4. Fase - Azalpenak

Laburpena

Azken fase honetan argiak eta objektuen materialak kudeatzen hasi behar gara. Horretarako hainbat tokitan aldaketak egin beharko dituzue (argien erabilera gaitzeko, Z-bufferra erabiltzeko, etab.). Horretaz gain, argien zein materialen ezaugarriak kudeatu beharko dituzue.

1 Deskripzio globala

Argiak erabiltzeko ondoko pauso hauek jarraitu behar dira:

- Z-bufferraren erabilera gaitu, sistemak atzean dauden planoak ezaba ditzan.
- Erpin/aurpegi bakoitzaren bektore normalak kalkulatu.
- Objektuak marraztean, normalen informazioa sartu.
- Argiztatze-sistema gaitu.
- Erabili nahi ditugun argiak gaitu.
- Argien parametroak (koloreak, mota, etab.) sisteman sartu.
- Objektu bakoitzeko bere materialaren propietateak (koloreak, gehien bat) sisteman sartu.

Hurrengo ataletan pauso hauek banan-banan aztertuko ditugu.

2 Z-bufferra

Z-bufferra atzeko planoak ebazteko sistema bat da. Bere funtzionamendua nahiko erraza da. Objektuak banan-banan prozesatzen dira. Poligono baten puntu bat prozesatzean, bere kolorea kalkulatzen da eta informazio hori dagokion pixelaren posizioan sartzen da (*frame* bufferrean). Horretaz gain, beste buffer batean, Z-bufferrean, puntu horren sakonera gordetzen da. Puntu berri bat sartu behar badugu pixel berdinean (eszenaren beste puntu bat proiekzio planoaren toki berdinean proiektatzen dena, alegia), sartu aurretik bere Z koordenatua Z-bufferrean dagoen balioarekin alderatzen da. Aurretik badago, sartzen da, atzetik badago, ez.

Orain arte ez dugu Z-bufferra erabili. *OpenGL*k eta *glut* liburutegiek Z-bufferra erabiltzeko, ondoko kodea erabili behar dugu (sistemaren hasieraketan):

```
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
glEnable(GL_DEPTH_TEST);
```

Lehendabiziko funtzioa orain arte erabiltzen genuen, baina `GLUT_DEPTH` parametrorik gabe. Bigarren funtzioari dagokionez, Z-bufferra gure leihorako ondo gaitzeko *leihoa sortu eta gero exekutatu behar* da.

Beste gauza bat kontutan hartu behar duguna bufferren *garbiketa* da. `display` funtzioan, orain arte, `glClear` funtzioa *frame* edo *color* bufferra garbitzeko bakarrik erabiltzen genuen; orain, Z-bufferra ere garbitu beharko dugu. Beraz, `glClear` funtzioan ondoko aldaketa sartu behar dugu:

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Ikasturtea: 3. Maila
Irakaslea: Borja Calvo
Telefonoa: 943 01 50 13
E-posta: borja.calvo@ehu.es

GIPUZKOAKO CAMPUSA
CAMPUS DE GUIPUZCOA
Paseo Manuel de Lardizabal, 1
20018 Donostia (Gipuzkoa)

3 Normalak kalkulatzen

Argiztatze-sistemak kolorea ondo kalkula dezan, erpin bakoitzaren bektore normala behar du. Sistemak ez du bektore hauek kalkulatzen eta, hortaz, gu arduratu behar gara pauso horretaz. Kalkulu horiek nahiko *garestiak* dira, askotan poligono kopurua oso handia baita. Hori dela eta, kalkuluak *behin bakarrik* egin behar dira, objektu bakoitza kargatzean, adibidez.

Behin objektua kargaturik, egituran aurpegi eta erpin guztien informazioa izango dugu. Informazio hori osatu beharko dugu normalen informazioarekin (eta, hortaz, objektuan bertan eremu berria(k) sortu beharko d(it)uzue).

Aurpegi guztiak poligonoak dira eta, hortaz, beraien erpin guztiak plano berdinean daude. Hori dela eta, edozein hiru puntu hartzen baditugu (lehenengo hirurak, poligono guztiek gutxienez hiru puntu izango baitituzte) bi bektore kalkula ditzakegu, zeinek plano berdinean dauden. Bi bektore horien produktu bektoriala[1] kalkulatuz gero, emaitza (bektore bat dena) poligonoarekiko perpendikularra izango da. Behin normalizatuta, bektore hori poligonoaren normala izango da.

