KARADENİZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
BİLGİSAYAR GRAFİKLERİ LABORATUARI

OpenGL Uygulamaları

1. Giriş

OpenGL, en yaygın kullanılan grafik programlama kütüphanesidir. Hızlı ve basit bir şekilde etkileşimli, 2B-3B bilgisayar grafik programları yapmanıza olanak sağlar. Kullanım alanı çok yaygındır ve bilgisayar grafiklerinin hemen hemen tüm alanlarında yaygın olarak kullanılır. Bazı kullanım alanları: araştırma, bilimsel görselleştirme, eğlence ve görüntü efektleri, bilgisayar destekli tasarım, etkileşimli oyunlar...

OpenGL, donanım-bağımsız bir arayüzdür. Görüntüde bulunan nesneleri tanımlamak ve bu nesneler üzerinde gerek duyulan işlemleri gerçekleştirmek için gerekli komutları içerir. OpenGL 'in donanım-bağımsız olmasının nedeni, pencere işlemlerini (ekranda bir pencere oluşturmak gibi) yapan ya da kullanıcıdan girdi alan herhangi bir komutunun bulunmamasıdır. Belirtilen bu işleri gerçekleştirmek için varolan işletim sisteminin mevcut özellikleri kullanılır. Ancak işletim sisteminde pencere işlemlerini gerçekleştirmek karmaşık işlemler içerdiğinden tüm bu işlevleri barındıran ve işletim sistemlerine özel olarak yazılmış GLUT (Graphic Library Utility) kütüphaneleri bulunmaktadır.

OpenGL, 3D nesneleri tanımlamak için yüksek-seviyede komutlar içermez. Bunun yerine; nokta, doğru ve poligon gibi alt-seviye geometrik primitif (ilkel) nesneleri içerir ve bu primitif nesneleri kullanarak karmaşık grafik nesneleri tanımlamamıza olanak sağlar.

2. Programlama Dilleri, İşletim Sistemleri ve Pencere Sistemleri Seviyesi Destek

2.1. Programlama Dilleri

Bir çok uygulama geliştiricisi, OpenGL kütüphanesini üst seviye dillerde kullanmak için bu dillere has uygulama programlama arayüzleri geliştirmişlerdir. Bu dillerden bazıları şunlardır: Ada, Common Lisp, C#, Deplhi, Fortran, Haskell, Java, Perl, Pike, Python, Ruby, Visual Basic...

Tartışma Sorusu-1 : OpenGL kütüphanesi neden herhangi bir programlama dili ile kullanılır? Buna neden ihtiyaç duyulmuştur? Bu diller ile kullanılmasaydı OpenGL ile uygulama geliştirme bazında neler yapılamazdı? Tartışınız.

2.2. İşletim Sistemleri ve Pencere Sistemleri

OpenGL, yaygın olarak kullanılan tüm işletim sistemleri ve pencere sistemlerince desteklenir. Ağ protokolleri ve topolojilerinden tam bağımsızlık sağlar. Tüm OpenGL uygulamaların işletim sistemi ve pencere sistemine bakılmaksızın herhangi bir OpenGL UPA (Uygulama Programlama Arayüzü- API) uyumlu donanımda aynı görsel sonucu üretir. Desteklenen bazı işletim sistemleri ve pencere sistemleri aşağıda listenmiştir:

- Microsoft Windows
- Apple Mac OS
- Linux - Debian, RedHat, SuSE, Caldera
- X Pencere Sistemi (X Window Systems - daha çok GNU/Linux ve Unix benzeri işletim sistemlerinde kullanılan grafik arayüz altyapısıdır.

3. OpenGL Tabanlı Bazı Uygulama Geliştirme Arayüzleri

3.1. OpenGL ES (OpenGL for Embedded Systems /Gömülü Sistemler için OpenGL)

OpenGL ES taşınabilir (mobil) cihazlar, PDA'lar, video oyun konsolları gibi gömülü sistemler için geliştirilmiş 2B/3B uygulama geliştirme arayüzü ve grafik işleme dilidir. Cihazlardaki süzgeçlerin (filtreler) verimli çalışması için telefon GPU'su ile bereber kullanılır. Glut ve Glu gibi kütüphaneler içermez. Telif ücreti gerektirmez ve platformalar arası çalışabilir. Günümüzde çoğu modern cihaz üzerinde bulunur ve bir çok uygulamada kullanılmıştır. Örneğin, mobil cihazlar için geliştirilen fotoğraf paylaşma uygulaması olan Instagram'da OpenGL ES kullanılmıştır.

