Képzeletbeli teremtmények irányítása Linuxszal

Vajon hogyan képes a beágyazott Linux kielégíteni a The Jim Henson Company robotjainak és számítógépes 3D-modelljeinek sokszínű, valós idejű igényeit?

A Jim Henson Company képzeletbeli karakterek megalkotójaként vált ismertté. Az egyszerűbb karakterek, mint a Muppet Show tagjai, nem igényelnek csúcsszintű műszaki eszköztárat, de a robotok – legyen szó akár kisegérről, akár sárkányról, nem is beszélve a 3D-s számítógépes grafikával életre hívott teremtményekről – már igen. Élőben, valós időben kell mozgatni őket, így látszólag kapcsolatba léphetnek a valódi színészekkel és filmre vehetők – csakhogy ehhez különleges műszaki kihívásoknak kell megfelelni.

Jim Henson eredeti célkitűzéseinek egyike az volt, hogy minden karaktert egy ember irányíthasson, ami így sokkal szabadabban viselkedhet, letisztultabb személyiséget kaphat; ezeket a célokat sokkal nehezebb elérni, ha egy-egy karakterrel többen dolgoznak. Bámulatos, hogy amikor végre sikerül életre kelteni egy ilyen karaktert, mindenki elfeledkezhet róla, hogy ki vagy mi irányítja valójában, és teljes természetességgel lépnek vele kapcsolatba. A színészek és nézők úgy kezdenek el beszélgetni egy kutyával, békával vagy hóemberrel, mintha ő is ember lenne.

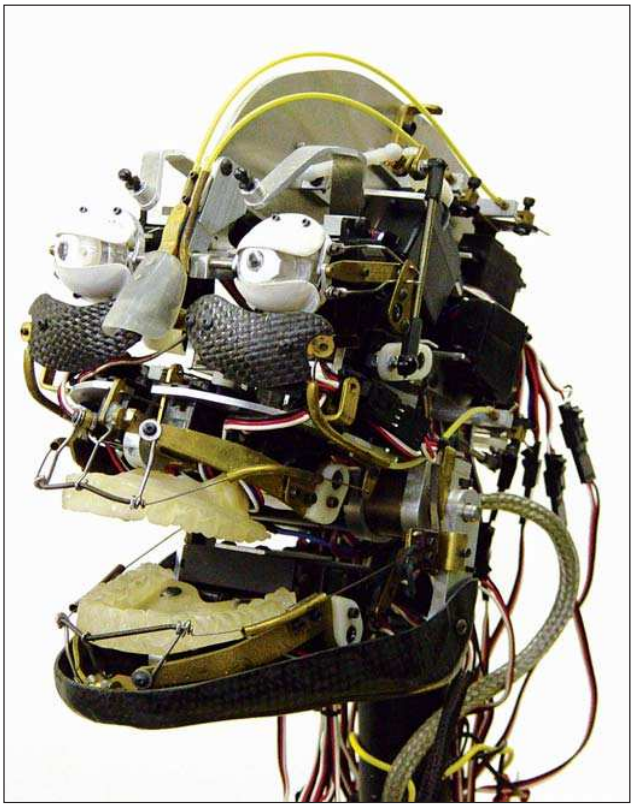
A szervomotorok az 1980-as évek elején terjedtek el a robotokban, ám a növekvő számú szervorobot irányítása egyre nehezebbé vált, így számítógépes vezérlést kezdtek el fejleszteni. Az elmúlt 15 év alatt a Jim Henson Creature Shop vezérlőrendszerek nemzedékeit fejlesztette ki, köztük olyat is, ami 1992-ben elnyerte a Technical Achievement Academy Award díjat. A legújabb Henson Performance Control System (HPCS) a korábbi rendszerekből a legjobb jellemzőket vette át, miközben az összes elérhető új számítástechnikai megoldást is magába olvasztja – így került képbe a Linux is.

A rendszer fejlesztése ***Jeff Forbes*** számítástechnikai-elektronikai igazgató vezetésével kezdődött 1998 elején. Az elképzelés az volt, hogy egy általános felépítésű rendszer a cég minden igényét ki tudná elégíteni. ***Steve Rosenbluth*** ekkor csatlakozott a tervezethez vezérlőrendszer-fejlesztőként, ***Michael Babcock*** pedig a multimédiás programozásért felelt. Igényeink azonban egyre sokrétűbbek lettek, és egyedül a Linux volt képes arra, hogy változásaikat különösebb gond nélkül kövesse.

A rendszernek kétféle hátteret, világot is támogatnia kellett: a robotizált bábokat és a számítógépes grafikát. A karakterek tehát valódi robotbábok, illetve poligonokból és képpontokból felépülő virtuális modellek is lehetnek. A két világ különkülön, de akár együtt is kezelhető.

