به نام خدا

دانشگاه شهید بهشتی

دانشکده علوم ریاضی

گزارش تمرین اول گرافیک (کار با کتابخانه openGL)

زهرا دهقانی تفتی (۹۶۲۲۲۰۳۷)

تاریخ تحویل : ۱۰ اردیبهشت ۱۴۰۰

فهرست مطالب

تمرین اول-تولید اجسام سه بعدی و نور پردازی

ه به کمک توابع آماده GLUT ، چند جسم هندسی ۳ بعدی مختلف رسم کنید.

الف – برای هر کدام از اشکال خود، یک ماده دلخواه تعریف کنید.

ب – برای تصویر خود نورپردازی مناسب را انجام دهید. (نور پردازی بصورت دلخواه می باشد، هم از نظر تعداد منابع نور هم از نظر مکان منابع نور)

ج- با مطالعه دستورات مربوط به نگاشت بافت، بر روی تصویر خود، بافت دلخواه را ایجاد کنید. (این بخش نمره مثبت دارد)❊

د – برای یکی از اجسام خود، یک پویانمای دلخواه انجام دهید.

پاسخ:

در این تمرین با استفاده از توابع آماده Glut ، کره و مکعب را رسم می‌کنیم. در این مثال از دو منبع نور استفاده می‌کنیم و برای این دو جسم ماده دلخواه را معرفی می‌کنیم که این تنظیمات در قسمت initGL انجام شده است. همچنین برای هر دو شکل پویانمایی دلخواه در نظر گرفته شده است.

ابتدا باید کتابخانه‌های لازم را فراخوانی کنیم:

```
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h
```

سپس متغیرها را تعریف می کنیم. متغیرها به ترتیب برای عنوان پنجره خروجی، زاویه ی چرخش کره و مکعب و تایم مورد نیاز برای رفرش انیمیشن هستند.

```
char title[] = "3D Shapes";
GLfloat angleSphere = 0.0f;  // Rotational angle for sphere
GLfloat angleCube = 0.0f;     // Rotational angle for cube
int refreshMills = 15;        // refresh interval in milliseconds
```

بعد از اینکار نیاز است تا تابع initGL را تعریف کنیم. این تابع برای انجام وظایفی که فقط یکبار باید انجام شوند مثل پاک کردن رنگ پنجره خروجی، تنظیمات مربوط به نور (اگر بخواهیم منبع نور ساکن باشد) و متریال استفاده می‌شود. این تابع فقط یکبار آن هم در تابع main فراخوانی می‌شود.

در تابع زیر آرایه های مربوط به منابع نور (مات، روشن و...) و متریال جسم ها را مشخص میکنیم.ما دراینجا از دو منبع نور استفاده کردیم و مکان هر کدام را با استفاده از آرایه مشخص کردیم. در هنگام استفاده از نور باید متریال اجسام نیز مشخص باشد تا بدانیم میزان جذب یا بازتاب نور توسط هر جسم چقدر است. برای استفاده از نورپردازی و متریال باید ابتدا آنها ار فعال کنیم.

```
void initGL() {
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // پنجره را پاک می کند و رنگش را مشکی می کند
    glClearDepth(1.0f);                   // عمق پس زمینه را تنظیم میکنیم
    glEnable(GL_DEPTH_TEST);    //بافر عمق را برای برنامه  فعال می کنیم
    glDepthFunc(GL_LEQUAL);     // مقدار مورد استفاده برای مقایسه بافر عمق
    glShadeModel(GL_SMOOTH);    // تنظیم سایه روشن

    GLfloat ambient[] = { 0.0, 0.0, 0.0, 0.1 };
    GLfloat diffuse[] = { 1.0, 1.0, 0.0, 1.0 };
    GLfloat specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat position[] = { 1.0, 1.0, 1.0, 0.0 };
    GLfloat positionn[] = { -1.0, 1.0, 1.0, 1.0 };

    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);
    glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
    glMaterialf(GL_FRONT, GL_SHININESS, 30);

    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);//light0
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
    glLightfv(GL_LIGHT0, GL_POSITION, position);

    glLightfv(GL_LIGHT1, GL_AMBIENT, ambient);//light1
    glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT1, GL_SPECULAR, specular);
    glLightfv(GL_LIGHT1, GL_POSITION, positionn);

    glEnable(GL_LIGHTING);                //فعال کردن نور پردازی
    glEnable(GL_LIGHT0);                  // فعال کردن منبع نور اول(source: gl-light0)
    glEnable(GL_LIGHT1);                  // فعال کردن منبع نور دوم (source: gl-light1)
    glEnable(GL_COLOR_MATERIAL);           // برای کار با رنگ فعلی جسم ها که در تابع نمایش مشخص شده
}
```

