FIATAL MŰSZAKIAK TUDOMÁNYOS ÜLÉSSZAKA

Kolozsvár, 2004. március 26-27.

3D MODELLEZÉS LEHETŐSÉGEI

László Sándor Elek, Szilágyi Sándor Miklós

Abstract

This paper presents designing methods used in a 3D modeler program. The three dimensional representation has a lot of well-known and widely used methods. This is the reason why the program uses beside the own methods the earlier developed and good working algorithms. The main step of the developed method is to generate three dimensional objects from minimai starting information. All 3D objects are considered a combination of base objects (like cube, cylinder etc). These objects are generated by equation of curved surfaces with different parameters. Doe to optimization for game programs the evaluation of the results was made by appearance. During evaluation we recognized that the sweetly usage of "Level of Detail" technique for generated 3d objects is possible in different environments. We adjusted that the developed methods is not efficient in all cases for automated texture coordinate calculations.

Összefoglaló

A dolgozat egy 3D modellező programot és az általa alkalmazott formatervező módszereket mutatja be. Tekintettel arra, hogy a 3D reprezentáció gazdag módszertannal rendelkezik, a program ötvözi a már jól bevált algoritmusokat a saját fejlesztésű módszerekkel. A módszer legfontosabb lépése a 3D objektumok képezése kevés kiinduló információ alapján. Minden háromdimenziós objektumot egyszerű alapobjektumok (hasáb, henger, stb.) kombinációjaként kezelünk. Az eljárás során az alapobjektumokat egy görbe felület egyenletének más-más paraméterezéséből kapjuk. Mivel ez a technika játékprogramokra lett optimalizálva, elégségesnek bizonyult a 3D objektumok végső aspektus szerinti kiértékelése. A módszer mérlegelése során kiderült, hogy a „level of detail” technikát használó programokban könnyen felhasználhatóak a generált 3D-s objektumok. Megállapítottuk, hogy a textúra koordináták automatikus kiszámítása nem minden esetben adott kielégítő eredményt.

I. Bevezető

A számítógépes grafika egyik fontos alkalmazási területe a komputeres játékok. Minden játékban a látvány alapját az előre megtervezett virtuális környezet alkotja, amelynek építőelemei az objektumok (egyszerű geometriai elemek kombinációi).

Mivel a lebegőpontos számítások rendkívül erőforrás-igényesek a grafika fejlődése az objektumok finomabb geometriai ábrázolása helyett az élethű textúrázás fejlesztésének irányába terelődött [1]. Úgy tűnik, hogy a formatervezés fejlődése ezen a téren eljutott egy lassú fejlődési szakaszba, amikor nem sok új modellezési technika jelenik meg.

Célunk egy olyan program készítése, amely a játékokban is felhasználható 3D alakzatokat generál. Legfontosabb feladat az eddig használt módszerek módosítása oly módon, hogy könnyebben programozhatóak legyenek, és lehetővé váljon az objektumok automatikus generálása [2].

II. Módszer ismertetése

Az objektumok létrehozásánál felhasznált matematikai módszerek legtöbbje a vektorokkal való műveletekre korlátozódik. Ahhoz, hogy változatos felületeket hozhassunk létre minél kevesebb bemeneti adat alapján, felhasználtuk a három kontrol-ponttal rendelkező Bezier görbe egyenletét:

$$B(t) = P_1 * (1-t)^2 + P_2 * 2*t*(1-t) + P_3 * t^2, t \in [0,1] \quad (1)$$

További felhasznált eszközök:

Képfeldolgozás: A *jpeg* formátumban megadott textúrákat egy erre a célra kifejlesztett program dekódolja. További információ a <http://www.iijg.org/> internetes címen található.

Multitexturing: a *Silicon Graphics* által készített *Glexth.h* állomány segítségével a grafikus kártyák összetett-textúrák funkciója is kényelmesen használható. További, a rendszer által támogatott eljárások információi megtalálhatóak a <http://www.opengl.org/> címen.

Egy bonyolult térbeli alakzat megjelenítéséhez szükséges a szerkezet lebontása egyszerűbb objektumokra, mint pl. kocka, gömb stb.. A háromdimenziós alapobjektumokat (gömb, kúp, kocka, hasáb, stb.) egy-egy különálló programmodul generálja. A modulok közös tulajdonsága, hogy az ún. "kontrol pontok" -ból kiindulva generálja az objektumokat leíró vertex (=térbeli pont) listát [1] és a textúra koordinátákat. A modulok közös vonásainak észrevétele leegyszerűsíti a programozási folyamatot. Elég csak egy általános objektumgeneráló modult készíteni, amely néhány paramétertől függően más és más objektumokat generál. így lehetővé válik a görbe felületek létrehozása is ugyanazzal a modullal, amely pl. gömböt, vagy kockát generál. Nagy előnye ennek az eljárásnak az, hogy gyorsan változtatható a generált objektumok kidolgozottsága. Ez a módszer a grafika gyorsításánál lényeges. Ha egy tárgy távol van, akkor nem szükséges minden apró részletét kirajzolni, ezáltal csökkentve a tárgyak formai részletességét, csökken a kirajzolt háromszögek száma. Ez egy

olyan „level of detail” megvalósítást eredményez, amely az objektumok legrelevánsabb karakterisztikáit hagyja meg.

