30/11/2012

FACULTATEA DE AUTOMATICA SI CALCULATOARE

ELEMENTE DE GRAFICA PE CALCULATOR

Laborator 7

Liste de afisare

O lista de afisare reprezinta un grup de comenzi OpenGL stocate pentru o executie ulterioara. La invocarea unei liste de afisare , comenzile din ea sunt executate un ordinea in care sunt intalnite.

Majoritatea comenzilor OpenGL pot fi stocate intr-o lista de afisare. Listele de afisare pot imbunatati programul deoarece instructiunile sunt stocate pentru executii ulterioare. Este indicat sa se foloseasca liste de afisare in cazul in care se redeseneaza de mai multe ori aceeasi figura geometrica sau daca trebuie aplicat de mai multe ori un set de modificari de stare.

Listele de afisare au si dezavantaje. Listele foarte mici nu vor imbunatati executia programului datorita overheadului executiei listei. Un alt dezavantaj consta din faptul ca o lista de afisare nu poate fi modificata si continutul sau nu poate fi citit.

Fiecare lista de afisare este identificata printr-un index intreg. La crearea unei liste trebuie sa nu se aleaga un index deja folosit altfel se va sterge lista de afisare corespunzatoare acelui indice. Pentru a evita acest lucru se poate folosi :

- *GLuint glGenLists(GLsize i range);* Aceasta functie va genera unul sau mai multi indici nefolositi. Functia aloca un domeniu de *range* numere nefolosite si valoarea intoarsa reprezinta indicele de inceput al intervalului. Functia intoarce 0 daca nu este disponibil numarul de indici ceruti sau daca *range* este 0.

- *void glNewList(GLuint list, GLenum mode);* Functia specifica inceputul unei liste de afisare.Functiile care vor fi apelate (pana la intalnirea functie *glEndList()*) sunt stocate in lista de afisare.
    - *list* : un intreg pozitiv diferit de 0 care identifica in mod unic lista de afisare
    - valorile posibile ale parametrului *mode* sunt :
        - GL_COMPILE : nu se doreste executarea intructiunilor la plasarea in lista
        - GL_COMPILE_AND_EXECUTE : se doreste executarea intructiunilor la plasarea in lista

- *void glEndList(GLuint list, GLenum mode);* Functia marcheaza sfarsitul unei liste de afisare
- *void glCallList(GLuint list);* Functia executa lista de afisare specificata de parametrul *list*. O lista de afisare poate fi executata de mai multe ori si de asemenea se poate imbina cu apeluri in modul imediat.

Alpha Testing & Z-Buffer & Blending & Obiecte transparente

Testul alpha permite acceptarea sau respingerea desenarii unui fragment in functie de valoarea alpha (opacitatea) asociata acelui fragment. Pentru activarea acestui test se apeleaza :

- *void glEnable(GL_ALPHA_TEST);* Testul consta in compararea valorii de opacitatea asociata fragmentului cu o valoare de referinta *ref*
- *void glAlphaFunc(GLenum func, GLclampf ref);* Functia de comparatie (*func* poate fi una din constantele :
  - GL_LESS
  - GL_LEQUAL
  - GL_EQUAL
  - GL_NOTEQUAL
  - GL_GEQUAL
  - GL_GREATER
  - GL_NEVER
  - GL_ALWAYS

Testul de adancime (z-buffer) este folosit pentru a determina daca o primitiva va fi sau nu afisata in functie de "adancimea" sa.

Pentru a fi folosit, fereastra de afisare trebuie sa aiba asociata un buffer de adancime. Astfel parametrul intreg al functiei GLUT *InitDisplayMode* trebuie sa contina bitul GLUT_DEPTH pozitionat pe valoarea 1 si pentru a activa testul trebuie folosit si apelul:

- *glEnable(GL_DEPTH_TEST)*

Functia de comparatie intre adancimea fragmentului si ce a valorii corespunzatoare care a fost deja memorata in z-buffer poate fi specificata prin apelul functiei :

- *void glDepthFunc(GLenum func);* Parametrul *func* are aceleasi valori posibile ca in cazul functiei *glAlphaFunc*

Testul de combinare (blending) este specific modului RGBA de specificare a culorilor scenei de vizualizat. In acest mode de afisare fiecarui fragment i se asociaza o pondere de amestec a colorii sale.

Daca valoarea A este 0 sau lipseste atunci fiecare nou fragment va suprascrie pixelul corespunzator din memoria video. Se poate deci considera ca valoarea A reprezinta gradul de opacitate (sau de transparenta a unui fragment). Modul de combinare a culorilor fragmentului nou care se deseneaza (si care va fi numit in continuare *sursa*) si respectiv a pixelului corespunzator din memoria video (numit in continuare *destinatie*) este specificat cu ajutorul functiei :

- *void glBlendFunc(GLenum fsursa, GLenum fdest);* Parametrii *fsursa* si *fdest* sunt asociati celor 2 factori de amestec sursa si destinatie

Cei 2 factori de amestec au fiecare cate 4 componente care corespund coeficientilor de amestec pentru fiecare culoare fundamentala si factorului de opacitate.

