ás elem nem mozog, ez lesz az objektumhierarchia kezdőpontja. Ez az elem a robot alapja, és ehhez képest fog elforgogni a `fanuc_1`-es elem. A 3.6 fejezetben leírtuk, hogy az egyes elemek hány fokos szögben tudnak elfordulni és azt is, hogy mihez képest.

A mozgást a `fanuc_test` program tesztelés céljából több *for* ciklus segítségével valósítottuk meg. A mozgást a későbbiekben a valós robotvezérlés fogja irányítani figyelembe véve a robot tényleges pozícióit. A 14 körből álló belső *for* ciklus szerkezete a következő:

```
for (i=0;i<14;i++)
{
    angle += i*1;

    fanuc_1->setOrientation(Ogre::Quaternion(Ogre::Degree(-angle),
Ogre::Vector3::UNIT_Y ));
    mpAPI->refreshParts();
    sleep(0.5);
}
```

4.3 A for ciklus működése

A *for* cikluson belül létrehoztunk egy `angle` nevű változót, amelyben a szögek aktuális értékét fogjuk tárolni. A szög értékét úgy kapjuk meg, hogy az *i* ciklusváltozót megszorozzuk az elfordulás fokban mért lépésközzel (amely jelen esetben 1 fok), ezt hozzá adjuk az `angle`-hez, majd elvégezzük még egyszer ezt a műveletet és az eredményt hozzáadjuk az előző eredményhez (lásd 2. egyenlet és 3. táblázat).

$$c += a * b$$

2. egyenlet: A forgatás számítása

ahol:

a: szorzó

b: az elfordulás értéke (1 fok)

c: az elfordulás végsőértéke (fokban)

a*b	szorzás eredménye	c értéke
0*1	0	0
1*1	1	1
2*1	2	3
3*1	3	6
4*1	4	10

3. táblázat: A forgatás lépései

Ezek után a `fanuc_test` programba a `for` cikluson belül elforgatjuk a `fanuc_1`-es elemet az `y` tengely körül az `angle` értékével. Így a `fanuc_1`-es elem elfordul szépen 90° -kal. A `sleep` értékét változtatva tudjuk beállítani a forgatás sebességét.

Ez után következik a `fanuc_2`-es és `fanuc_3`-as elemek mozgatása. Ahogyan az előbb tárgyaltuk a `for` ciklus működését, ez itt is hasonlóképpen történt, azzal a különbséggel, hogy az elemek forgatása egyszerre történik. Ennél a résznél is egy `for` ciklus van, de itt már feltételes utasítások is szabályozzák a program lépéseit. A feltételes utasításokra azért volt szükség, mert így a két különálló részt egyszerre tudtuk mozgatni. Ebben az esetben `angle2` változót 0,5-tel szoroztuk meg, így az elfordulás szöge a felére csökkent. Ez egyenletesebb mozgást eredményez. Az első feltételes utasítás a `fanuc_2`-es elemre vonatkozik, amelyben a következő feltételeknek kell teljesülni, ahhoz, hogy ez a rész elforogjon. Feltétel a 0° -tól nagyobb és 28° -nál nem nagyobb szögértékek legyenek. Ez azt jelenti, hogy 28° -ot fordul el a `fanuc_1`-es elem. A második feltételes utasítás a `fanuc_3`-as elemre vonatkozik. Itt a feltétel az, hogy 0° -tól 51° -ig terjedő szögértékek. Tehát ez az elem 51° -ot fordul el. Erre azért volt szükség, mert a két elemet egyszerre akartuk mozgatni, és ezt úgy oldottuk meg, hogy egy cikluson belül állítottuk be a szögek értékét az elforduláshoz. Ezt mutatja a következő programrészlet:

```
for (j=0;j<10;j++)
{
    if(angle2<=28)
        angle2 += j*0.5;
    fanuc_2->
setOrientation(Ogre::Quaternion(Ogre::Degree(angle2),
Ogre::Vector3::UNIT_Z));
    mpAPI->refreshParts();

    if (angle3<=51)
        angle3 += j*1;
```

```

fanuc_3->
setOrientation(Ogre::Quaternion(Ogre::Degree(-angle3),
Ogre::Vector3::UNIT_Z));
mpAPI->refreshParts();

sleep(0.25);
}

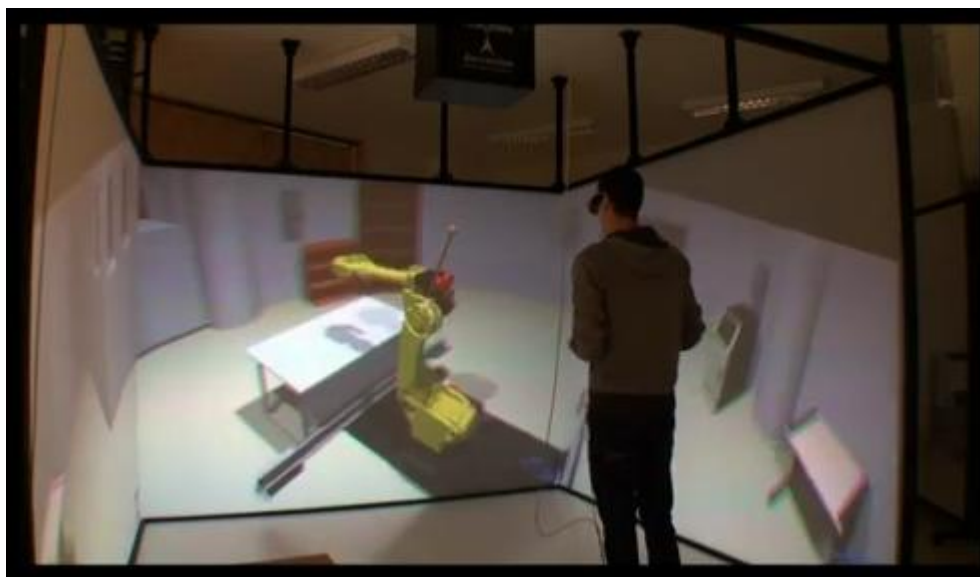
```