Miután az egy bábuhoz tartozó programok a helyükre kerültek, akár a területen teljesen kezdők is néhány óra alatt begyakorolhatják a robotbábok irányítását. A beviteli eszközök kezelése a zenéléshez hasonlítható. A báb kezelője egy idő után eljut egy pontra, amikor már nincs tudatában annak, hogy pontosan mit is tesz – egyszerűen csak csinálja.

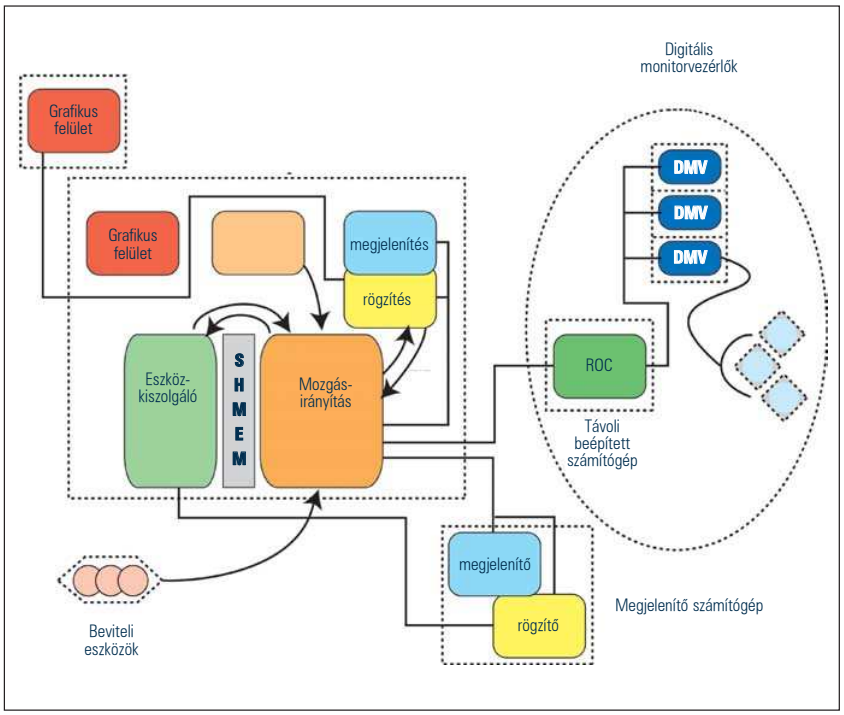
A Henson beviteli eszközök nem mozgásrögzítő megoldások. A mozgásrögzítés egyrészt csak a kezelő követésére képes, másrészt jellemzően nem programozható. A mozgásrögzítésnél a kezelőnek – avagy a színésznek – például a karját közvetlenül a karakter karjának feleltetik meg, a térdét a térdének és így tovább. A kezelő ezeket a megfeleltetéseket nem módosíthatja és nem bővítheti. A Henson beviteli rendszere viszont nem közvetlen megfeleltetést végez, és a felhasználó által újraprogramozható. A beviteli eszközök elvont értelmezést is támogassanak például lehetséges, hogy a művész mutatóujjának a mozgása a teremtmény egész arckifejezését a barátságosból arányosan a gúnyosba viszi át. A kezelő emellett könnyedén átprogramozhatja a karakter mozgását, akár jelenetek közben is. Mozgásrögzítéssel elég nehéz lenne például egy nyolckarú polipot eljátszani, ezzel a vezérlőrendszerrel viszont semmi gondot nem okoz.



Egy jellegzetes robotbáb feje szervomotorokkal és gépkarokkal

A vezérlő számítógép és a mozgásmotor

A rendszer lelke az RTLinuxot futtató vezérlő számítógép, amely a mozgásokkal kapcsolatos adatok feldolgozását és a robotokhoz való elosztását végzi. A vezérlő-számítógépen futó, a mozgásokat egyesítő algoritmust megvalósító folyamatot mozgásmotornak nevezzük, ezt Steve Rosenbluth írta C++ nyelven. A kezelők mozdulatait jeladók továbbítják a mozgásmotornak, és ez vezérli a hálózatba kapcsolt bábokat. A mozgásmotor az élő adatokat különféle algoritmusokkal dolgozza fel, és kiszámítja a gépkarok végleges helyzetét. A gépkarok úgymond a bábok izmai, a robotizált bábokban lehetnek elektromechanikus vagy hidraulikus szervomotorok, a számító-



A PCS/HDPS rendszer vázlata

gépes karakterben pedig a poligonháló alakváltoztatásait kell elvégezni. A mozgásegyesítési viszonyok alkalmazásból módosíthatók; lehetnek egy–több és több–egy jellegűek, az összetettebb keveréseket pedig felsőbb szinten lehet végrehajtani.
Az olyan fizikai hatások, mint a nehézkedés vagy a simítások, akkor adhatók hozzá a mozgásadatokhoz, amikor a mozgásmotor feldolgozza őket.

