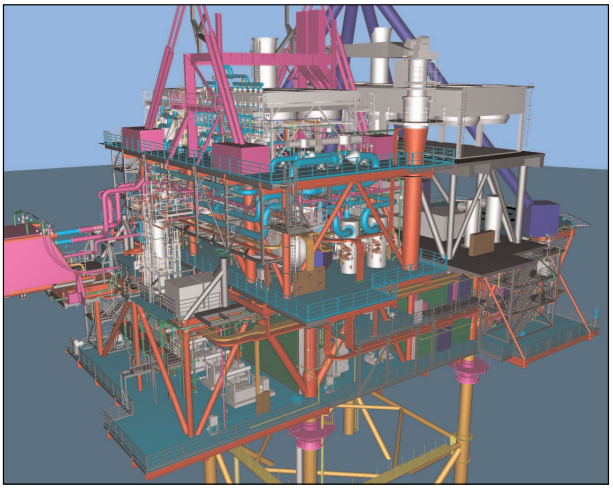
Fejlett térbeli grafika: a GNU Maverik

Programozói környezet csúcsminőségű, interaktív grafikus alkalmazásokhoz.

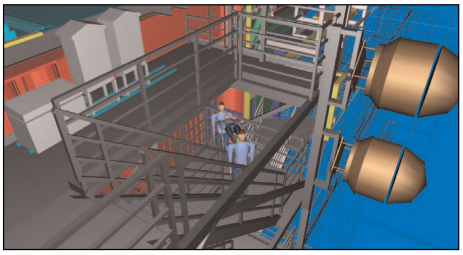
© Kiskapu Kft. Minden jog fenntartva

Egyszerűen modellezhetnek virtuális környezeteket (VK) a programozók az akadémiai kutatók által fejlesztett GNU Maverik segítségével. A fejlesztők elsődleges célja az volt, hogy a jelenlegi rendszerek hiányosságait kiküszöböljék.
A Maverik fő erőssége, hogy több alkalmazás összefogásával egységes keretet ad a fejlesztésnek, de számos beépített szolgáltatással igyekszik megkönnyíteni az alapokról induló felhasználók munkáját is. Lehetőségeit elsősorban a programozók használhatják ki, hiszen nem önálló programról van szó.
A Maverik a GNU része, és a GNU GPL alapján forráskódjával együtt szabadon terjeszthető. A csomagban gyakorlatok, példák és a teljes leírás is helyet kapott. 1999. februári megjelenése óta több ezren töltötték le, s a fejlesztőkhöz számos dicsérő visszajelzés érkezett, üzleti és nem haszonelvű felhasználóktól egyaránt.
Cikkünkben a virtuális környezetet tervezők előtt álló akadályokról írunk, majd arról, hogy a Maverik miként könnyíti meg ezek leküzdését.
A térbeli grafikus tervezésbe nagyon könnyű beleszeretni. Mindenki szívesen körbejárná – ha csak virtuálisan is – jövőbeli házát. A mérnökök szívét is megdobogtatja a gondolat, hogy milyen egyszerű lenne a dolguk, ha a papíron látható épülettervet munkatársaiknak bemutathatnák, esetleg véleményt is kérhetnének róla.