OpenGL destekli bazı cihazlar: Samsung taşınabilir telefonlar, BlackBerry OS 7.0 ve sonrası BlackBerry cihazları, Aplple (iPad, iPhone vs.), Google Native Client...

3.2. WebGL

Web sayfaları üzerinde 3 boyutlu grafikler oluşturmak için kullanılan platforma bağımsız ve ücretsiz bir uygulama geliştirme arayüzüdür. HTML 5'in web üzerinde yaygınlaşmasıyla birlikte kullanımı artmıştır. Güncel internet tarayıcılarının çoğu tarafından desteklenmektedir.

4. OpenGL

4.1. Kullanım Avantajları

OpenGL kullanarak grafikler oluşturmanın avantajları aşağıda sıralanmıştır.

- Platform bağımsızdır (Windows, Linux, Mac) ve tüm OpenGL UPA uyumlu donanımlar üzerinde çalışır.
- Çok çeşitli sistemler üzerinde koşulabilir. (Kişisel bilgisayarlar, iş istasyonları, süper bilgisayarlar, gömülü sistemler vs.)
- Sistem kaynaklarını optimum şekilde kullanır.
- Bir çok programlama dili tarafından çağırılarak kullanılabilir.
- Kolay anlaşılır, hızlı öğrenilir.
- İçerdiği işlevlerin belgelendirmesi çok iyi yapılmıştır ve ücretsiz bol miktarda eğitici dokümana sahiptir.
- IBM, Sony, Google, Intel, Samsung'un da içinde bulunduğu şirketler tarafından grafik alanında açık standartları oluşturması amacıyla desteklenir ve finanse edilir.

4.2. Open GL Utility (GLUT)

OpenGL platformdan bağımsız olduğu için bazı işlemler bu kitaplık ile yapılamaz. Örneğin kullanıcıdan klavye veya fare ile veri almak, bir pencere çizdirmek gibi işler hep kullanılan pencere yöneticisi ve işletim sistemine bağlıdır. Bu yüzden bir an için OpenGL'in platform bağımlı olduğu düşünülebilir. Çünkü çalışma penceresini her pencere yöneticisinde (her ortamda) farklı çizdirecek bir canlandırma programı yazmak demek her bilgisayarda çalışacak ayrı pencere açma kodu yazmak demektir. Bu ise OpenGL'in doğasına aykırıdır. Bu gibi sorunları aşmak için OpenGL Araç Kiti (GLUT - OpenGL Utility Toolkit) kullanılmaktadır. Bu yüzden bu deneyde GLUT kitaplığı kullanılarak klavye ve fare için işletim sisteminden bağımsız giriş/çıkış işlemleri yapılması sağlanmıştır.

Aşağıda sık kullanılan bazı pencere işlevleri listelenmiştir :

- **glutInit()** işlevi GLUT'ı ilkler, diğer GLUT rutinlerinden önce bu komutun yazılması zorunludur.
- **glutInitDisplayMode()** işlevi renk modunu belirlemektedir.
- **glutInitWindowPosition()** işlevi ekranın sol-üst köşesini baza alarak grafik penceresinin ekrandaki yerini belirler.
- **glutInitWindowSize()** işlevi pencerenizin büyüklüğünü ayarlar.
- **glutCreateWindow()** işlevi OpenGL conteksli bir pencere oluşturur.
- **glutDisplayFunc()** işlevi, çizim penceresinin içeriğinin yeniden gösterileceği durumlarda çalıştırılacak fonksiyonu (çizim işlemlerinin yapıldığı fonksiyon) çağırır. Parametre olarak bu fonksiyonun adını alır.
- **glutKeyboardFunc()** ve **glutMouseFunc()** işlevleri klavyenin veya farenin herhangi bir tuşuna basıldığında çalıştırılacak fonksiyonu çağırır.
- **glutReshapeFunc()** işlevi, pencere büyüklüğünün değişeceği durumlarda çalıştırılacak fonksiyonu çağırır.

