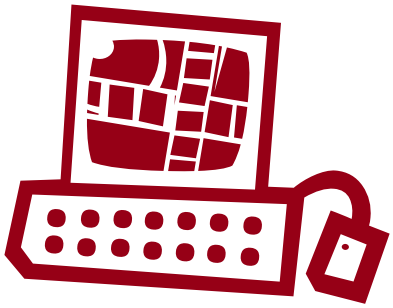
Az OpenAL története

Nyílt forráskód és nyílt szabványok. A Loki egyik ingyenes programfejlesztésének áttekintése.



A Loki Entertainment Software-nél sokféle programmal foglalkozunk, a szabadon terjeszthetőtől kezdve a BSD-szerződéses forráskódon és a GPL, LGPL elvei szerint terjeszthető szabad programokon át a zárt forrású alkatrészmeghajtókig minden megfordul nálunk. Természetesen gépikódokat is írunk. A Linuxra átültetett játékaink nem mindegyike nyílt forrású, de saját, ingyen terjesztett munkánk a Fenris, a Setup, az SMPEG vagy az SDL programokban is megtalálható.
Miután sokszínű tevékenységünk során találkozunk a szabad és a nyílt forrású programok iránt elkötelezett alkotókkal is, így saját véleményünk és irányelveink is módosultak. Egyik fejlesztésünk például abból a felismerésből született, hogy a nyílt, jól leírt szabványokra van a legnagyobb szükség. Ez a fejlesztés az OpenAL volt, melynek története jó iskolapélda arra nézve, hogy milyen fontosak a szabványok – és hogy milyen nehéz megvalósítani őket.

Versenyhelyzet

A Heretic 2 és a Heavy Gear 2 linuxos változata esetében a Loki mérnökei új hanghatásokkal kezdtek el dolgozni, ezek messze felülmúlták a „balról vagy jobbról hallom?” típusú játékok hangjait. A Heretic 2 a hangkártyákat gyártó cégek számára is számos újdonsággal szolgált: az Aureal A3D-jének és a Creative EAX-jának támogatását is megkaptuk. Abban az időben mindkét gyártó túllépett a DirectSound3D lehetőségein, de teljesen különböző irányokban haladva kívánta megvalósítani a térbeli hangzást. A három API közé szorított windowsos játékfejlesztők (és akkor a Windows alatt elérhető számos hangfejlesztő környezetet még nem is vettük figyelembe) inkább nem okoztak maguknak további fejfájást, és kihagyták a térbeli hangzást a programokból (vagy esetleg egy-egy fejlesztőkörnyezet lehetőségeitől vérszemet kapva mégis belevetették magukat a téma kidolgozásába). Itt léptek közbe a gyártók: sokan közülük kiegészítőkkel (EAX, A3D) siettek a kártyák tulajdonságait jobban kihasználni igyekvő programozók segítségére. Ekkor került egyre több játék dobozára a „Térbeli hangzás” (Spatialized 3D sound) felirat, és a windowsos játékfejlesztők egy jó darabig nyugodtan dolgozhattak a meglévő fejlesztőkörnyezetek segítségével.
A Linux-felhasználók helyzete sajnos kedvezőtlenebb volt. A hangkártyák körében szabványosnak tekinthető SB16-hoz illeszkedő meghajtó már megbízhatóan működött, és a PCI-os hangkártyák is meglepően