Demagun $\mathbf{e}^0$, $\mathbf{e}^1$ eta $\mathbf{e}^2$ lehenengo poligonoaren lehenengo, bigarren eta hirugarren erpinak direla. Bi bektore kalkula ditzakegu ,$\mathbf{a} = \mathbf{e}^1 - \mathbf{e}^0$ eta $\mathbf{b} = \mathbf{e}^2 - \mathbf{e}^0$. Bi bektore horiek plano berdinean daude eta beraien produktu bektoriala, behin normalizaturik, aurpegiaren bektore normala izango da: $\mathbf{n} = \frac{\mathbf{a}\times\mathbf{b}}{|\mathbf{a}\times\mathbf{b}|}$.

Ikus dezagun adibide bat. Demagun hiru puntu hauek ditugula: $e^0 = (1,1,1)$,$e^1 = (1,2,1)$ eta $e^2 = (2,3,2)$. Bi bektore kalkulatzen ditugu: $\mathbf{a} = \mathbf{e}^1 - \mathbf{e}^0 = (1,2,1) - (1,1,1) = (0,1,0)$, $\mathbf{b} = \mathbf{e}^2 - \mathbf{e}^0 = (2,3,2) - (1,1,1) = (1,2,1)$. Orain, bektore normala lortzeko produktu bektoriala kalkulatzen dugu: $\mathbf{a} \times \mathbf{b} = (1,0,-1)$. Bektore honen modulua ez da 1, $\sqrt{1^2+0^2+(-1)^2} = \sqrt{2}$ baizik. Hortaz, bektore normal unitarioa lortzeko koordenatu guztiak $\sqrt{2}$ balioarekin zatitu behar ditugu. Hori dena kontutan hartuz, bektore normala $(\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}})$ izango da.

Objektuak marrazterakoan, ondo argiztatzeko, objektuen aurpegi bakoitza sisteman sartzean erpinen bektore normala ere sartu beharko dugu:

```
glBegin(GL_POLYGON);
...
glNormal3d(normal.x,normal.y,normal.z);
glVertex3d(coord.x, coord.y, coord.z);
...
glEnd();
```

Kontutan hartu, aurpegiek bektore normal bakarra izan arren, *informazioa aurpegiaren erpin bakoitzeko sartu behar dela*, hau da, `glVertex3d` bakoitzeko `glNormal3d` komandoa izango dugu.

4 Argiztatze-sistema martxan jartzen

Argiztatze-sistema erabiltzeko argien eta objektuen materialak definitu behar ditugu. Gero, argiztatze-sistema eta erabiliko ditugun argiak gaitu behar ditugu.

4.1 Argien propietateak

Nahiz eta inplementazio batzuetan gehiago egon, printzipioz *OpenGL*n 8 argi ditugu eskuragarri. Bakoitzak identifikadore bat du, `GL_LIGHT0`tik `GL_LIGHT7`ra.

Argien propietateak definitzeko bi funtzio mota erabiltzen dira, bata (`glLightfv`) bektore motako propietateak sartzeko eta bestea (`glLightf`) balio numerikoak sartzeko. Funtzioen sintaxia hau da:

- `glLightfv (GLenum light, GLenum propietatea, const GLfloat * parametroak)`
- `glLightf (GLenum light, GLenum propietatea, const GLfloat parametroak)`

Hau da, lehendabizi zein argiaren propietateak ezarri nahi ditugun adierazi behar dugu, gero zein da aldatu nahi dugun propietatea eta, azkenik, esleitu nahi diogun balioa. Bektore motakoa bada, `GLfloat` bektore bat (pointer bat, alegia) izango dugu eta balio numeriko motakoa bada, berriz, `GLfloat` bat.

[1] Produktu bektoriala kalkulatzeko: $\mathbf{n} = (n_x, n_y, n_z) = \mathbf{a} \times \mathbf{b} = (a_y b_z - a_z b_y, a_z b_x - a_x b_z, a_x b_y - a_y b_x)$.

Hainbat propietate definitu behar dira; argi motaren arabera propietate batzuk ezarri beharko ditugu. Guztientzat, hiru kolore mota definitu behar dira:

- `GL_DIFFUSE` - Hau argiaren kolore nagusia da. Lehendabiziko argiarentzat balio lehenetsia zuria da, hau da, (1.0,1.0,1.0,1.0)[2]; beste gainontzekoentzat, berriz, beltza (0.0,0.0,0.0,0.0).
- `GL_AMBIENT` - Argien izpiak aurpegi batekiko paraleloak badira poligonoaren kolorea, printzipioz, beltza izango da. Balio hori aldatu nahi izanez gero, kolore honen bidez *giro* argia defini dezakegu. Aldatzen ez badugu, kolore hau beltza da. Argiaren osagai honentzat grisa iluna kolore egokia izan daiteke, esate baterakok, (0.2,0.2,0.2,1.0).
- `GL_SPECULAR` - Objektua distiratsua denean, distira horiek aparte kalkulatzen dira, argiaren osagai hau erabiliz. Balio lehenetsiak `GL_DIFFUSE` kolorearen berberak dira.