Fie (Sr,Sr,Sg,Sa) si (Dr,Dg,Db,Da) cei 2 factori de amestec. Notam cu (Rs,Gs,Bs,As) respectiv (Rd,Gd,Bd,Ad) informatia de culoare asociata fragmentului sursa respectiv destinatiei. In urma procesului de *blending* va rezulta un pixel cu coeficientii RGBA : (RsSr+RdDr,GsSg+GdDg,BsSb+BdDb,AsSa+AdDa)

Parametrii functiei (*glBlendFunc* pot fi :

- GL_ZERO -> corespunde unui factor de amestec (0,0,0,0)
- GL_ONE -> (1,1,1,1)
- GL_SRC_ALPHA -> (As,As,As,As)
- GL_ONE_MINUS_SRC_ALPHA -> (1-As,1-As,1-As,1-As)
- GL_DST_ALPHA -> (Ad,Ad,Ad,Ad)
- GL_ONE_MINUS_DST_ALPHA -> (1-Ad,1-Ad,1-Ad,1-Ad)

Spre exemplu pentru a afisa un desen rezultat prin compunerea in proportii egale a doua imagini se pot genera pe rand cele doua imagini setand factorul destinatie la GL_ONE si factorul sursa la valoarea GL_SRC_APLHA unde valoarea A asociata fiecarui fragment este 0.5

Modul de lucru cu compunerea culorilor se ativeaza cu *glEnable(GL_BLEND)* si dezactiveaza cu *glDisable(GL_BLEND)*

Vizualizarea obiectelor transparente se poate simula in modul urmator :

- se afiseaza scena acceptand numai fragmentele cu alpha egal cu 1 (perfect opace)
- se dezactiveaza scrierea in z-buffer (*glDepthMask(GL_FALSE)*)
- se redeseneaza scena acceptand numai fragmente cu alpha mai mic decat 1 (transparente) si folosindu-se ca functie de amestecare *glBlendFunc(GL_SRC_APLHA,GL_DST_ALPHA)*

Observatie : in mod normal pentru a fi redat corect fenomenul de transparenta ar trebui ca si corpurile transparente sa fie afisate in ordinea adancimii lor realizandu-de o sortare a acestora.

Picking(selectie)

OpenGL pune la dispozitie un mecanism de selectie a obiectelor in spatiul 3D. Pasii pentru a realiza aceasta operatie sunt urmatorii :

- 1. Preluarea coordonatelor de la mouse din interiorul ferestrei de afisare
- 2. Intrarea in modul de selectie
- 3. Redefinirea volumului vizual in jurul cursorului mouselui
- 4. Randarea scenei
- 5. Iesirea modului de selectie si identificarea primitivelor care au fost desenate in volumul vizual redefinit
- Pentru a identifica obiectele renderizate acestea trebuiesc sa fie *numite*. API-ul OpenGL permite acest lucru numele asignate obiectelor regasindu-se intr-o *stiva de nume* Numele in realitate sunt reprezentate prin intregi. Manipularea *stivei de nume* se realizeaza astfel :
  - *void glInitNames(void)*; creaza o stiva de nume goala
  - *void glPushName(GLuint name)*; adauga in varful stivei pe *name*
  - *void glPopName(void)*; sterge un nume din varful stivei
  - *void glLoadName(GLuint name)*; inlocuieste varful stivei cu *name*

In mod normal atunci cand primitivele sunt randate pentru selectie ele sunt desenate intre un *glPushName(name) si glPopName()* care identifica primitiva sau grupul de primitive (de exemplu cele sase fete ale unui cub pot fi desenate intre glPushName(CUB) ... glPopName())

- In *modul de selectie* nici un obiect nu este de fapt randat in memoria video (framebuffer). In schimb numele obiectelor (plus informatia de adancime) sunt colectate intr-un vector.
- In terminologia OpenGL un *hit* are loc de fiecare data cand o primitiva este desenata in *modul de selectie* . *Hit recordurile* sunt stocate in *bufferul de selectie*
- Cand OpenGL iese din *modul de selectie* intoarce *bufferul de selectie* ce va contine un set de *hit records* Avand in vedere ca pentru fiecare *hit record* sunt disponibile si informatiile de adancime se poate determina usor care obiect a fost selectat chiar daca unul sau mai mult obiecte sunt suprapuse.
- Structura unui *hit record* este urmatoarea :
  - primul camp reprezinta numarul de nume care le contine *hit recordul* (un obiect poate sa aiba mai multe nume)
  - al doilea si al treilea camp reprezinta adancimea minima si cea maxima asociata pt *hit*
  - o secventa de nume avand totalul corespunzator cu numarul acestora din primul camp (poate sa si lipseasca daca primul camp are valoarea 0)
- Inainte de intrarea in *modul de selectie* trebuie specificat *bufferul de selectie* astfel :
  - *void glSelectBuffer(GLsizei size, GLuint *buffer);*
- Intrarea in *modul de selectie* se face astfel :
  - *void glRenderMode(GL_SELECT);*
- Intoarcerea in *modul normal de randare* se face astfel :
  - *nhits glRenderMode(GL_RENDER);*

Functia intoarce numarul de *hit records* ce au fost create si stocate in *bufferul de selectie* in cadrul procesului de selectie.