jól működtek Linux alatt. A legfőbb hiányosság azonban az volt, hogy a kártyák legújabb lehetőségeit sem az OSS, sem pedig az ALSA alkalmazásával nem lehetett kihasználni.
Az Apple-felhasználók számára fejlesztő programozók hasonló helyzetben találták magukat: bár a Mac OS multimédiás API-jai összetettebbek és befejezettebbek voltak linuxos testvéreiknél, ennek ellenére a DirectSound szabványt még annyira sem támogatták, és az EAX, A3D lehetőségei is ismeretlenek voltak arrafelé. A windowsos fejlesztők számára fontos volt a programok hordozhatósága, egy csodálatos, rengeteg lehetőséggel bíró hangkörnyezetet kaptak, melyhez azonban nem tartozott hordozható API. Mivel a két másik résztvevő teljesen más utakat járt be, a dolgok kezdtek egyre rosszabbul alakulni – az időszakot jól jellemzi, hogy akkoriban a Microsoft nem jelentette meg a DirectSound3D NT-s változatát.
A hangkártyák piacvezetőjeként ismert Creative-nak ezzel egy időben kellett szembenéznie az Aureal nevű – a térbeli hangzás PC-s megvalósítását célul kitűző – céggel. Az Aureal a „hullámkövetés” (wavetracing) nevű eljárást helyezte fejlesztései középpontjába. Ennek lényege, hogy a tér tulajdonságait, amelyben a hang megszólal, átadjuk a hangkártya meghajtójának. Fontos megjegyeznünk, hogy a grafikával ellentétben a hangmódszerek a mai napig is főleg programból megvalósított megoldásokra épülnek. A fejlesztőkörnyezetek (Miles Sound System, QMixer) és az egymással versengő alkatrészmeghajtók jelentik a bizonyítékot erre. Az Aureal előnye a szimulációkkal, korai visszaverődések kezelésével és a hatásfok javításával kapcsolatos korábbi tapasztalataiban rejlett. Fejlesztéseit azonban hátráltatta a PC-s piac, ahol általánosan jellemző a „Na, vegyünk két ó’csó hangszórót is a masinához!” típusú hozzáállás. A minőségi hangfeldolgozás átköltöztetése egy olyan rendszerre, ahol hagyományosan sem a fejlesztők, sem pedig a felhasználók nem törődtek a térbeli hangzással, eleve vesztésre álló játszmának tűnt. Mindezek ellenére az Aurealnak sikerült valamit elérnie: felkeltette a felhasználók érdeklődését a jobb térbeli hangzású játékok iránt. Az első érdeklődők számára ez új hirdetési lehetőséget jelentett, és hamarosan minden játékfejlesztő megpróbált lépést tartani velük. A komoly fejlesztési tervekkel bíró Creative visszavágásképpen saját kártyáinak bővített lehetőségeit hangsúlyozta – ezek alapját egy egyszerűbb módszer, a kései visszaverődések valószínűségi zengetéssel megvalósított utánzása képezte.
A korai és kései visszaverődésekre alapuló megoldások kiegészítik egymást, az API-k és azok megvalósításai azonban teljesen különbözőek voltak. A Creative pedig jól választott: az EAX egyszerűbben használhatónak bizonyult, és a valószínűségi zengetés megfelelőbb megoldás volt az amúgy sem a csúcsminőségű hangrendszerekről híres PC-s piac számára.
Természetesen mindkét vállalat odafigyelt a Linuxra is. Az Aureal kidolgozott egy

A2D névre keresztelt rendszert, mely az A3D használatát tette lehetővé más gyártók kártyáin. Ez alapját képezhette volna akár egy nyílt forrású környezetnek is. Az Aureal kódbázisa hatékony volt, gépi kódban íródott, és ez a Linuxra történő átvitelt kissé megnehezítette volna. A Creative, linuxos programozók egy csapatával, meghajtók fejlesztésébe fogott, és az EAX linuxos megvalósításában látta a jövőt. Mindkét API a Microsoft DirectX és COM rendszereinél érvényes ajánlások figyelembevételével íródott, de az Aureal téralapú A3D API-jának tervezésénél az OpenGL-ből is vettek át ötleteket. Itt léptek a képbe a játékfejlesztők.

A nehéz kezdetek

Az EAX és az A3D előtt néhány játékfejlesztő egy hordozhatóbb hangrendszer megvalósításán törte a fejét. Az OpenGL-GameDev levelezőlista, melyen a hordozható kódok írásával foglalkozó fejlesztők fő fóruma 1998-ban egy új listát indított egy új, nyílt forrású hangkönyvtár megszületéséért. A lista neve – az OpenGL mintájára – OpenAL lett. Egyre több javaslat érkezett a listára, de hamar kiderült, hogy szinte mindenkinek másra lenne szüksége. Néhány listatagot csak a zene és a zeneszerzés érdekelt, ők főleg programból megvalósított hangképzéssel foglalkoztak azelőtt. Mások csak a térbeli hangzással foglalkoztak volna, mint a DirectSound3D idejében. Néhányan különleges, kifinomult OOP-megoldásokat szerettek volna az API-ban látni, míg mások beérték volna valami egyszerűbb, általánosabban használhatóval is. A kezdeti lelkesedés ellenére e véleménykülönbségek miatt csupán néhány fejlécfájl és egy, az elveket tartalmazó leírás készült el, és a fejlesztés félbeszakadt.