بعد از اینکار نوبت معرفی تابع display است که نشان دهنده ی صحنه نمایش ما است. در این تابع ابتدا باید بافرهای رنگ وعمق پاک شوند. سپس ماتریس را از جنس مدلسازی تعریف می کنیم تا بتوانیم عملیاتی مثل چرخش را در آن وارد و از آن خارج کنیم. سپس جسم ها با استفاده از توابع آماده Glut رسم می کنیم و جابجایی و چرخش مورد نیاز را انجام می دهیم. از glpushmatrix و glpopmatrix برای بی تاثیر کردن تغییرات یک قسمت بر روی قسمت دیگر استفاده می شود.

```
void display_3Dshapes() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear color and depth
buffers
    glMatrixMode(GL_MODELVIEW);     // To operate on model-view matrix

    // Render a sphere
    glPushMatrix();                 // Reset the model-view matrix
    glColor3f(1, 0, 0);
    glTranslatef(1.5f, 0.0f, -7.0f);  // Move right and into the screen
    glRotatef(angleSphere, 1.0f, 1.0f, 0.0f);  // برای پویا نمایی زاویه مکعب در طی زمان تغییر می کند
    glutSolidSphere(1, 16, 16); //SPEHERE
    glPopMatrix();

    // Render a cube
    glPushMatrix();              // Reset the model-view matrix
    glColor3f(0, 0, 1);
    glTranslatef(-1.5f, 0.0f, -6.0f);  // Move left and into the screen
    glRotatef(angleCube, 1.0f, 1.0f, 1.0f);  // برای پویانمایی زاویه مکعب در طول زمان تغییر می کند
```

```
    glutSolidCube(1); //CUBE
    glPopMatrix();

    glutSwapBuffers();   //از دو بافر استفاده شده برای انیمیشن و بافرها را جابجا می کند

    //زاویه های مربوط به چرخش مکعب و کره
    angleSphere += 0.2f;
    angleCube -= 0.15f;
}
```

در تابع timer انیمیشن و مقدار زاویه ی چرخش برای کره و مکعب کنترل می شود. قسمت مهم این تابع glutpostredisplay است که باعث میشود هربار صحنه از نو رسم شود بدون این تابع انیمیشن نداریم

```
void timer(int value) {
    glutPostRedisplay();      // Post re-paint request to activate display()
    glutTimerFunc(refreshMills, timer, 0); // next timer call milliseconds later
}
```

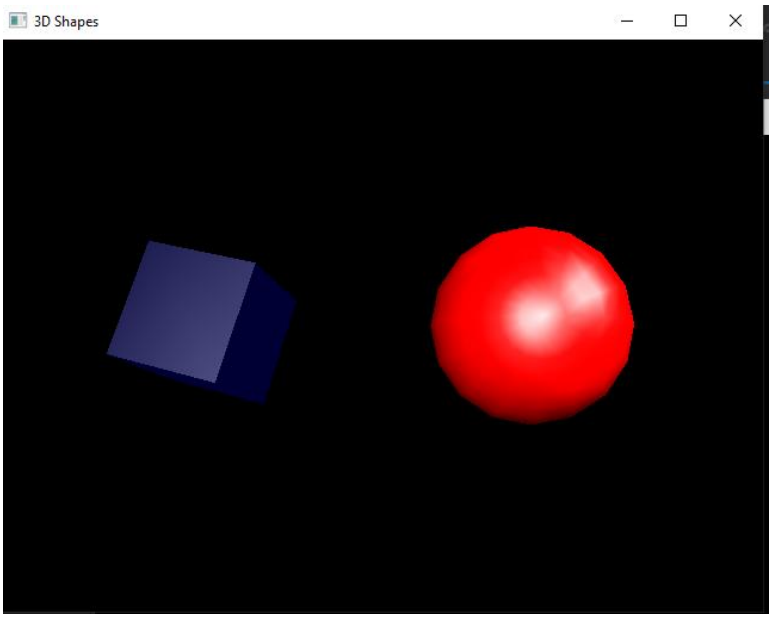
در تابع reshape اطلاعات مربوط به دوربین را داریم و مشخص می کند هنگام نغییر اندازه پنجره چه تغییری ایجاد می شود.

```
void reshape(GLsizei width, GLsizei height) {  // GLsizei for non-negative integer
   // Compute aspect ratio of the new window
    if (height == 0) height = 1;                // To prevent divide by 0
    GLfloat aspect = (GLfloat)width / (GLfloat)height;

    //  در هر تغییر اندازه پنجره آپدیت می شود

    glViewport(0, 0, width, height);

    // Set the aspect ratio of the clipping volume to match the viewport
    glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
    glLoadIdentity();             // Reset
    //از حالت پرسپکتیو دوربین استفاده می کنیم و زاویه دید را ۴۵ می گذاریم
    gluPerspective(45.0f, aspect, 0.1f, 100.0f);
}
```