Jarraian duzu lehendabiziko argiaren balioak aldatzeko kodea (`display` funtzioan sartu behar da, objektuak marraztu aurretik):

```
GLfloat horia [4] = {0.0, 1.0, 1.0, 1.0};
GLfloat grisa [4] = {0.2, 0.2, 0.2, 1.0};
GLfloat txuria [4] = {1.0, 1.0, 1.0, 1.0};
glLightfv (GL_LIGHT0, GL_AMBIENT, grisa);
glLightfv (GL_LIGHT0, GL_DIFFUSE, horia);
glLightfv (GL_LIGHT0, GL_SPECULAR, txuria);
```

Mundu errealean argi iturri bat dugunean, iturritik urrundu ahala argiaren intentsitatea jaisten da. Ahultze hori hiru faktoreen bidez kontrola daiteke, ondoko formulan erabiliz:

$$I(d) = \frac{I_0}{a_c + d \cdot a_l + d^2 \cdot a_q},$$

non $I(d)$ d distantziara dagoen puntu baten intentsitatea den, I_0 argiaren jatorrizko intentsitatea den, d distantzia den eta a_c, a_l eta a_q ahuldura faktoreak diren. Faktorehoriek ondoko parametroen bidez sar ditzakegu:

- `GL_CONSTANT_ATTENUATION` - Ahultze-konstantea (a_c). Ia kasu guztietan 1.0 izan behar da.
- `GL_LINEAR_ATTENUATION` - Ahultze-lineala (a_l). Ez badugu ezer ere ez aldatzen 0.0 da.
- `GL_QUADRATIC_ATTENUATION` - Ahultze-koadratikoa (a_q). Hau ere 0.0 da. Ahultze efektua lortu nahi izanez gero, aldatzen den parametro tipikoa hau da (1.0 jarriz, adibidez).

Demagun lehendabiziko argiaren ahultzea ezarri nahi dugula. Ondoko kode hau erabil genezake:

```
glLightf (GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1.0);
glLightf (GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.0);
glLightf (GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 1.0);
```

Beste hainbat parametro daude, baina beraien balioak argi motaren araberakoak direnez, lehendabizi argi motak ikusiko ditugu.

Hiru argi mota simula daitezke *OpenGL*n: eguzkia, bonbillak eta fokuak. Lehendabizikoa argiri *direkzionala* esango diogu, eszenaren puntu guztietan norabide berdina baitu. Argi horrek ez du kokapen konkretu bat, hau da, infinituan dagoela suposatuko dugulako. Beste biak argi *posizionalak* dira, kokapen konkretua baitaukate. Bien arteko diferentzia nagusia argiaren igortzea da. Bonbillak norabide guztietan igortzen du argia; fokuak, berriz, norabide konkretu batzuetan bakarrik. Aspektu guzti hauek ondoko parametroen bidez kontrola daitezke:

- `GL_POSITION` - Argi posizionaletan (bonbillan eta fokuan), parametro honek kokapena ezartzen du. Argi direkzionaletan (eguzkia) parametro hau *norabidea* ezartzeko erabiliko dugu. Hau da, lehenengo bi kasutan puntu bat sartu behar dugu eta bestean bektore bat. *Koordenatuak sistema homogeneoan* sartzen direla kontutan hartuz, *azken koordenatuan 0.0* jartzen badugu *argi direkzionala* lortuko dugu *eta 1.0 jartzen badugu* argi posizionala. Beste era batean esanda, eguzki bat nahi badugu, bere norabidea (x,y,z,0.0) balioak erabiliz sartuko dugu eta bonbilla bat edo foku bat nahi izanez gero, kokapena (x, y, z, 1.0) balioak erabiliz ezarriko dugu.

[2]Balioak gorria, berdea, urdina eta gardentasuna dira, 0-tik 1-era definituta.

- `GL_SPOT_CUTOFF` - Bi aukera ditugu. Foku motako argi bat nahi badugu, fokuaren *zabalera* angelu honen bidez definituko dugu. Balioak 0 eta 90 tartean egon behar dira kasu honetan, eta gradutan sartu beharko ditugu. Geroz eta angelu handiagoa orduan eta foku *zabalagoa*. Bonbilla nahi izanez gero, balio berezi bat, 180, sartu behar dugu.
- `GL_SPOT_DIRECTION` - Fokuaren kasuan, kokapenaz gain, norabidea ere behar dugu. Parametro honek norabide hori definitzen du. Kontutan hartu behar da parametro honetan, posizioan ez bezala, beti bektore bat sartu behar dugula eta, hortaz, ez dira koordenatu homogeneoak behar. Hori dela eta, bakarrik hiru balio sartu behar ditugu, x, y eta z, eta ez lau.
- `GL_SPOT_EXPONENT` - Parametro honen bidez fokuen argia/iluntasuna trantsizioaren 'gogortasuna' kontrola daiteke.