Talán nagyobb egyetértésre lett volna szükség, de a bukás fő oka az volt, hogy senki nem tudta igazán, hogy mi ennek az egésznek a célja. Ráadásul a hangkártyák tucatnyi új lehetőségét is kezelni kellett volna. Ez utóbbi végül is kétszer buktatta meg az OpenAL-t. 1999-ben a lista feltámadása látszott körvonalazódni: a Game Developer Magazine májusi számában *Jonathan Blow* cikke melletti oldalhasábon *Terry Sikes* (az OpenAL leírás szerzője) és jómagam az OpenAL céjait ismertettük. Akkoriban az Aurealnál egy javított, hordozható A3D kifejlesztésén gondolkodtak, mi pedig leírtuk, hogy az OpenAL képes lesz az OpenGL-lel való közvetlen együttműködésre, főleg a térgeometriai adatok feldolgozását illetően. Sem az Aurealnál, sem pedig az OpenAL levelezőlistán nem történt semmi változás a cikk hatására, és az állóvizet csak a Loki 1999 végén történő beszállása kavarta föl.

A Heretic 2 és a Heavy Gear 2 fejlesztése akkoriban kezdődött, és mivel nem voltunk (sőt, igazság szerint most sem vagyunk) abban a helyzetben, hogy az EAX-ot vagy az A3D-t támogassuk, nyilvánvalóvá vált,

hogy valamiféle megoldásra van szükség. Mindenképpen szerettük volna megvalósítani a DirectSound3D lehetőségeit (távolságérzékelés, Doppler-hatás, térbeli hangzás, hangmagasság- és iránykúpok) Linux alatt is. Ezenkívül természetesen célszerű volt a Creative-nál ás az Aurealnál érdeklődni a linuxos meghajtók felől. A Loki az eredeti OpenAL fórumon közelítette meg a fejlesztőket, felállított egy levelezőlistát, majd *Joseph I. Valenzuelát* megbízta az első elemek kifejlesztésével, majd mindhárman

munkához láttunk. *Michael Vance* (a Heavy Gear 2 fő programozója) és jómagam (a Heretic 2-n dolgoztam) elemeztük az eredeti OpenAL vitafórumot és a létező javaslatokat. Számos döntés meghozatalakor figyelembe vettük a régi hibákat, például legyenek-e OpenGL-szerű előírások, milyen legyen az API felépítése. Természetesen

a hang világa sok tekintetben különbözik a grafikától, de a felesleges egyezkedések elkerülése céljából sokszor a GL-ben alkalmazott eljárásokat vettük át. Az elején úgy terveztük, hogy az API nagy része az A3D-hez hasonló módon kezeli a térkörnyezetet, sőt, így okoskodtunk az OpenAL többi részével kapcsolatban is. A GL „utánzása” során eddig jutottunk: az OpenAL tulajdonképpen egy jelenetleíró API, mely számos, pontosan körülírt objektummal dolgozik. A GL-től más területeken viszont eltértünk: az OpenAL kevesebb belépési ponttal és több tokennel rendelkezik, ez azzal az előnnyel járt, hogy az API eljárásait úgy módosíthatjuk, hogy megőrizzük az ABI-val való megfelelőséget. A GL alrendszereihez (GL, GLX/WGL, GLU, GLUT) hasonlóan bevezettük az AL, ALC, ALU és ALUT alrendszereket, de szinte kizárólag az AL API-jával foglalkoztunk. Olyan régen vártunk már az OpenAL-ra, hogy nem bírtunk magunkkal és a Heavy Gear 2-be már be is építettük. Az OpenAL fejlesztésvezetőjének már ebben az első szakaszban komoly nehézségei támadtak az állandóan változó szabvány és maga a megvalósított könyvtár összehangolása terén. A Creative még jóval a San Joséban megrendezett 2000. évi Game Developer’s Conference (GDC) előtt értesített bennünket arról, hogy érdeklődik az OpenAL iránt. A saját linuxos meghajtójukkal kapcsolatos munkák kiegészítéseként, és tiszteletben tartva a hordozhatóságot és a széles körű elfogadottságot, a kártyagyártók számára egyre fontosabbá vált egy gyártóktól és operációs rendszerektől független hang API. A Microsoftról már régebben kiderült, hogy nem sok kedve van a DirectSound3D-t tovább fejleszteni, az Interactive 3-D Level 1 és 2 iránymutatásait közzétevő Interactive Audio Special Interest Group (IASIG) pedig nem kívánt az API-k szabványosításában részt venni. A Loki mindent elkövetett azért, hogy a szabvány és a megvalósítás ne csak a rövid távú igényeknek feleljen meg, mi pedig bíztunk a fejlesztés nyílt természetében (mi másért hívnák OpenAL-nak), de a Creative és más gyártók érdeklődése olyan váratlan meglepetés volt, mely megváltoztatta a szabályokat. Az OpenAL tehát segített bennünket abban, hogy a hangkártyák képességeit teljes mértékben kihasználó linuxos játékokat dobjunk piacra, majd a gyártók érdeklődése nyilvánvalóvá tette, hogy itt a lehetőség egy új szabvány megszületésére.