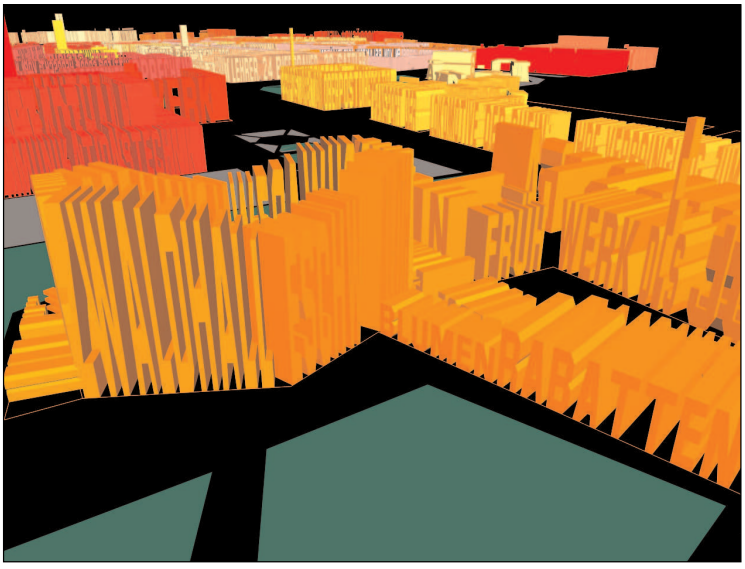


1. kép Tengeri rakodópart

Ezek csodálatos elképzelések, a valóság azonban jóval hétköznapibb. Mi is a gond? Egészen a közelmúltig úgy tűnt, az otthoni számítógépek és VR-eszközök teljesítménye a fő akadály. Az utóbbi néhány évben azonban a grafikus gyorsítókártyák zavarbaejtő sebességgel fejlődnek egyre több PC-tulajdonos gépében találjuk meg valamelyiket. A jelenlegi olcsó gyorsítókártyák teljesítménye a komoly munkaállomások teljesítményével vetekszik, és néha még 3D-s szemüveget is kapunk a kártyához.
Mára világossá vált: álmaink beteljesülését sokkal inkább a kevésbé hatékony programok, mintsem a számítógépek hátráltatják. A térbeli grafikai alkalmazások fejlesztése kemény munka. A lélegzetelállítóan élethű szörnyek, akikre otthon vigyorogva lövöldöz mindenki, programozók és művészek több száz órai munkájának köszönhetik létüket. Egy-egy animációs film modellezése pedig még ennél is nagyságrendekkel több időt vesz igénybe. A Játékháború valósidejű modellezéséhez a jelenlegi legjobb PC-s grafikus kártyáknál tízezerszer nagyobb teljesítményre lenne szükség, főként az összetett grafikai elemek és a megvilágítás miatt. Ahogy azt a számítógépes játékok is bizonyítják, valós időben szintén lehet látványos jeleneteket készíteni. Ne felejtsük el azonban, hogy a legtöbb akciójátékban nem tehetünk meg akármit, például a falra akasztott képet sem vehetjük le, hiszen az igazából a fal része.
És ez még nem minden. Egy valósághű tengeri rakodópart vagy javítómedence leképezése könnyen elfogyaszthat fél gigabájtot is, és ezt az irdatlan mennyiségű adatot egy használható programnak másodpercenként legalább tizenötször fel kell dolgoznia *(1. kép)*.
Vizsgáljuk meg közelebbről, hogy milyen műveleteket kell végeznünk egy ilyen összetett leképezéssel, ezáltal a Maverik előnyeit is értékelni tudjuk. A rakodópart esetében a megjelenítést a szükséges részletekre kell korlátoznunk. Ha egy sok szobából álló épületet belülről kell megjelenítenünk, a leképezés legnagyobb részét figyelmen kívül hagyhatjuk, hiszen egyszerre csak kis terület látszik. Hasonlóképpen járhatunk el egy városi utcakép leképezése során: a takarásban lévő épületekkel nem kell törődnünk. Tehát a felhasználástól függően a leképezést sokféleképpen és gyorsan egyszerűsíthetjük. Ez a módszer azonban nem minden esetben alkalmazható. A rakodópart körül vitorlák, kötélzetek, csövek látszanak, tehát az egyszerűsítést másképpen kell megoldanunk. Ha azonban a művelet sokáig



2. kép Drótvázas szerkezet



3. kép A Maverik segítségével készített Legible City II.

© Kiskapu Kft. Minden jog fenntartva

tart, kifuthatunk a rendelkezésre álló időből (ez általában a másodperc töredéke), tehát gyökeres változtatásra lesz szükség. A legegyszerűbb, ha a távolabbi részeket drótvázas szerkezetként készítjük el. Ez nem mindig tetszetős, de ha így *(2. kép)* gyorsabban mozoghatunk a virtuális térben, akkor még mindig sokkal jobb választás, mint a kitakarás.

Egy másik ígéretes módszer, ha a drótvázas részeket állóképpel helyettesítjük, ezt a nézőponttól függően torzítva szép eredményt kaphatunk. Ez a megoldás jelenleg még kutatási szakaszban van. A virtuális térben különösen fontos, hogy a tárgyak ne „ugráljanak” a különféle megjelenítési módszerek közötti váltásokkor.

Ezen módszerek mindegyike jól használható külön-külön, de együttes alkalmazásuk hatékonysága nagymértékben függ az adott alkalmazástól. A képkockák tökéletes egymáshoz illesztése és a felhasználói beavatkozások figyelése nagy kihívást jelent. Jelenleg a feladat megoldására két módszer kínálkozik.