Az eszközkiszolgáló

A mozgáskeverő algoritmusokat az eszközkiszolgálónak nevezett folyamat által rendelkezésre bocsátott programok biztosítják. Az eszközkiszolgáló közvetlen hozzáféréssel rendelkezik a megosztott memóriában található objektumokhoz és adatcsomagokhoz, és létrehozza a kapcsolatot az élő mozgásadatok és a mozgásmotor által rajtuk futtatott algoritmusok között. Tervezését Michael Babcock végezte, és mint neve is utal rá, egy olyan aszinkron kiszolgálóról van szó, ami a vele grafikus felületen érintkező ügyfelekkel létesít kapcsolatot.

A grafikus felhasználói felület

A grafikus felület egy „nem túl vékony” ügyfél, ami egy foglalaton (socket) keresztül csatlakozik az eszközkiszolgálóhoz. A Michael Babcock által írt jelenlegi alkalmazás GTKmm alapú, és a ***Robert McNally*** tanácsadó által tervezett magas szintű felületterv alapján készült. A kiszolgálómódú felépítés lehetővé teszi, hogy a műszaki munkatársak vagy a bábkezelők a grafikus felületből akár több példányt is futtassanak.
A forgatások idején előfordul, hogy a műszaki munkatársak különféle kisegítő tevékenységekkel segítik a bábkezelőket – a hálózati felületnek hála ezt úgy tehetik meg, hogy közben nem kell másokat a rendszerből kirúgniuk. A grafikus felület és az eszközkiszolgáló egyedi protokollt használva tartja a kapcsolatot egymással. Annak lehetősége, hogy egy műszaki munkatárs a vezérlőrendszert távolról felügyelje, különösen fontos számunkra, hiszen a forgatások a világon bárhol lehetnek; a The Creature Shopban lévő támogatási személyzet mégis teljes hozzáféréssel rendelkezik, így el tudja hárítani a hibákat, illetve egyéb jellegű segítséget tud nyújtani.
Amint *az ábrán* is látható, a vezérlőrendszer mozgásmotor mögötti–alatti része módosulhat. A robotok vezérlését beágyazott processzorral rendelkező, távoli helyszíni számítógépekkel is el tudjuk végezni, a 3D-s számítógépes modellek életre keltése pedig egy „nézegető” háttérrendszerrel történik.

A távoli helyszíni számítógép

A mozgásmotor minden képkockánál kap egy pillanatképet az analóg–digitális átalakítóktól a fizikai beviteli eszközök állapotáról, a karakter beállításai alapján lefuttatja a mozgáskeverő algoritmusokat, majd az adatokat átadja a távoli helyszíni számítógépes ügyfeleknek.
Esetünkben a távoli helyszíni számítógépek DR DOS-t futtató beágyazott PC-k voltak. Steve Rosenbluth félig-meddig objektumközpontú C-kódot írt a távoli helyszíni gépekhez. A programozás során mindvégig szem előtt kellett tartania, hogy a kód gyorsasága és megbízhatósága alapvető fontosságú. A távoli helyszíni számítógépek protokollja lehetővé teszi, hogy egy összeköttetést – ami jelenleg egy RS-232 kapcsolat – több eszköz is használjon. A meglehetősen idősnek számító kapcsolattípus ugyan nem éppen korszerű eszköz, ám ezt a legkönnyebb üvegszál, rézkábel vagy rádiós összeköttetés felett használni. Az RS-232 valós idejű felület előre megjósolható jelekkel és késleltetésekkel, és nekünk erre van szükségünk. Ha egy robot vezetékkel rendelkezik, akkor üvegszál, egyéb esetben rádiós összeköttetés felett továbbítjuk az RS-232-jeleket.