Bonbilla bat (0, 10, 0) puntuak jartzeko, ondoko kodea erabil dezakegu:

```
GLfloat kokapena [4] = {0.0, 10.0, 0.0, 1.0};
glLightfv (GL_LIGHT0, GL_POSITION, kokapena);
glLightf (GL_LIGHT0, GL_SPOT_CUTOFF, 180.0);
```

Eguzkia nahi badugu, (1, 0, 0) norabidean, kodea hau izango litzateke:

```
GLfloat norabidea [4] = {1.0, 0.0, 0.0, 0.0};
glLightfv (GL_LIGHT0, GL_POSITION, norabidea);
```

Foku kasua konplexuena da. Ondoko kodeak, (0,10,0)-n kokatutako beherantz begiratzen duen eta 20 graduko zabalerako duen foku bat sortzen du:

```
GLfloat kokapena [4] = {0.0, 10.0, 0.0, 1.0};
GLfloat norabidea [4] = {0.0, -1.0, 0.0};
glLightfv (GL_LIGHT0, GL_POSITION, kokapena);
glLightfv (GL_LIGHT0, GL_SPOT_DIRECTION, norabidea);
glLightf (GL_LIGHT0, GL_SPOT_CUTOFF, 20.0);
glLightf (GL_LIGHT0, GL_SPOT_EXPONENT, 1.0);
```

Objektuekin gertatzen den bezala, puntuak eta bektoreak sisteman sartzean `GL_MODELVIEW` *matrizearekin biderkatzen dira*. Honek esan nahi du matrize horretan aldaketa matrizeren bat izanez gero, aldaketa horiek argiari ere aplikatuko zaizkiela.

4.2 Argiak gaitzen

Behin argiak definituta, hauek gaitu behar dira. Hori behin bakarrik egingo dugu beti 'pizturik' mantendu nahi baditugu, ondoko kodea exekutatuz:

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
```

Lehenengo komandoak argiztatze-sistema orokorra gaitzen du eta bigarrenak lehendabiziko argia. Sistema desgaitzeko (eta, hortaz, objektuak orain arte ikusi dugun moduan bistaratzeko) edo argi bat itzaltzeko, `glDisable` funtzioa erabil daiteke.

4.3 Materialak definitzen

Argiak gaiturik daudenean, objektuen kolorea definitzeko bi era ditugu. Orain arte bezala, `glColor` funtzioa erabil dezakegu. Hori lortzeko ondoko kode hau beharko dugu:

```
glEnable(GL_COLOR_MATERIAL);
```

Beste aukera objektuaren materiala definitzea eta erabiltzea da. Hori lortzeko, alde batetik goiko aukera desgaitu behar dugu (`glDisable(GL_COLOR_MATERIAL);`) eta gero, objektua marraztu aurretik, bere materiala ezarri behar dugu. Materialak definitzeko lau kolore gehi beste balio bat definitu behar dira:

- `GL_DIFFUSE`
- `GL_AMBIENT`

- GL_SPECULAR
- GL_EMISSION
- GL_SHININESS

Lehenengo hirurak materialaren portaera argiaren osagai bakoitzarekiko definitzen dute. Laugarrena, berriz, bakarrik materialak argia igortzen badu erabiliko dugu, eta azkena materialaren dirdira ezartzeko erabiltzen da. Interneten topa daitezke zenbait webgune hainbat material imitatzeko erabili behar diren balioak jasotzen dituztenak[3].

Adibide gisa, ikus dezagun nola defini dezakegu kobrea imitatzeko materiala:

```
GLfloat ambient[4] = {0.19125, 0.0735, 0.0225, 1.0};
GLfloat diffuse[4] = {0.7038, 0.27048, 0.0828, 1.0};
GLfloat specular[4] = {0.256777, 0.137622, 0.0806014, 1.0};
glMaterialfv (GL_FRONT_AND_BACK, GL_AMBIENT, ambient);
glMaterialfv (GL_FRONT_AND_BACK, GL_DIFFUSE, diffuse);
glMaterialfv (GL_FRONT_AND_BACK, GL_SPECULAR, specular);
glMaterialf (GL_FRONT_AND_BACK, GL_SHININESS, 0.1);
```

Ikus daitekeenez, parametroak sartzeko funtzioak argien kasuan ikusi ditugunen antzerakoak dira, *Light* hitza *Material* hitzarekin ordezkatuz.

[3]Esate baterako, hemen: `http://devernay.free.fr/cours/opengl/materials.html`.