Utolsó elemként a fanuc_5-ös elemet mozgattuk meg. Ennek az elemnek a forgatása hasonlóképpen történt, mint a fanuc_1-es elemé. Természetesen ez az elem nem az **y** tengely, hanem a **z** tengely körül fordult el. A fanuc_5-ös elem 55° -ot fordult el.

5 Az életre keltés

Az előző fejezetekben megtett lépések után a munkánknak még nem lenne látható eredménye. Elérni azt, hogy a szoba modellje láthatóvá váljon és a benne lévő Fanuc robot megmozduljon a `fanuc_test` programmal, a VirCA köztetőjével és magával a VirCA rendszerrel együttesen érhetjük el.

Az életre keltésre az MTA-SZTAKI 3D barlangjában [2] került sor (lásd 22. ábra). Ez egy igen korszerű technikával felszerelt szoba, amely a TDK előadáson bemutatott, és a <http://www.robotmodell3d.hu> címen is megtekinthető videóban szerepelt. A 3D barlang 3 fehér vászonból készült fallal rendelkezik, ezeken a falakon jelenik meg a számunkra látható háromdimenziós kép. A képet hat darab projektor vetíti egy-egy tükörlapra, amelyet azok a vászonra tükröznek. Azért van szükség hat darab projektorra, mert az egy-egy falra kerülő képeket 2 darab projektorral vetítik ki. Így ha felvesszük a megfelelő 3D-s szemüveget, akkor 3 dimenzióban láthatjuk a falra vetített képet. A szoba tetején található egy mozgásérzékelő berendezés, amely követi a fejünkön lévő 3 dimenziós szemüveg mozgását (gyakorlatilag a néző mozgását), így a 3D barlangban valóban azt látjuk, és abból a szögből, ill. pozícióból, ahogy mozgunk, ahogy az a valóságban is történne. A 3D barlang igazi erejét két nagy teljesítményű külső Nvidia Quadro Plex típusú videokártya adja, amelyek 960 CUDA processzormaggal és 16 GB memóriával rendelkeznek.



22. ábra 3D-s szimuláció a SZTAKI 3D barlangjában

Mivel a kommunikáció a VirCA komponensei között internet alapú és közvetlen kapcsolatot igényel a számítógépek között, a VirCA kötöztetője nélkül a rendszer nem lenne működőképes, ezért szükség van arra, hogy létrehozzunk egy VPN (Virtual Privat Network) kapcsolatot. Erre a célra az OpenVPN nevű programot használtuk.

A következő lépés az, hogy a számítógépünkön elindítottuk a VirCA rendszert, majd a fanuc_test programot. Ahhoz hogy a robot is láthatóvá váljon, arra lesz szükség, hogy a fanuc_test programot és a VirCA rendszert összekössük a VirCA kötöztetőjével. A komponensek behelyezéséről és összekötéséről szintén készült felvétel, amelyet a TDK konferencián is bemutattunk, ill. megtekinthető a www.robotmodell3d.hu weblapon.