Beindult a dolog

A Loki és a Creative a GDC-n közölte szándéknyilatkozatát, mi pedig közzétettük *Michael Vince* szabványleírását. Azóta a követelmények némiképp megváltoztak, és az utolsó fél évben rengeteg kiegészítő munkára volt szükség. Bár eredetileg mi a térbeli hangzással kívántunk foglalkozni, de rájöttünk, hogy ugyanilyen fontos a sztereó és a többcsatornás formátumok támogatása, melyek közül egynéhány (például az MP3) nem igazán nyílt fejlesztés.
A tömörítés és a hangfolyamok nehéz témakörnek bizonyultak. E területeken nem is egy megoldás született már, és *Ian Ollman*, az OpenAL levelezőlista egyik tagjának javaslatára beépítettük az átmeneti tárolók sorkezelését (buffer queueing), ezzel lehetővé téve a hangfolyamok megvalósítását. Minél teljesebbé vált a kép, annál jobban izgultunk a végtermék jellege miatt.
A visszirányú csereszabatosság elvét nem egyszer, nem kétszer, hanem többször is megsértettük, régebbi játékainkhoz pedig bővítések kiadásával igyekeztük megoldani a régebbi, a szabvány új megvalósításából kimaradt lehetőségek támogatását. A Heavy Gear 2-t a Soldier of Fortune, majd a Sim City 3000 Unlimited követte. Manapság az Unreal Tournament is az OpenAL-t használja, és az UT-szerződésre épülő játékok is örökölni fogják e tulajdonságot. A Cognitoy csapat Mindroverje is OpenAL-t használ Linux alatt, és ők (sok más windowsos fejlesztőhöz hasonlóan) a Win32 OpenAL alkatrész-támogatással rendelkező meghajtóinak megjelenésére várnak. Mára szinte az összes OpenAL-t használó játék Linuxra íródott, de remélhetőleg nemsokára más felületek is bekapcsolódnak.
A Lokinak, illetve *Jean-Marc Jot*, *Garin Hiebert*, *Carlo Vogelsang* és mások révén a Creative-nak köszönhetően az OpenAL szabvány már az 1.0-s teljes változatnál tart, mely magában foglalja a DirectSound3D tulajdonságait, másrészt viszont eltér azoktól. Például az átmeneti tárolók (buffers) szigorúan elkülönülnek a forráskódtól: a kódok megosztva érik el azokat. Néhány változás inkább árnyalatnyi, például a viszonyítási távolságok közti különbségtétel, vagy a Doppler-számításokban használt viszonyítási sebesség pontos megfogalmazása. Néhány esetben, például a többcsatornás adatok kezelése során viszont szinte alig volt szükség változtatásra.
Ebben a szakaszban nem maga a szabványleírás számít igazi eredménynek, hanem amit a fejlesztés során tanultunk. Még mindig messze vagyunk attól, hogy befejezettnek tekintsük művünket, de a munka folyamata jól halad, létrejött a megfelelő párbeszéd és állandóan javítgatunk. Néhány dolgot az IETF-ből is kölcsönöztünk, no és az OpenGL bővítésének és módosításának működését is alapul vettük. A jelenleg általunk használt fő eszközcsoport – a DocBook DTD SGML változata és a megfelelő linuxos olvasó és formázó eszközök – elég jól bővíthető, és bár maga a leírás elég modulárissá vált, a kidolgozás és a leképezés még sok kívánnivalót hagy maga után. Számos további lehetőség beépítését tervezzük, de ezek megvalósítása még várat magára. Azon döntésünk, hogy az SGI OpenGL-t követjük, segített megőrizni az eredeti lendületet, mely a korábbi OpenAL próbálkozásokból mindig kiveszett az idő során. Az OpenAL név választása is egyértelmű: elkötelezett hívei vagyunk egy gyártófüggetlen, nyílt forrású rendszer kifejlesztésének – és azt hiszem, hogy eddig be is tartottuk az ígéretünket.
A következő mérföldkő az 1.0-s szabvány hivatalos lezárása lesz, melyet az OpenAL