Nézegetők

A számítógépes grafikai nézegető az a programmodul, ami a számítógépes modelleket leképezi és megjeleníti a képernyőn. Az ábrán szereplő nézegető lehet a számítógépes grafikai modellezőcsomagok és játékok motorjaiból álló gyűjtemény tagjainak egyike, ha elég gyorsan le tud képezni egy élő OpenGL-környezetet. A Henson Company háza táján ***Hal Bertram*** elektronikai részlegvezető a London Creature Shopban az 1990-es évek elején honosította meg a számítógépes karakterek készítését. Mostanában bizonyos PC alapú grafikai alkalmazásokat – Discrete 3D Studio Max, Side Effects Houdini, Kaydara Filmbox és Alias/Wavefront Maya – használunk, amelyekhez Michael Babcock készítette az irányítástechnikai beépülő modult.
Az, ami a vezérlésben egy gépkar, a számítógépes grafika világában egy skaláris csatorna adata, amely a 3D-poligonháló alakváltozását vagy módosulását írja le. A vezérlő-számítógép mozgásmotorja felől UDP-protokoll feletti kapcsolaton keresztül élő mozgásadatok kerülnek a nézegetőbe. A nézegetők a mozgásmotor nézőpontjából távoli helyszíni számítógépként viselkednek, legalábbis ugyanazt a protokollt használják. Kétprocesszoros AMD Athlon gépekkel körülbelül 100 képkocka/másodperc frissítést sikerült elérnünk, ami ugyanabban a jelenetben akár több karakter mozgatását is lehetővé teszi. ***Jeff Christie*** karakterfejlesztési műszaki igazgató egy 3D-modellbe-

állítás tökéletesítésével segített gyors, élethű megjelenést kicsalni a számítógépes grafikai modellekből – így vált teljessé a kép. A nézegető egyidejűleg több mozgásmotorral is kapcsolatot tud tartani, tehát egy olyan helyszínen, ahol több karakter is található, mindegyikük saját vezérlőrendszerrel és kezelővel rendelkezhet, akár egy hálózati játékban.

A felvevő

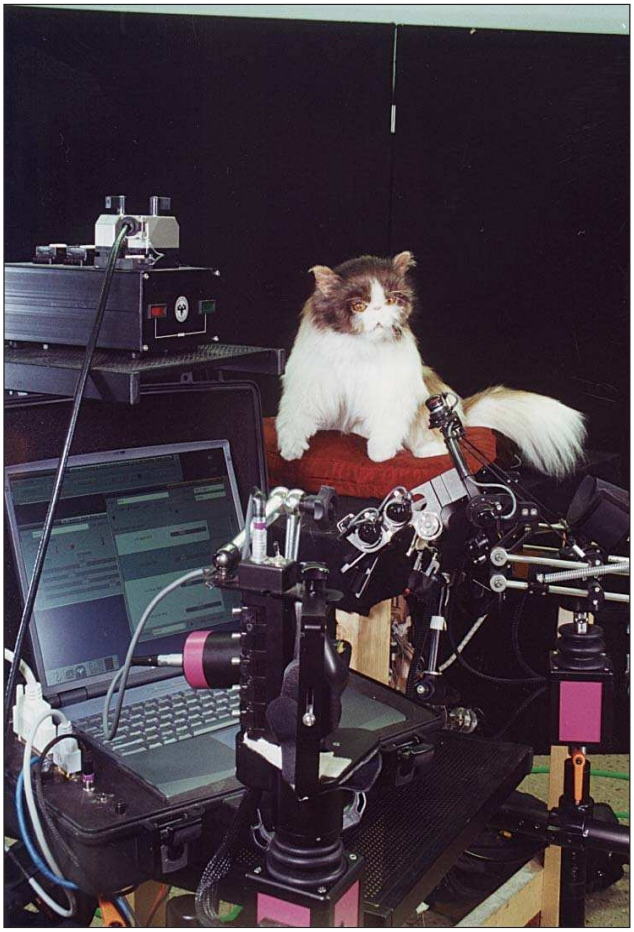
A mozgások és hangok rögzítését és visszajátszását egy nemlineáris multimédia-szerkesztő, a Michael Babcock által fejlesztett felvevő végzi. Az eszköz felépítését Michael és Steve dolgozták ki, és egy többszálú, a mozgásmotorral UDP-kapcsolatot tartó folyamatból áll. A felvevő a mozgásmotorral összhangban dolgozik, kimenetét ugyanis egy távoli helyszíni ügyfélgépnek adja át, illetve a tárolt adatok visszasugárzását is elvégzi a mozgásmotor felé, ami továbbítja őket a többi távoli helyszíni ügyfélgépnek. A hálózati szervezés lehetővé teszi, hogy minden folyamat – a többi zavarása nélkül – saját időzítéssel, be- és kiviteli jellemzőkkel rendelkezzen, ahogy az eszközkiszolgáló–mozgásmotor kapcsolatnál ezt láttuk.
Mivel a felvett mozgások élőben rendezhetők és visszajátszhatók, a művész egy-egy jelenetet úgy állíthat össze, mintha többsávos hangfelvételt készítene. Ez különösen a szájmozgás összhangba hozatalát igénylő helyzetekben hasznos, ahol a karakter ajkának mozgását más időpontban is ki lehet dolgozni, majd vissza lehet játszani, miközben a művész a bábu egyéb részeit élőben irányítja.
Dan Helfman egy hangfelvevő résszel járult hozzá az SDL-hez, a felvevőben használt nyílt forrású multimédia API-hoz.

