OpenGL

Roșca Alexandru

***Ce este OpenGL***

OpenGL este prescurtarea de la Open Graphic Librarie. Reprezintă o specificație standard care definește o aplicație cross-platform API (application programming interface) foarte utilizat pentru programarea componentelor grafice 2D și 3D ale programelor de calculator. Interfața constă în peste 250 de apeluri diferite care pot fi folosite pentru a desena scene 3D complexe din primitive simple.

OpenGL a fost dezvoltat de Silicon Graphics Inc. (SGI) în 1992 și este foarte utilizat în softuri CAD, realitate virtuală, vizualizare științifică, simulări de zboruri sau jocuri pe calculator. Acest ultim domeniu este in stransa competiție cu tehnologia DirectX de la Microsoft.

***Tehnologia 3D***

Obiectele tridimensionale simple pot fi reprezentate cu ecuații care funcționează pe un sistem de referință cartezian tridimensional : de exemplu, ecuația x² + y² + z² = r² este perfectă pentru o sferă de rază r . Ecuații nu sunt suficiente pentru a descrie cu exactitate formele complexe care alcătuiesc cea mai mare parte a lumii reale. O tehnică eficientă este modelarea poligonală sau modelarea poligonală. Acest lucru permite un nivel mare de detaliu, dar în detrimentul cantității mai mari de informații necesare pentru a stoca obiectul rezultat, numit model poligonal. Un model poligonal poate fi rafinat cu algoritmi pentru a reprezenta suprafețe curbate: această tehnică se numește „ suprafețe de subdiviziune”. Modelul este rafinat cu un proces de interpolare iterativ, făcându-l din ce în ce mai dens cu poligoane, care vor aproxima mai bine curbele ideale, derivate matematic din diferiții vârfuri ale modelului.

Grafica 3D este astfel un sistem de producere a culorilor pentru pixeli care te convinge că scena pe care o privești este mai degrabă o lume 3D decât o imagine 2D. Procesul de conversie a unei lumi 3D într-o imagine 2D a acelei lumi se numește redare. Există mai multe metode de redare a unei lumi 3D. Procesul utilizat de hardware-ul grafic în timp real, implică o cantitate foarte mare de falsuri. Acest proces se numește rasterizare, iar un sistem de randare care utilizează rasterizare se numește rasterizare. În rasterizatoare, toate obiectele pe care le vedeți sunt cochilii goale. Există tehnici care sunt folosite pentru a vă permite să deschideți aceste cochilii goale, dar asta pur și simplu înlocuiește o parte a carcasei cu o altă carcasă care arată cum arată interiorul. Toate aceste cochilii sunt făcute din triunghiuri. Chiar și suprafețele care par a fi rotunde sunt doar triunghiuri dacă te uiți suficient de atent. Există tehnici care generează mai multe triunghiuri pentru obiectele care apar mai apropiate sau mai mari, astfel încât privitorul să nu poată vedea aproape niciodată silueta fațetată a obiectului. Dar ele sunt întotdeauna făcute din triunghiuri.

**Crearea scenei**. O scenă poate fi compusă pornind de la „primitive”, adică modele tridimensionale reprezentând primitive geometrice, care pot fi împărțite în fețe simple sau pot fi combinate pentru a forma obiecte mai complexe. Cea mai simplă modalitate de organizare este crearea unei matrice de primitive, dar această metodă nu permite o descriere mai detaliată a scenei, pur și simplu „explică” redării cum să o deseneze. O tehnică mai avansată organizează obiectele într-o structură de date de copac, care permite gruparea obiectelor în mod logic. Primitivele sunt în general descrise în propriul lor sistem de referință local și sunt poziționate pe scenă prin transformări adecvate. Cele mai multe transformări afine folosite, cum ar fi homothy , rotație și translație , poate fi descrisă într-un spațiu proiectiv cu o matrice 4x4: ele sunt aplicate prin înmulțirea matricei cu patru componente vectorului care reprezintă fiecare punct de control al curbei. A patra dimensiune se numește coordonată omogenă . O transformare este asociată cu fiecare nod al graficului de scenă , care se aplică și fiecărui nod transmis, recreând interacțiunea fizică dintre obiectele grupate. Chiar și în sistemele de modelare și redare care nu utilizează grafice de scenă, conceptul de transformare aplicat „vertical” este în general prezent.

***Mașina cu stări finite***

Automatele cu stări finite ne ajută să modelăm execution flow-uri, lucru necesar în diverse domenii cum ar fi matematica, inteligența artificială sau jocuri. Un FSM modelează o mașină ipotetică având un numar finit de stări. Trăsătura fundamentală a acestei mașini este ca numai una din aceste stări poate fi activă în oricare moment de timp. Asta înseamnă că pentru a putea executa toate acțiunile pentru care a fost proiectată, aceasta trebuie să își schimbe starea activă (sau curentă) în funcție de niște condiții prestabilite. Automatele finite, pe lângă faptul că au stări şi tranziţii, pot primi intrări şi pot da la ieşire diverse informaţii, ceea ce le dă şi utilitatea. Din punctul de vedere al condiţiilor în care automatele dau informaţii la output, ele sunt împărţite în două mari categorii generale: automatele Mealy și automatele Moore. La automatele Mealy, ieşirea depinde de starea curentă şi de input-ul curent, la automatele Moore, pe de altă parte, output-ul este determinat exclusiv de starea în care se află.

Pentru a implementa şi programa un automat finit, avem nevoie de o metodă de memorare a stării automatului într-un moment de timp oarecare, deci de o metodă de codificare. Cele mai folosite metode de codificare a stărilor unui automat sunt:

1. Numărare efectivă, în binar: de exemplu, la o codificare pe 4 biţi, se numără stările de la 0000, 0001, 0010, … 1111

2. Codul Gray, în care numerele consecutive diferă doar printr-un singur bit: 0000, 0001, 0011, 0010 …

3. Codul “one-hot”, în care avem nevoie de atâţia biţi câte stări există. În starea i, toţi biţii sunt 0, mai puţin bitul i, care e pe 1 (hot): 0010 = automatul se află în starea 2 (din cele 4 posibile)

4. Codul “one-cold”, asemănător cu cel de mai sus, numai că valorile sunt inversate: 1101 = automatul se află în starea 2 din cele 4 posibile.