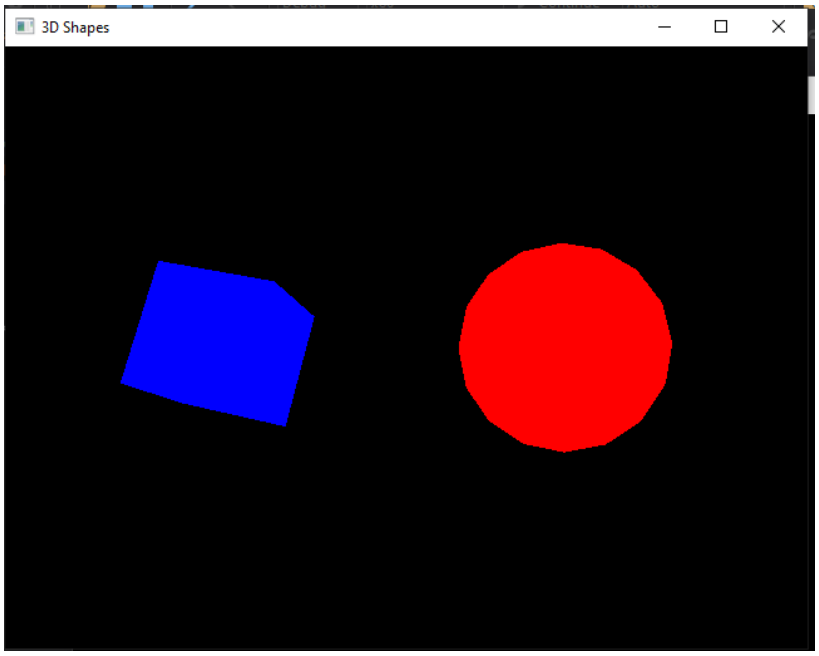
در تابع main اطلاعات مربوط به پنجره خروجی را داریم. در این تابع باید از تابع هایی که در بدنه برنامه معرفی کردیم استفاده کنیم.

```
int main(int argc, char** argv) {
    glutInit(&argc, argv);            // Initialize GLUT
    glutInitDisplayMode(GLUT_DOUBLE); // Enable double buffered mode
    glutInitWindowSize(640, 480);   // سایز پنجره خروجی
    glutInitWindowPosition(50, 50); // موقعیت گوشه سمت چپ بالای پنجره
    glutCreateWindow(title);          // پنجره را ایجاد می کند
    glutDisplayFunc(display_3Dshapes);       //  تمام رویدادهای مورد نیاز برای طراحی مجدد صحنه را در اینجا
میگذاریم
    glutReshapeFunc(reshape);       //اطلاعات دوربین
    initGL();                       // Our own OpenGL initialization
    glutTimerFunc(0, timer, 0);     // برای انیمیشن استفاده می شود و تابع تایمر را فراخوانی میکند
    glutMainLoop();                 // تابعی که همیشه برنامه را آپدیت نگه می دارد و لوپ ایجاد می شود
    return 0;
}
```

خروجی :



همانطور که می بینیم خروجی طبق انتظار تولید شده است. مکعب و کره در کنار یک دیگر در صفحه رسم شده اند و رنگ مکعب و کره طبق رنگی است که در glcolor3f برای آنها تعریف کرده‌ایم.اثرات نور پردازی و متریال اجسام نیز در خروجی مشخص است و هر دو جسم در حال چرخش در حول محورهای تعرف شده در کد هستند.



شکل بالا مربوط به زمانی است که نوز منبع نور و متریال تعریف نشده باشند.