Az alapobjektumokat egy-egy különálló modul generálja. A modulok közös tulajdonsága, hogy az ún. "kontrol pontok" -ból kiindulva hozzák létre az objektumokat leíró vertex listát, textúra koordinátákat és a háromszögeket. A Bezier görbe segítségével készíthető el az alábbi felület (1. ábra), amelyben minden egyes csomópontot két görbe metszéspontja ad meg. Négy rögzített pontból kiindulva kiszámíthatóak a közbeeső pontok amennyiben rendelkezésre áll még öt további görbületszabályozó pont. Minél sűrűbb felosztás szerint változik a Bezier görbe egyenletében (1)

1. ábra: Bezier görbe alkalmazása felületgenerálásnál.

2. ábra: Textúrázás

szereplő "t" változó értéke annál finomabb a generált felület. Sajnos a nagy felbontás megnöveli a felületet alkotó háromszögek számát.

Az objektumok textúrázása a textúra koordináták kiszámításával történik, az adott objektum méretének figyelembevételével [3]. A 3D-s objektum minden pontjában szükséges az (u,v) textúra koordináta kiszámítása, amely meghatározza, hogy a textúra melyik része illeszkedik az adott ponthoz. Ez a bonyolult folyamat egyszerűsíthető, ha figyelembe vesszük, hogy a Bezier görbével leírt görbe felületek felbonthatóak elemi téglalapokra. Egyszerű objektumok esetében a program önállóan számítja ki a helyes textúrázáshoz (2. ábra) szükséges arányossági együtthatókat. A használt eljárás viszont görbe felületek esetében nem használható.

Talán az egyik legfontosabb látványfokozó elem, a megvilágítás utánzására szolgáló *lightmap*. Ez valójában egy textúra [4]. A fényforrások paraméterei (koordináta, fényesség, hatótávolság, szín) meghatározzák a generált *lightmap* tulajdonságait [3]. Minden háromszögnek van egy hozzárendelt *lightmap* textúrája. Látni lehet, hogy a sok háromszögből álló modellek megvilágítása sok számítást igényel. Tehát a *lightmap* csak statikus megvilágítást valósíthat meg.

III. Eredmények

Az elért eredményként említhető a megvalósított program és a vele készített teszt "környezet" (3. ábra), amely a felhasznált eljárások hatásosságát bizonyítja. Az általunk használt objektumgeneráló módszer lehetővé teszi a generált 3D környezetek optimalizált megjelenítését, mivel a tárgyak kidolgozottsága változtatható, annak megfelelően, hogy a megjelenített tárgy mennyire közel helyezkedik el. Ugyancsak fontos eredmény, hogy a görbe felületek létrehozásánál a felhasználó

könnyen változtathatja a felület alakját befolyásoló paramétereket. Ezáltal a modellezési folyamatok könnyebben kezelhetőek.

IV. Elemzés és következtetés

Az általunk kidolgozott módszer hatékonyan egyszerűsíti a háromdimenziós objektumok képezésének folyamatát. A legenerált objektumokat meghatározó információ mennyisége nagyon lecsökkent (megközelítőleg ezredrészére). Bizonyos esetekben hátrány lehet, hogy az adatokból minden későbbi felhasználás előtt egyszer ki kell számítani az objektumot leíró háromszöglistát, mivel a program csak a kiinduló adatokat tárolja. Hatalmas előny, hogy a későbbi felhasználásra elmentett információ mennyisége nem függ az objektumok kidolgozottságától.

3. ábra: A leírt megoldásokra alapuló program felhasználói

V. Könyvészet

- [1] Mark J. Kilgard "All About OpenGL Extensions" including specifications for some significant OpenGL extensions, NVIDIA Corporation
- [2] László Sándor Elek '3D World Editor', II. Műszaki Tudományos Diákköri Konferencia, Marosvásárhely 2003.
- [3] Jeff Molofee „OpenGL Tuotutorials”, www.nehe.gamedev.net
- [4] Jackie Neider, Tom Davis, Mason Woo „The Red Book" OpenGL Programming Guide

László Sándor Elek - Hallgató

EMTE Sapientia - Marosvásárhely, Piata Trandafirilor 61

Tel.: 0265-257412 E-mail: sankoboy@yahoo.com

Szilágyi Sándor Miklós - Egyetemi Adjunktus

EMTE Sapientia - Marosvásárhely, Piata Trandafirilor 61

Tel.: 0265-264490 E-mail: szsander@hotmail.com