windowsos, linuxos stb. változatainak megjelenése követ. Az OpenAL mindig hatékony meghajtótámogatást fog jelenteni, ehhez természetesen a Creative és más gyártók munkájára is szükség lesz. A Loki azt szeretné, ha a gyártók a nyílt forrású megoldásokat részesítenék előnyben, de a végső döntést maguk a cégek hozzák meg. Ez minden nyílt szabvány természetes velejárója – ahogy minden ingyenes programokat fejlesztő csapatnak joga van az OpenAL API kidolgozásához, a vállalatok ugyanígy saját változatok használata mellett is dönthetnek. A hatékony megvalósításokat és a fejlett eljárásokat mindenki fontosnak tartja, mégis eltart egy darabig, míg a nyílt forráskód elfogadottá válik.

A jelenlegi szabványleíráson túl az 1.1-es OpenAL-változathoz már egy sokkal letisztultabb tervünk van. Főként a Creative javaslatain alapuló IASIG I3DL2 előírások határozzák meg a használt lehetőségeket, ezeket gyártókra jellemző bővítésekként tervezzük megvalósítani, az 1.1-es változatban már gyártófüggetlenek lehetünk. Sok kérelmet, javaslatot át kell néznünk, és az OpenAL 1.0 esetében figyelmen kívül hagyott szolgáltatásokkal is foglalkoznunk kell. Ahhoz képest, hogy jelenleg a felhasználók száma meglehetősen csekély, meglepően sok váratlan javaslatot kaptunk. A környezetkezelő API-nak (ALC) több felületen is bizonyítania kell ahhoz, hogy (az OpenGL-lel ellentétben) egy teljes mértékben hordozható megoldást adjunk a fejlesztők és a felhasználók kezébe. Az API, melynél jelenleg csak a tokenek nagy számával oldható meg a kevés belépési ponttal való működés, talán jobban egyensúlyba kerül, ha a rendszer megbízhatónak bizonyul.

Mindeközben folytatjuk a játékok áthozását Linuxra. Bár mindegyik sajátos módon kezeli az erőforrásokat, úgy gondoljuk, hogy a jelenlegi API megfelelő lesz. A legújabb lehetőségeken (például a mikrofonos vezérlésen) kívül a munka fő részét mostanában a hangfolyamok kezelése és a további kodekek kifejlesztése jelenti. Talán kissé csúfondárosan hat, de a jelenlegi fejlesztési tervben nem szerepel a visszaverődések és más fejlett lehetőségek támogatása. Bár talán lenne értelme a visszaverődések kezelésének, de a játékok motorja általában sokkal pontosabban képes ezekre figyelni, és az OpenAL fejlesztésében egyre inkább azt az irányt követjük, hogy a programozó saját maga dönthesse el, hogy a jelenet miként „válaszol” az ehhez hasonló körülményekre. A fő szempont ezután az lett, hogy az OpenAL-t lehessen használni szerkesztésre, tehát nem a gyártójellemző lehetőségeken törtük a fejünket, és a cégek saját tulajdonú megoldásaiból sem akartunk mindenáron bizonyos elemeket átvenni. A Sensaura cég munkájának köszönhetően az AL-ból eredetileg kihagyott HRTF-ek is megjelentek, lehetséges bővítésként. Az Aureal fejlesztési tervében szereplő Dolby-támogatás is minden bizonnyal bekerül az OpenAL-ba.

A szórt műsorfolyam Khronos-féle, az SGI által támogatott megvalósítását az OpenAL-hoz kell még igazítanunk – az OpenGL-hez hasonlóan a jövőben az OpenAL-t is az együttműködés megkönnyítésére bővítésekkel látjuk el.