A kapcsolatkezelő és a bábuügyfelek

A mozgásmotor kapcsolatkezelő modulja végzi a hálózati szórást és a távoli helyszíni ügyfelekkel való kapcsolattartást, legyen szó bármilyen hálózati kapcsolatról vagy megvalósításról. Így egy-egy művész egy karakternek akár több megvalósítását is vezérelheti. Előfordul például, hogy egy robotmacska és annak számítógépes modellje egyszerre mozog. Míg a karakter testét és arcát veszi a kamera, a számítógép mozgatja a száját, és a két képet azonnal össze is lehet vágni. Minden megtestesülés tehát azt végzi, amire a leginkább alkalmas. A valódi „lény” összetett megvilágítása, fizikája és a számítógépes szájmozgás az utólagos feldolgozás során, a film véglegesítése előtt tovább finomítható. Az élő előnézet lehetővé teszi, hogy a rendező a bábok mozgatását közvetlenül irányítsa, a színészek pedig bábtársaikkal kapcsolatba léphetnek.

Feldolgozás

A mozgásmotor, az eszközkiszolgáló, a grafikus felület és a felvevő feladatkörei élesen elkülönülnek egymástól. Mivel a bonyolult multimédiás és hálózati modulok olyan programmegoldásokat igényelnek, amelyek kedvezőtlenül hathatnak a feldolgozás ütemezésére és üzembiztosságára, Steve Rosenbluth és ***Tim McGill*** olyan rendszert hozott létre, amely egyfajta falat képez a mozgásmotor körül. A cél az volt, hogy a mozgásmotor a lehető legegyszerűbb legyen, így folyamatos működését a lehető legkevesebb dolog veszélyeztesse. Az eszközkiszolgáló – amelynek nagyra és bonyolultra hízása valószínűsíthető – ezzel szemben leállhat vagy újraindulhat, ez azonban nem befolyásolja a mozgásmotor működését. A grafikus felület szintén szükség szerint indítható és állítható le, ez sem az eszközkiszolgáló, sem a mozgásmotor, sem a felvevő működését nem érinti. Ennek érdekében a rendszer folyamatmodulokból épül fel, amelyek Unix IPC-hívásokkal és hálózaton keresztül tartják egymással a kapcsolatot.
Az eszközkiszolgáló és a mozgásmotor egy közös, megosztott System V memóriablokkal rendelkezik, így a legfontosabb adatobjektumok módosítása haladéktalanul elvégezhető. Az ütemezés szempontjából létfontosságú üzeneteiket két FIFO-n keresztül továbbítják. A mozgásmotor és a felvevő az adatokat UDP hálózati foglalatok használatával, adatfolyamok révén valós időben továbbítja egymásnak. A mozgásmotort az egyik legfontosabb elem, meghibásodása kellemetlen következményekkel járhat. Ha forgatás közben következik be a leállás, a



PCS összeállítás robotizált bábok irányításához, a bemeneti vezérlés és a grafikus felületet futtató hordozható vezérlőszámítógép

költségek óránként több ezer dollárra rúghatnak. A filmipar másik jellegzetessége, hogy a színészek és a segítők közeli kapcsolatban lehetnek a gépi karakterekkel. Elég érdekes lenne, ha egy robotkutya addig harapná valamelyik színészt, míg a műszaki munkatárs be nem jelentkezik, és újra nem indítja a megfelelő programot. Ez az, amiért a grafikus felület és az egyéb nélkülözhető részek kikerültek a mozgásmotor kódjából. Mivel rendkívül kényes körülmények között dolgozunk, a Linux üzembiztos működése fontos előnyt jelent számunkra.
A független folyamatokból felépülő rendszer tette lehetővé azt is, hogy a fejlesztést és kipróbálást egyéni programozók modulonként, a saját területükre összepontosítva végezzék. Így alkalmat kaptak arra, hogy biztonságosan használjanak egyedi, sokszor élvonalbeli programozási megoldásokat, amelyek más moduloknál esetleg szükségtelennek vagy oda nem illőnek bizonyultak volna.