A VirCA kötöztető futtatásához egy böngészőre volt szükség, amelybe az indítás után következő címet írtuk: 192.168.1.1:8080/rtmext/rtcse/index.html. A kötöztető oldalsávjába betöltődő komponensek között láthatóvá váltak számunkra a fanuc_test program, a VirCA rendszer, valamint egy wii controller, amelyet arra használtunk, hogy a VirCA rendszerben a kamerát mozgassuk. Ezeket egymás után behúzzuk az üres felületre, majd ott összekötöttük azokat.

Miután összekötöttük a komponenseket azok még inaktív állapotban vannak, a rendszer még nem működőképes. Ehhez az „Action” menüben a „*Refresh the components of the system*” menüpontra kattintottunk. Az ezután felugró ablakban a Start gombra klikkeltünk. A komponensek azután aktívvá váltak. Visszatérve a VirCA rendszerbe a robot is megjelent a szobában és végezte beprogramozott feladatát.

6 Összefoglalás

Munkánk során a célul kitűzött feladatokat sikeresen megoldottuk. A robotlabor alaprajzának segítségével, laborban végzett mérések nélkül elkészítettük annak háromdimenziós modelljét. A robotlaborban végzett méréseket követően a bútorokat és a Fanuc robotot modelleztük a Google SketchUp 8, valamint a Solid Works programmal. A robotot részekre bontottuk a későbbi mozgítás céljából, valamint a koordinátarendszerek origójához pozicionáltuk a robot megfelelő darabjait, illetve a többi modellezett tárgyat is, hogy a későbbi elhelyezés egyszerűbb legyen. Ezek után a méretarányos modellt átkonvertáltuk a megfelelő „ogre” formátumra, amelyet a VirCA rendszer a megjelenítéshez használ.

A konvertálás után a szoba berendezése következett, amelyet a „scene” fájlban tettünk meg, amelynek felépítése az XML szintaktikájához hasonló. A node-ok létrehozásával és pozicionálásával a tárgyak a megfelelő helyre kerültek a szobában. Ezt a robot elhelyezése követte.

A robotot a fanuc_test program segítségével állítottuk össze. A darabokat, hasonlóan a szoba összerakásához, pozicionáltuk. Egyes részeit hierarchikus objektumként regisztrálva illesztettük egymásra, hogy a későbbi mozgítás során ezek együtt tudjanak mozogni. A robot mozgítása ugyanígy a fanuc_test program segítségével történt, amit egy sok körös ciklus segítségével oldottunk meg. A robot a vele szemben lévő keret és a tőle balra elhelyezkedő asztal között mozog.

Végül mindezek élete keltéséhez az elkészült modellt a VirCA rendszer segítségével, a fanuc_test programmal és a VirCA köztöztetőjével együtt az MTA-SZTAKI 3D barlangjában jelenítettük meg, ahol képesek voltunk az általunk modellezett robotlabort három dimenzióban bejárni, látva a robot működését.

7 További terveink

A dolgozatban bemutatott és megoldott feladatokkal munkánk nem ért véget. A továbbiakban is folytatjuk azt, és az eredményekkel részt kívánunk venni a következő (2012. évi) TDK konferencián. Azon kívül, hogy a robotlabort és a Fanuc robotot modelleztük, illetve a robotot mozgattuk, szeretnénk megvalósítani azt is, hogy a szimuláció a valós robotvezérléssel szinkronizálva működjön. A robotot a vezérlő programja segítségével mozgatjuk, és ezeket a mozgásokat a modellezett Fanuc robot a valódi robottal megegyező módon végre fogja hajtani. Célunk továbbá, hogy az itt bemutatott Fanuc robot a későbbiekben arra beprogramozva működjön együtt más konfigurációkkal is.

A robot-szimulációt szeretnénk kiterjeszteni valós alkalmazásokra is. Az első konkrét alkalmazásban a Fanuc robot a Rubik kockát fogja kirakni. Ehhez az MTA SZTAKI robotlaborjában elkészültek a szükséges eszközök (kockabefogó, látó- ill. alak felismerő rendszer), továbbá rendelkezésünkre áll a kockakirakó algoritmus. A kocka valós kirakását a robotvezérlő irányítja, miközben a VirCA rendszer a virtuális térben nyomon követhetjük a folyamatokat. Ehhez el kell készítenünk a bűvös kocka 3D-s modelljét, majd ezt a fentiekben leírt módon beépíteni a VirCA rendszerbe. Az alkalmazás továbbfejlesztéseként megvalósítható, hogy a kocka valós kirakását a VirCA rendszerben – akár hangutasításokkal – kiadott parancsok segítségével irányítsuk.