A program vázát megírhatjuk bármilyen korszerű programnyelven, a grafikai feladatokat pedig egy Mesa- vagy OpenGL-szerű grafikus könyvtárra bízzuk. Így szinte bármit megtehetünk, a számítógép teljesítményét a legteljesebb mértékben kihasználhatjuk. Hátulütője a dolognak, hogy az egész túlságosan alacsony szintű – a munka ilyenkor leginkább arra hasonlít, mintha operációs rendszert kellene írnunk gépi kódban. Emellett, mivel a módszer alkalmazásfüggő, szinte semmit nem vihetünk át az egyik fejlesztésből a másikba.

4. kép A játékprogramokban használatos leképezési módszerek

5. kép Megvilágítások használata

A másik megoldás, ha a virtuális világot valamilyen tervezőcsomaggal építjük fel, ez magas szintű szolgáltatásokat is lehetővé tesz, például egyszerre több felhasználó is barangolhat a térben. Természetesen ez a módszer sem tökéletes, hiszen előfordulhat, hogy a program lehetőségei meg sem közelítik az igényeinket. Egy általunk fejlesztett programnak nyilván saját szerkezete és algoritmusa van, ezeket az adott alkalmazásra kívánjuk kiélezni, tehát a számítógéppel éppen annyit szeretnénk elvégeztetni, amennyi szükséges, se többet, se kevesebbet. Visszatérve a példához: egy CAD-modellben a rakodópartot akár magas szintű kifejezésekkel is leírhatjuk: használhatunk létrákat, csöveket, szelepeket stb. Ha ezt az adathalmazt egy virtuális tértervező programban szeretnénk felhasználni, először a megfelelő formátumra kell alakítanunk azt. A közös nevező általában egy óriási poligonhalmaz, melynek beolvasása elkerülhetetlenül adat- és sebességvesztéssel jár.

A második nehézség, hogy ha valaki a virtuális térben elmozdít egy tárgyat, ezt valahogyan a CAD-program számára is jeleznünk kell, melynek ezután módosítania kell az adatokat. Tehát a világot két változatban is tárolnunk kell: egyszer magában az alkalmazásban, másodszor pedig a térszerkesztőben.

Tervezéskor tehát hamar beleütközünk az alapvető kérdésbe: vagy túl sokat kell programoznunk, vagy a látvány minőségéből kell engednünk amiatt, hogy egyszerre két változatban is tárolnunk kell a teret.

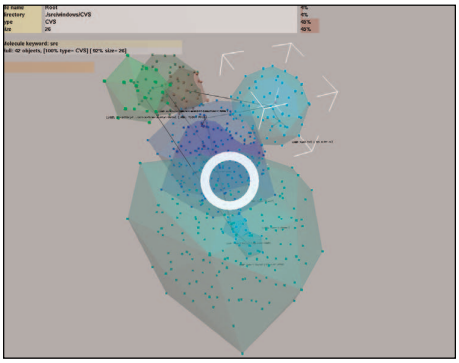
Maverik: a középút

Vajon megtalálhatjuk-e az arany középutat? Készíthető-e olyan keretalkalmazás, mellyel a virtuális terek készítését és kezelését megalkuvások nélkül, hatékonyan végezhetjük?

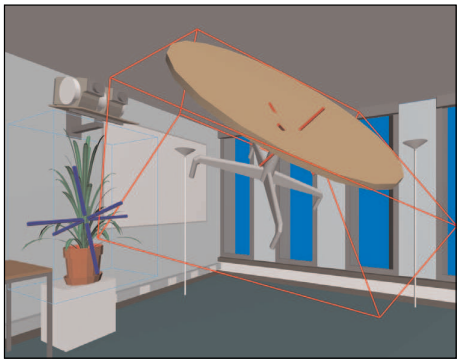
Mi bízunk ebben. Erőfeszítéseinket a Maverik léte igazolja, ugyanis ennek segítségével már néhány hónap múltán komoly, virtuális tereket kezelő alkalmazások készültek világszerte.

A Maverik egy C-alapú segédeszközgyűjtemény, amely egy 3D grafikai könyvtárat (Mesa, OpenGL) használ. Tulajdonképpen nem más, mint a könyvtárra épülő héj, mely megkönnyíti annak használatát. A Maverik egyik fő erőssége, hogy nincs saját belső adatszerkezete, hanem az X11 visszahívásaihoz hasonló módszerrel fér hozzá az alkalmazás adataihoz.