Ütemezés és időzítés

Az egyes összetevők időzítési követelményei változók. A mozgásmotort pontosan 60 Hz-es gyakorisággal kell meghívni, hogy a robotok mozgása egyenletes legyen. Ha a motoroknak kiküldött helyzetleíró adatok időtartományban ingadoznak, hirtelen gyorsulások jelennek meg bennük, és a robotlények karjai remegni kezdenek. Mindenképpen szerettük volna a legnagyobb pontosságot elérni a mozgásmotor 60 Hz-es meghívásában, ám ez csak harmadszori nekifutásra jött össze. Először időzített, programozott megszakításokat használtunk, a mozgásmotort a megszakítást kezelő alkalmazásként hívtuk meg. Emellett `POSIX.1b SCHED_FIFO` elsőbbségi sorrendmeghatározást használtunk, hogy miután a futása megkezdődött, a rendszermag ütemezője ne vegye el az erőforrásokat a mozgásmotortól. Így a mozgásmotor felhasználói szinten, és ami még fontosabb, hibakeresőben is remekül futott. A hasonló riasztáskezelők viszont két szempontból hátrányosak:

1. Időzítésük akár több időosztásnyit is ingadozhat, ha a rendszermag ütemezője terhelt.
2. Ütemezésük a rendszermag időzítési egységeitől függ, ezért pontosan nem jósolható meg.

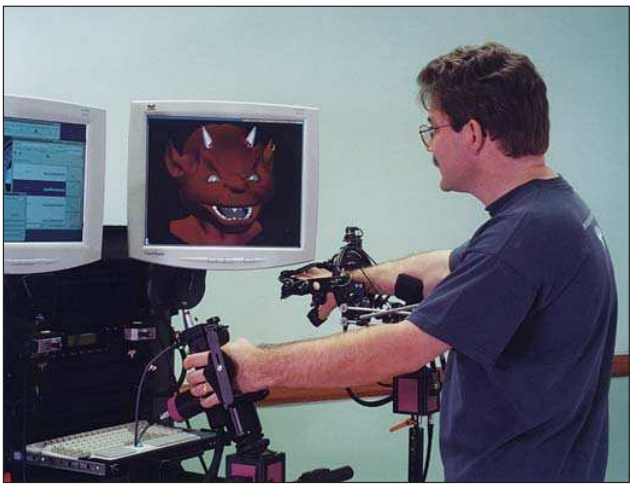
A rendszermagokat tehát újrafordítottuk, hogy növeljük a rendszermag időzítési egységeinek a számát, ezt az utóbbit két okból kifolyólag szerettük volna: a mozgásmotor futtatásának időzítését közelíteni akartuk a 60 Hz-es értékhez, illetve biztosak akartunk lenni abban, hogy a kisebb fontosságú vezérlőfolyamatok újraütemezése gyakrabban történik meg, így jobban követni tudják a mozgásmotor állapotváltozásait.

Második próbálkozásként RTLinux alatt egy szigorúan valós idejű, periodikus futású szálat hoztunk létre 60 Hz-es órajellel. Az RTLinux területen futó szál körülbelül három nagyságrenddel pontosabb időzítéssel futott. Ezt a szálat szigorúan valós idejű időzítőnek kereszteltük el – ő adja az ütemet. Amikor szóhoz jut, egy RTLinux FIFO vezetékbe RTLinux területről egy jelzőt helyez el, amelynek megérkezésekor a felhasználói területen futó, a FIFO által megakasztott mozgásmotor feléled. Ugyan a mozgásmotor meghívása továbbra is a Linux-rendszermag feladata, ezzel a megközelítéssel a vártnál pontosabb eredményt sikerült elérni, ugyanis a be- és kiviteli műveletek nagyobb fontosságúak a rendszermag számára, mint a jelzők kezelése. A FIFO alapú megoldás átlagos késleltetése – ha más, a rendszert leterhelő program nem futott – 40 μsec alatt volt, tehát a mozgásmotor meghívásának gyakorisága a processzor időzítői által lehetővé tett mértékben megközelítette a 60 Hz-es értéket.

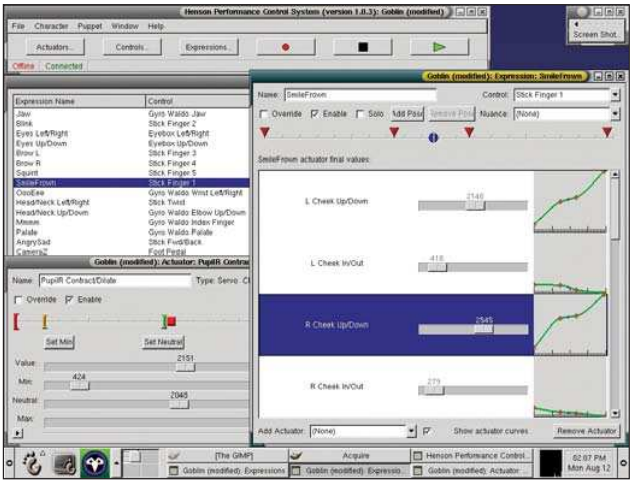
Nagyobb terhelés esetén a rendszermag nem feltétlenül indítja el időben a mozgásmotort, a rendszer tehát nem szigorúan vett, előre tervezhető módon valós idejű, habár éles környezetben, munka közben is remekül működött. Egy kétprocesszoros Athlon alaplappal fenntartható a mozgásmotor meghívásának pontossága, miközben elfut rajta a grafikus felület, az eszközkiszolgáló és a folyamatosan OpenGL-színhelyeket leképező nézegető!