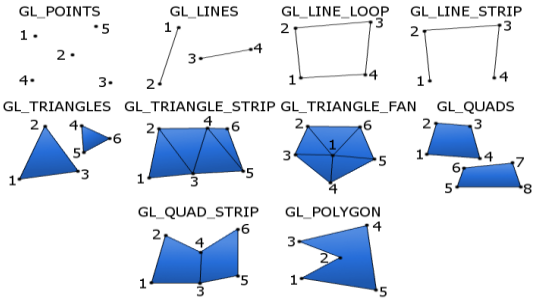
4.3. OpenGL Söz dizimi

OpenGL'de her komutunun önüne gl ön eki getirilmektedir (örneğin glBegin()) Aynı şekilde, tanımlanmış OpenGL sabitlerinin önüne GL ön eki getirilir (örneğin GL_ POLYGON). Ayrıca komut bildirindeki bazı ekler de bu komutlara birer anlam katmak için kullanılır. Örneğin glColor3f() komutunu incelersek, Color eki renk ile ilgili bir komut olduğunu, 3 eki 3 tane parametre aldığını ve f eki ise aldığı parametrelerin kayan noktalı sayı (float) tipinde olduğu anlaşılır.

4.4.OpenGL İlkel (Primitif) Geometrik Nesneleri

İlkel geometrik nesneler, OpenGL'in çizebildiği basit nokta, çizgi, poligon gibi nesnelerdir. (Şekil 1'de OpenGL ile çizilebilen ilkel geometrik nesneler gösterilmiştir.) Bu geometrik nesneler koordinat bilgileri ile tanımlanırlar ve bu koordinat bilgilerine köşe (vertex) denmektedir. OpenGL bu köşe bilgileri ile ilkel olan geometrik şekilleri çizebilmektedir. Fakat çizilecek olan nesnenin nokta, çizgi veya poligon olup olmadığını OpenGL'e bildirmek gerekir. Bu bildirim glBegin() fonksiyonu tarafından gerçekleştirilir. Ardından köşe bilgileri aktarılıp nesnenin çizimini ve çizme modunun bittiğini göstermek için glEnd fonksiyonu kullanılır. Aşağıdaki glBegin, glEnd fonksiyonları ve köşe değerleri ile bir poligon nesnesinin çizimi gösterimektedir:

```
glBegin(GL_POLYGON);                    //Poligon çizmeye başla komutu.
        glVertex2f(0.25, 0.25);         //1. köşenin x ve y bileşenleri
        glVertex2f(0.75, 0.25);         //2. köşenin x ve y bileşenleri
        glVertex2f(0.50, 0.75);         //3. köşenin x ve y bileşenleri
glEnd();                                //Poligon çizmeyi bitir komutu.
```