A program közli a Maverikkel, milyen tárgyakat kíván leképezni, majd a Maverik térkezelő szerkezeteket (SMS, spatial management structure) hoz létre a különböző tárgytípusokhoz. Miközben a felhasználó a térben bolyong, a Maverik figyelemmel kíséri az SMS-eket, és a leképezéshez visszahívja az alkalmazást.



6. kép Különleges szerkezetek megjelenítése

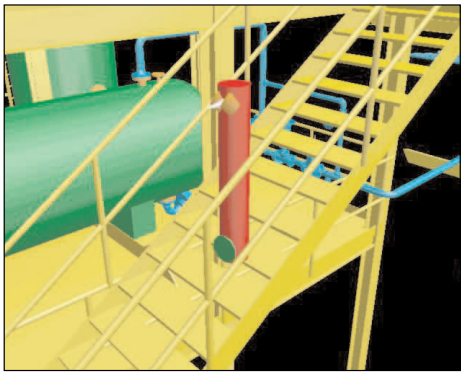


7. kép Hatékony vezérlés

Mit nyerünk ezzel?

Az adatokat az alkalmazás kezeli, tehát egyszerre csak egy adathalmazra kell figyelni. Ami még fontosabb: az alkalmazás minden erőforrását a leképezésre fordíthatja, nem kell a poligonrengeteg átfésülésével foglalkoznia. A térbeli szöveget egy hatékony adatformátumban néhány értékkel meghatározhatjuk – elég a szöveget, a betűtípust és a szöveg helyzetét tárolnunk. Ha azonban egy hagyományos térszerkesztőbe szeretnénk ezt bevinni, mindenképpen

a poligonok közvetítésére lenne szükségünk, így a rendszernek jóval több adatot kellene adott idő alatt feldolgoznia, ez pedig sebességcsökkenéssel, illetve minőségromlással jár.
A Maverik használatával ez a következőképpen néz ki. Az alkalmazásban bejegyezzük a „3D Szoveg" tárgytípust, majd ahogy a felhasználó a térben közlekedik, a Maverik az SMS-ek alapján észleli, hogy tárgy van a látómezőben. Kideríti, hogy szövegről van szó, s az alkalmazás megfelelő eljárásaival valós időben hozza létre a poligonokat. A program ezután elvégezheti a megfelelő minőségmeghatározást – így nagyon egyszerűen megvalósítható az, hogy gombnyomásra változzon a tárgyak részletezettsége. Ezáltal a Maverik tökéletes finomhangoló eszköz lehet a fejlesztők kezében.



8. kép Vezérlési eljárások

A szövegpéldák *Jeffrey Shaw* (ZKM Germany) Legible City II. című alkotásából származnak. Művében a fenti módszer alkalmazásával szavakból képezte le Amszterdam, Manhattan és Karlsruhe városát *(3. kép)*. A *4. képen* ugyanez a leképezés látható a játékprogramokban használatos módszerekkel.
Az előző példához visszatérve, a CAD-program saját adatszerkezete, melyben csövek, szelepek szerepelnek, hasonló módon kerül felhasználásra. Amikor az alkalmazás visszahívást kap egy torony leképezésére, eldöntheti, hogy az adott képkocka esetében mi lenne a legmegfelelőbb – le kell-e képezni a tornyot (mert az egy fontos tárgy a térben), avagy nem. Így az alkalmazás a feltételeknek megfelelően villámgyorsan módosíthatja a megjelenítést. Ugyanígy az összetett megvilágítási módszerekkel rendelkező *(5. kép)*, illetve különleges szerkezeteket megjelenítő *(6. kép)* programok saját adatszerkezetet és módszereket használhatnak a valósidejű teljesítmény megőrzése céljából.
A Maverik segítségével készített műveket a program honlapján is megtalálhatjuk. Fontos megemlíteni, hogy a filmeket egy 450 MHz-es Pentium III PC-n, Voodoo2 kártyával mentették, így jól lemérhető, mire képes a Maverik egy elérhető gépen is.

Ez valami csalás?