A tervezési és a programozási munka mellett az OpenAL foglalkozó fejlesztőcsoportot is át kell szerveznünk. Ami ma önkéntesek laza hálózata néhány vállalat támogatásával, annak holnap egységes, nem bevételközpontú szervezetként kell működnie. Ismét az OpenGL példájából kiindulva: létrejön egy elemző testület, egy szavazórendszer, és minden olyan elem, mely egy nyílt szabvány kialakításánál fontos lehet. Az OpenGL-lel ellentétben a példamegvalósítás és a próbakörnyezet nyílt forrású lesz. Az OpenGL-hez hasonlóan a bejegyzett védjegyet csak azok használhatják, akik átjutnak a hivatalos ellenőrzésen.

Mindennek a fontosságát a *Brian Paul* neve által fémjelzett Mesa is jelzi: ez a semmilyen más ingyenes fejlesztéshez nem hasonlítható letisztult projekt öt év alatt elvezetett a mai tökéletes 3D grafikával rendelkező Linuxhoz. A Mesa biztosította a keretet, a 3dfx linuxos Glide-ja pedig a meghajtót, melyek együttesen a linuxos játékokban is elterjedté tették a gépi leképezést. Az SGI-nek az OpenGL-en végzett gondos munkája és a Brian Paul számára nyújtott támogatásuk nélkül ma nem létezne a DRI nyílt forrású megoldás Linuxra, de valószínűleg az nVidia meghajtója sem. Ha kívánhatunk valamit, akkor az az, hogy az OpenAL ugyanazt a biztonságot jelentse a Linuxnak és más felületeknek, mint amit az OpenGL. A Loki számára most a legfontosabb, hogy az alkatrészek egyaránt támogassák a Linuxot és a Windowst. Végül az is nagy álmunk, hogy minden játékfejlesztő az új, nyílt forrású szabványokat részesítse előnyben a saját megoldások helyett.

Összegzés

Megérte vajon? Többet értünk el, mint amennyit eredetileg terveztünk, de ha az általunk elvégzett munka elősegítette egy újabb nyílt forrású API megszületését, akkor a válaszunk határozott igen. A Linuxnak csupán a közelmúltban kellett szembenéznie a szabványosítás nehézségeivel, és elmondhatom, hogy az IETF-fel szemben táplált tiszteletem megsokszorozódott az évek során. A Linuxnak szüksége lesz egy multimédiás API bővítésre és egy csak ezzel foglalkozó csoportosulásra azért, hogy a gépgyártók biztos alapokra építhessenek a jövőben – és akkor még nem is említettük azt a rengeteg ingyenes programot, melynek fejlesztői azért küzdenek, hogy műveik hordozhatók legyenek. A Linux-hívők valószínűleg alábecsülik a DirectX-nek a windowsos játékokra és magára a Windowsra gyakorolt hatását. Az is lehetséges, hogy sok linuxos fejlesztő alábecsüli az otthoni felhasználók igényeit. Azonban a Linux csak egy azon felületek közül, melyeknek egy átfogó, hordozható multimédiás API-ra lenne szüksége. Ha az OpenAL is hozzájárul ezen operációs rendszerektől független „OpenX” megvalósításához, akkor elmondhatjuk, hogy sokkal többet értünk el, mint amiért ezt az egészet elkezdtük.

Bernd Kreimeier játékfejlesztő, író és fizikus. A Loki programozójaként részt vett a Heretic 2, a Quake 3 és más híresség linuxos változatainak elkészítésében. Többek között ő felügyeli az OpenAL szabványleírását.

Kapcsolódó címek

Az OpenAL szabványleírása és a példamegvalósítás
➲ http://www.openal.org/

Az IASIG honlapja (I3DL1, I3DL2 ajánlások)
➲ http://www.iasig.org/

A Creative Labs fejlesztői oldala (EAX SDK)
➲ http://developer.creative.com/

Miles Sound System
➲ http://www.smacker.com/miles.htm

QSound Labs
➲ http://www.qsound.com/

Khronos Initiative
➲ http://www.khronos.org/

További honlapok a témában

A Loki Software nyílt forrású fejlesztései
➲ http://www.lokigames.com/development/

OpenGL ARB
➲ http://www.opengl.org/developers/faqs/arb_faq.html

Game Developer Magazine
➲ http://www.gdmag.com/