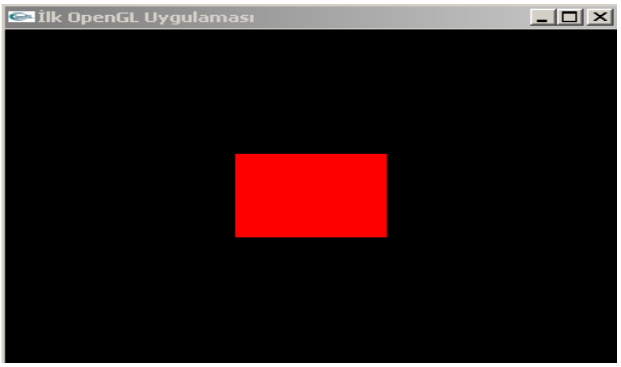


Şekil 1. İlkel (Primitif) Geometrik Nesneler

4.5. İlk OpenGL Uygulaması

Programda ilk olarak bir pencere oluşturulmakta daha sonra da display fonksiyonu ayarlanmaktadır. Display fonksiyonu içerisinde de her defasında çizilecek olan grafik çizilmektedir. Bu hali ile verilen kod basit bir OpenGL programının iskeletini oluşturmaktadır. Program çalıştırıldığında elde edilen ekran görüntüsü Şekil 2'de verilmiştir.

```
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>

void ayarlar(void){
    glClearColor(0.0,0.0,0.0,0.0);
    glOrtho(-2.0, 2.0, -2.0, 2.0, -1.0, 1.0);       //Koordinat sistemini ayarla
}

void display(void){
    glClear(GL_COLOR_BUFFER_BIT);           // Renk bufferını  temizle
    glColor3f(1.0, 0.0, 0.0);               //Renk değeri ata
    glBegin(GL_POLYGON);                            //Poligon çizmeye başla
    glVertex2f(-0.5, -0.5);                 //Köşe değerleri
    glVertex2f(-0.5, 0.5);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, -0.5);
    glEnd();                                        //Poligon çizimi bitir
    glFlush();                                      //Çizim komutlarını çalıştır
 }

int main(int argc, char **argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB );
    glutInitWindowPosition(0,0);
    glutInitWindowSize(500,400);
    glutCreateWindow("OpenGL Uygulamaları-I");
    ayarlar();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```



Şekil 2. İlk OpenGL Uygulaması - Dörtgen Çizimi

Tartışma Sorusu-2 : main fonksiyonu içerisinde pencere oluşturmak için kullanılan Glut kütüphanesine ait pencere fonksiyonlarına gönderilen parametreleri değiştirerek ortaya çıkan farkları inceleyiniz. glOrtho fonksiyonu ile koordinat sistemin nasıl değiştirildiğini gönderilen parametreleri değiştirerek gözlemleyiniz. Çizim nesnelerinin koordinat sisteminde nasıl yerleştirildiğini inceleyiniz.

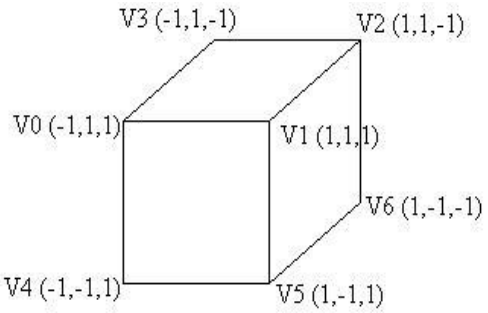
4.6. OpenGL Koordinat Sistemleri ve Dönüşümleri

OpenGL'de 3D grafik işlemlerinde birçok farklı koordinat sistemi kullanılır. Bunlar aşağıdaki gibidir ve bir uzaydan diğerine geçiş için dönüşüm matrisleri kullanılır.

- Nesne Uzayı (Object Space)
- Dünya Uzayı (World Space)
- Kamera Uzayı (Camera Space /Eye Space/View Space)