A Maverik célja, hogy egymástól meglehetősen különböző alkalmazásokat egységes keretbe foglaljon, így megkönnyítve az azokkal való munkát. Ez előnyös tulajdonság, de úgy tűnik, még mindig a programozóra hárul a legnehezebb feladat. Viszont e keretet nagyon egyszerűen bővíthetjük a megfelelő szolgáltatásokkal.
A csomagban számos kiegészítés is helyet kapott: geometriai alapelem, fájlbeolvasók, a vezérlést segítő szolgáltatások, térbeli matematikai függvények, a 3D szemüveg és az egér kezelésére szolgáló programrészletek stb. *(7. és 8. kép)*. Ezeket bárki tetszés szerint felhasználhatja, akár változtatás nélkül, de saját igényei szerint átalakítva is.
Miután megismerkedtünk a programcsomag szolgáltatásaival, három irányban indulhatunk el:

1. A készen kapott tárgyak és eljárások alkalmazása. Így használva a Maverik egy programozók számára készült térszerkesztővé válhat. Az oktató jellegű példákkal egy teljes rendszert építhetünk fel, ütközésérzékeléssel, valamint testreszabott vezérlési beállításokkal.
2. Saját tárgyosztályok meghatározása. A felhasználó által fejlesztett leképezési és más, ehhez kapcsolódó visszahívások használatával további sebességnövekedés érhető el. A csomag tartalmazza e témakör gyakorlófeladatait is, azonban jogi kérdések miatt a már említett rakodóparti példa nem kerülhetett be.
3. A Maverik rendszermag módosítása és bővítése. A leképezési és vezérlési visszahívások módosítása mellett szinte minden testre szabható: saját kitakarási, SMS és vezérlési módszerekkel is bővíthetjük a Maveriket. Ezenkívül az a legszebb, hogy ezek a fejlesztések nagyon egyszerűen, a már működő elemek veszélyeztetése nélkül beépíthetők a rendszerbe.

A Maverik igazi ellenőrzése, hogy milyen hatékonysággal használhatjuk fel saját bővítéseinket. Célunk nem a térszerkesztés és – kezelés programozásának leegyszerűsítése volt, hanem az, hogy a felhasználónak munka közben ne a rendszer nyűgjeivel kelljen foglalkoznia. A rendszer ezeddig beváltotta a hozzá fűzött reményeket.
A Maverik legutóbbi fejlesztései között egy erőtérvezérlő-rendszer, valamint egy kiegészítő eljárás szerepel, mely a térben barangoló felhasználó előtt megjelenő tárgyakat ellenőrzi – hogy föl lehet-e rájuk mászni stb. Ezeket az elemeket egyelőre próbaváltozatban terjesztik, külön forráskódként.

Mire nem alkalmas a Maverik?

A Maverik nem tartalmaz mozgókép- és hangkezelő eljárásokat, ezeket magunknak kell elkészítenünk.
A Maverik egy egyfelhasználós kis rendszermag (microkernel). Egyszerre több felhasználó jelenlétét, vagy egyszerre több tér használatát jelenleg nem támogatja. A hálózati alkalmazáshoz szükséges feltételeket (a terek összehangolása, frissítése) a felhasználónak kell megteremtenie. Ez természetesen óriási kihívást jelent; mi is dolgozunk rajta, fejlesztéseink eredménye az év második felében várható.
Mindenki véleményére, javaslatára számítunk, hiszen ezek adnak támpontot a további kutatásokhoz, és segítenek minket a programozók igényeinek felmérésében.
A visszajelzéseket a ➲maverik@aig.cs.man.ac.uk címre várjuk.

Kapcsolódó címek

A Maverik honlapján rengeteg animáció található, ezeket egy Linux-alapú PC-n, Voodoo2 grafikus kártyával készítettek.
➲http://www.aig.cs.man.ac.uk/systems/Maverik

A Maverik a Free Software Foundation honlapjáról is letölthető.
➲http://www.fsf.org/gnulist/production/maverik.html.

Adrian West (ajw@cs.man.ac.uk) számítástechnikát oktat a manchesteri egyetemen. A virtuális tértervező rendszerek fejlesztésével foglalkozó Advanced Interfaces Research Group nevű társaság tagja.

Köszönetnyilvánítás
Az AIG köszöni az itt közzétett fejlesztéseket anyagilag támogató nagy-britanniai EPSRC csoport segítségét, illetve a kutatásban részt vevő szervezetek – a CadCentre Ltd., a Sharp Research és a Brown & Root támogatását.

© Kiskapu Kft. Minden jog fenntartva