További tervezett alkalmazás – amely a robotlabor kialakításának egyik eredeti fő célja – az inkrementális lemezalakítás [13]. Ebben az alkalmazásban a Fanuc robot szolgálna ki két másik robotot, amelyek az inkrementális lemezalakítást végzik. Ennek során a Fanuc robot a nyers munkadarabot a megmunkáló robot elé helyezi, majd a megmunkálás befejezésekor onnan elveszi, egy tároló helyre lerakja, majd egy másik tároló helyről a következő nyers munkadarabot ismét a munkatérbe helyezi. A feladatunk ebben az alkalmazásban a Fanuc robot működésének modellezése és szimulációja lenne, ill. ezek összekapcsolása a már megvalósított inkrementális lemezalakító szimulációval [14].

8 Irodalomjegyzék

- [1] Google Earth: <http://www.google.com/earth/index.html>
- [2] VirCA - Virtual Collaboration Area: <http://www.virca.hu> letöltés: 2011. szeptember
- [3] 3DInternet: <http://www.3dinternet.com/> letöltés: 2011. szeptember
- [4] RoboLogix: <http://www.robologix.com/> letöltés: 2011. szeptember
- [5] Google SketchUp 8: <http://sketchup.google.com/> letöltés: 2011. október
- [6] PC World: <http://pcworld.hu/google-sketchup-8-kepben-3d-ben-20110804.html> letöltés: 2011. október
- [7] Solid Works: <http://www.solidworks.hu/> letöltés: 2011. október
- [8] Ogre formátum: <http://www.ogre3d.org/tikiwiki/Getting+Started> letöltés: 2011. október
- [9] Converter: http://www.di.unito.it/~nunnarif/sketchup_ogre_export/ letöltés: 2011. október
- [10] CyberDevice:
http://www.virca.hu/mediawiki/index.php/Main_Page#VirCA_Cyber_Devices
letöltés: 2011. október
- [11] Quaternion: <http://www.ogre3d.org/tikiwiki/Quaternion+and+Rotation+Primer>
letöltés: 2011. október
- [12] Nvidia Quadro Plex: <http://www.nvidia.com/page/quadroplex.html> letöltés: 2011. szeptember
- [13] Imre Paniti, CAD API based tool path control for novel incremental sheet forming, POLLACK PERIODICA 5:(2) pp. 81-90.(2010), DOI: 10.1556/Pollack.5.2010.2.8
- [14] J. Nacs, I. Paniti, S. Kopácsi: Incremental Sheet Forming in Cyberspace – a Process Oriented Cognitive Robotics Application, CogInfoCom 2011, 2nd International Conference on Cognitive Infocommunications in Budapest – July 7-9, 2011.

9 Ábrajegyzék

1. ábra: VirCA kötözgető	4
2. ábra: fanuc_0 elem (2 és 3 dimenzióban)	10
3. ábra: fanuc_1 elem (2 és 3 dimenzióban)	10
4. ábra: fanuc_2 elem (2 és 3 dimenzióban)	11
5. ábra: fanuc_3 elem (2 és 3 dimenzióban)	11
6. ábra: fanuc_4 elem (2 és 3 dimenzióban)	12
7. ábra: fanuc_5 elem (2 és 3 dimenzióban)	12
8. ábra: fanuc_6 elem (2 és 3 dimenzióban)	13
9. ábra: fanuc_base elem (2 és 3 dimenzióban)	13
10. ábra: A konvertálás lehetséges lépései.....	14
11. ábra: SketchUp tengelyek	15
12. ábra: Ogre tengelyek	15
13. ábra: A labor modellje felülnézetből.....	17
14. ábra: A lámpa (2 és 3 dimenzióban)	18
15. ábra: A szekrények és tengelyeik pozicionálása	19
16. ábra: frame_0	20
17. ábra: frame_1	20
18. ábra: Robot kontrolller (2 és 3 dimenzióban)	20
19. ábra: desk_1	21
20. ábra: desk_2	21
21. ábra: plant (2 és 3 dimenzióban).....	21
22. ábra 3D-s szimuláció a SZTAKI 3D barlangjában	29

10 Rövidítések jegyzéke

3D	három dimenzió
CAD	Computer Aided Design (számítógéppel segített tervezés)
CIM	Computer Integrated Manufacturing (számítógéppel integrált gyártás)
CUDA	Compute Unified Device Architecture
GDF	Gábor Dénes Főiskola
MTA SZTAKI	Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutatóintézet
skp	SketchUp formátum
TDK	Tudományos Diákköri Konferencia
VirCA	Virtual Collaboration Arena
VPN	Virtual Privat Network
XML	Extensible Markup Language