Harmadik és egyben utolsó próbálkozásként egy olyan RTLinux-bővítésnek az elkészítésével bíztuk meg az FSMLabst, amely lehetővé tette a felhasználói területen futó Linux-folyamatok előre kiszámítható ütemezését. A PSC-nek nevezett rendszer révén képesek leszünk arra, hogy egyfajta ugrást hajtsunk végre az RTLinux rendszeres időközönként futó szálából a felhasználói területre, ahol a mozgásmotor fut, majd a műveletsor az RTLinux területre való visszalépéssel fejeződjék be.

A szerződés egy pontja értelmében a forráskód RTLinux-csomagként, nyílt forrással bárki számára elérhető.



Bábkezelő munkában a HDPS digitális bábállomásán



Képernyőkép a HDPS grafikus felületéről

Beviteli eszközök

A vezérlőrendszerben használt beviteli eszközök könnyű és rugalmas használatra tervezett lineáris potenciométerekből állnak össze. Kialakításuknál az elsődleges szempont a bábkezelők által megkívánt mozgások előállításának megkönnyítése volt. Eleinte nagyra törő terveink voltak: az egész rendszert egy linuxos hordozható gépen akartuk futtatni, amit könnyedén a forgatások helyszínére lehetett volna vinni. Csak akkor kezdtünk el 19”-os, állványra szerelt vezérlőrendszereket építeni, amikor CGI-megbízásokat teljesítettünk.

A hordozható gépek kapcsán a legnagyobb gond az volt, hogy 64 analóg csatornát kellett bevezetni egyetlen gépbe. Nem volt megfelelő A/D-átalakító illesztőprogram, így Steve Rosenbluth írt egyet a Computer Boards DAS16s/16 típusához. Még ma sincs olyan PCMCIA A/D-átalakító kártya, amely 16 csatornánál többel bírna, ezért Steve egy külső, analóg multiplexert is tervezett. A multiplexert minden képkockánál (16,6 ms) mind a négy bankon végig kellett kapcsolni – ezt a feladatot az RTLinuxra bíztuk, amit az 20 μ alatti pontossággal, előre kiszámítható módon végzett el.

Ugyan alacsony szintű A/D-átalakítóhoz dicső dolog illesztőprogramot írni, mi jobban is fel tudtuk volna használni kutatási–fejlesztési erőforrásainkat. 2000-ben örömmel értesültünk arról, hogy a United Electronics Industries Linux és RTLinux

Állványra szerelt HDPS összeállítás háromdimenziós, számítógépes karakterekhez – látható a kézi vezérlés, a grafikus felület és a nézegető

illesztőprogramot is kínált PCI-felületre csatlakozó Powerdaq A/D-átalakító kártyáihoz. 64 csatornás PD2-MF-64-333/16L kártyájuk kifogástalanul működött, és igényeinket az illesztőprogram fejlesztésekor az UEI figyelembe is vette.

Robotvezérlő eszközök

Ugyan mindvégig készen kapható termékeket igyekeztünk használni, a különleges igényeinknek megfelelő eszközöket nekünk kellett megterveznünk. A legtöbb gyártó által „kicsinek" gondolt dolgok egyszerűen nem férnek el például egy robothörcsög belsejében.
A digitális motorvezérlő (Digital Motor Controller) egy nagyjából két postai bélyeget kitevő méretű eszköz, a robotizált bábok vezérlésének kulcsfontosságú eleme. PWM- (Pulse With Modulation) jelek tucatjait juttatja el a motorokhoz a robot belsejében. ***Glenn Muravsky*** segítségével Steve egy egylapkás, a Texas Instruments 320LF2406 motorvezérlő DSP-jére épülő gépet tervezett. A lapkában található párhuzamosságnak hála olyan dolgokat is meg lehet vele tenni, amit egy PC-s adatbusszal nem. Steve egy ciklikus PID-algoritmust is készített az egyedi szervomotorok vezérlésére.

Miért pont Linux?

A Linuxot eredetileg, rendszermagszinten nem szigorúan valós idejűre tervezték, ám a valós idejű kiterjesztésekkel elértük azt, hogy a fontosabb feladatok kellő elsőbbséget élvezzenek, miközben továbbra is egy általános célú operációs rendszert használhattunk.
Időosztásos többfolyamatúság, memóriavédelem, a folyamatok közötti kapcsolattartási lehetőség, hálózati és multimédiás API-k – meglétük alapvető feltétel a többi, nem szigorúan valós idejű dologhoz, amit meg akartunk valósítani. Úgy véltük, az RTLinux szerkezete, amelyben maga a Linux a legkisebb fontosságú feladatként fut, számos támogatási alkalmazásunk használatát lehetővé teszi azon a gépen, amelyen a valós idejű feladatok is futnak.
A legnagyobb kihívást nem valamelyik adott rendszerösszetevő kezelése jelentette, hanem a rengeteg eltérő követelmény összehangolása úgy, hogy közben a kirakó egyik elemének igényeit se kelljen csökkenteni. Nem szigorúan valós idejű kódunknak esélyt kellett adni a futásra; mozgásvezérlő kódunk futását kisebb fontosságú feladatok nem szakíthatták meg; a létfontosságú kódrészeket védenünk kellett a kisebb fontosságúaktól; általános rendszermag-szolgáltatásokat kellett alkalmazunk; a segédprogramokat terjesztésünk részeként kellett használnunk. Az operációs rendszer magja által nyújtott lehetőségeknek, illetve a köré szerveződött segédprogramoknak hála, a fejlesztés felgyorsult, hiszen nem kellett mindenből a saját változatunkat elkészítenünk. Ha bármi gondunk volt, tudtuk, hogy a forráskód elérhető, így magunk irányítjuk a sorsunkat. Az összes elvárásunkat figyelembe véve lényegében a Linux volt az egyetlen választási lehetőségünk.
Megjegyezném, hogy vezérlőrendszereink az elmúlt három év során folyamatosan működtek, az újraindítások között eltelt átlagos idő több hónap volt. Gyakran szükség van rá, hogy a gépeket egyik helyről a másikra költöztessük, de forgatás közben egyik vezérlőrendszer sem állt le soha.

Munkáink

Hol látható munkánk eredménye? Először 1999-ben léptünk színre a Webisodes című alkotásban, amelyben a Henson ➲ http://www.muppetworld.com címen elérhető weblapján is látható karakterek szerepelnek. Rengeteg bemutatót tartottunk kiállításokon, köztük a SIGGRAPH 2000 megnyitásán egy interaktív számítógépes békakarakterrel, Brekivel (a Muppet Showból) jelentünk meg.
Az új vezérlőrendszer a Walt Disney Snowdogs című filmében, 2001-ben kezdte meg pályafutását. A film főszereplőjét, a Demon nevű huskyt a nehezebb jelenetekben egy robot helyettesíti – tessék kitalálni, melyek ezek a jelenetek! Mint reméltük is, a rendszer remekül működött. A Stuart Little 2 2001-es forgatásán egy robotvadászsólymot keltettünk életre, Horace D'Fly pedig a Henson hamarosan megjelenő „Kermit's Swamp Years" című filmjének egyik számítógépes karaktere. Jelenleg azt tervezzük, hogy a Warner Brothers' Kutyák és macskák című filmjének folytatásában – továbbá egyéb robotokat igénylő forgatásokon is – használjuk a rendszert. A szórakoztatóipar részéről nagy az érdeklődés: számítógépes grafikai karaktereinket a közeljövőben filmekben, tévéműsorokban és videojátékokban szeretnék használni. Találkozunk tehát a moziban!

Linux Journal 2002. november, 103. szám

David Barrington Holt
A Jim Henson Creature Shop Los Angeles-i igazgatója. Londonban ipari tervezésből szerzett kitüntetéses diplomát, később divattervezéssel, grafikával, fényképezéssel és műszaki modellezéssel is foglalkozott.

Steve Rosenbluth
34 éves, nős és van egy macskája. 13 éve foglalkozik robotokkal és a velük kapcsolatos műszaki megoldásokkal. Korábban bábfilmkészítést, szobrászatot, elektronikát, programozást és kisvállalkozás-fejlesztést tanult.

Michael Babcock (michael@kanji.com)
Montanában nőtt fel, hegyek, fák és szarvasok között. 1992 óta használ Linuxot. Programozási oldalról nézve érdeklik a többnyelvű programok, illetve az elvont programtervezési módszerek. A Jim Henson Creature Shopnál, Los Angelesben négy évig dolgozott egy bábvezérlő rendszeren, különösen annak hálózati részén, felhasználói felületén és 3D-s grafikai megoldásain.