- Ekran Uzayı (Screen Space/Clip Space)

Nesne Uzayı: Geometrik nesneleri oluştururken nesnenin orjinine göre köşe değerlerinin hesaplanmasında kullanılan koordinat sistemidir. Örneğin, sekiz köşesi olan bir küpü geometrik olarak modellemek için köşe koordinat bilgileri kullanılabilir.

Vertex	Coordinates
0	-1,1,1
1	1,1,1
2	1,1,-1
3	-1,1,-1
4	-1,-1,1
5	1,-1,1
6	1,-1,-1
7	-1,-1,-1



Şekil 3. Modellenmiş Küp ve Köşe Değerleri

Dünya Uzayı : Nesne uzayında modellenen geometrik cisimlerin dünya koordinat sisteminde bir konuma yerleştirilmesi için kullanılan uzaydır. Geometrik nesneler model matrisler kullanılarak bu uzaya taşınır.

Örneğin, başlangıçta (0,0,0) konumundaki yukarıdaki küpü dünya uzayında (5, 0, 0) koordinatlarına taşımak istiyorsak küpün tüm köşe değerleri aşağıdaki gibi bir model matris ile çarpılmalıdır. Aşağıda, (-1, -1, 1) köşesinin +x yönünde 5 birim taşındığında dünya uzayındaki yeni koordinatları hesaplanmıştır.

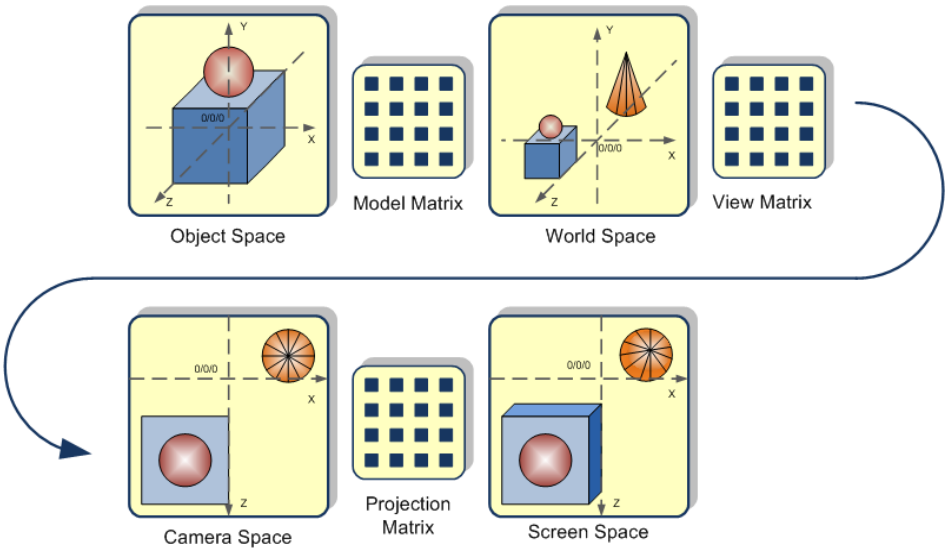
$$\begin{bmatrix} 1 & 0 & 0 & \mathbf{5} \\ 0 & 1 & 0 & \mathbf{0} \\ 0 & 0 & 1 & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{-1} \\ \mathbf{-1} \\ \mathbf{1} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{4} \\ \mathbf{-1} \\ \mathbf{1} \\ 1 \end{bmatrix}$$

Model Dönüşüm Matrisi | Nesne Uzayı Köşe Koor. | Dünya Uzayı Koordinatları

Kamera Uzayı: OpenGL kütüphanesi ile uzayda istenilen bir noktaya kamerayı koymak ve bu noktadan istenilen bir yöne istenilen açıyla bakmak için kullanılır. Dünya uzayından kamera uzayına dönüşüm için view matrisler kullanılır.

Projection Space: 3 boyutlu kamera uzayı üzerindeki görüntülerin 2 boyutlu ekranda görüntülenecek biçime dönüştürülmesi için kullanılır. Günümüzde, 3 boyutlu holografik ekranlara sahip olmadığımızdan bu dönüşüm gereklidir. Dönüşüm için projection matrisleri kullanılır.

Aşağıda, OpenGL'de kullanılan tüm koordinat sistemleri ve dönüşüm işlemleri gösterilmiştir.



Şekil 4. OpenGL Koordinat Sistemleri ve Dönüşümler

4.7. OpenGL ile Dönüşüm (Transformation) İşlemleri

4.7.1. Taşıma (Translation)

Bu işlemin amacı bir şekli mevcut konumundan bozulmadan farklı bir konuma taşımaktır. glTanslatef() ve glTranslated() fonksiyonları bu işlemi gerçekleştirir. İki farklı şekilde kullanılabilir:

```
void glTranslatef(GLfloat x, GLfloat y, GLfloat z);
void glTranslated(GLdouble x, GLdouble y, GLdouble z);
```

Örneğin, bir küpü koordinat sisteminin merkezinden (5, 5, 5) noktasına taşımak istenirse, ilk olarak modelview matrisini yüklenmeli ve ilklenmelidir. Daha sonra glTranslatef() fonksiyonu ile taşınmalıdır. Aşağıdaki kod yapacağımız işlemi gerçekleştirir:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glTranslatef(5.0, 5.0, 5.0);
KupCiz();                        //küp çizecek fonksiyon
```

Tartışma Sorusu-3 : glMatrixMode(GL_PROJECTION) ve glLoadIdentity işlevleri neden kullanılmaya ihtiyaç duyulmuştur? Modelview matrisi nedir? Araştırınız ve tartışınız.

4.7.2. Döndürme (Rotating)

OpenGL'de döndürme işlemi glRotate*() fonksiyonu ile gerçekleştirilmektedir. İki farklı şekilde kullanılabilir.

```
void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);
void glRotated(GLdouble angle, GLdouble x, GLdouble y, GLdouble z);
```

Örneğin, çizilen bir modeli y ekseni etrafında saat yönüne, 135 derece döndürmek istenirse, aşağıdaki kod bu işlemi gerçekleştirir. Burada y argümanın aldığı 1.0 değeri, y ekseni yöndeki birim vektörü belirtmektedir. İstenilen eksene göre döndürme işlemini yapmak için sadece birim vektörü belirtmek gerekir.

```
glRotatef(135.0, 0.0, 1.0, 0.0);
```

4.7.3. Ölçeklendirme (Scaling)

Modelin boyutundaki ayarlamaları yapmak için ölçeklendirme işlemi kullanılmaktadır. Nesnenin boyutları eksenlere göre büyültülüp küçültülebilir. OpenGL'de ölçeklendirme işlemi **glScale***() fonksiyonu ile gerçekleştirilir. İki farklı şekilde kullanılabilir.

```
void glScalef(GLfloat x, GLfloat y, GLfloat z);
void glScaled(GLdouble x, GLdouble y, GLdouble z);
```

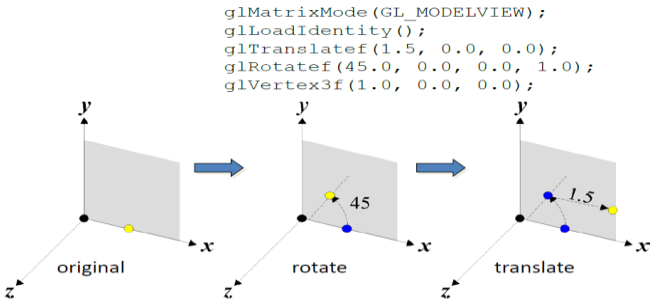
x, y, z parametrelerine geçirilen değerler her bir eksene göre ölçeklendirme değerini belirler. Örneğin, çizilen bir modelin derinlik ve yüksekliğini değiştirmeden, x ekseni üzerindeki genişliğini 2 katına çıkartılmak isteniyosa, aşağıdaki kod bu işlemi gerçekleştirir.

```
glScalef(2.0, 1.0, 1.0);
```

4.8.OpenGL Dönüşüm İşlem Önceliği

OpenGL'de dönüşümlerin uygulanma sırası dönüşüm fonksiyonlarının çağrılma sırası ile terstir. Yani çizim nesnesine yakın olan dönüşüm işlemi öncelikli olarak gerçekleştirilir.

Örneğin, aşağıdaki kod parçası çağrıldığında, öncelikle z ekseni üzerinde 45 derece döndürme işlemi yapılmış daha sonra ise x ekseni üzerinde 1.5 birimlik öteleme işlemi gerçekleştirilmiştir. (glVertex3f() çizim nesnesine en yakın dönüşüm en önce gerçekleştirilmiştir.)



Şekil 5. Dönüşüm Uygulanma Sırası

OpenGL'de farklı dönüşüm işlem sırası farklı sonuçlar üretir. Örneğin, aşağıdaki öteleme ve dönme işlemleri farklı sırada gerçekleştirilmiştir ve farklı sonuçlar üreteceklerdir.

```
// Example 1
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0, 0.0, -6.0);
glRotatef(45.0, 0.0, 1.0, 0.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
= Trans * Rot * Scale * v
```

```
// Example II
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(45.0, 0.0, 1.0, 0.0);
glTranslatef(0.0, 0.0, -6.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
= Rot * Trans * Scale * v
```

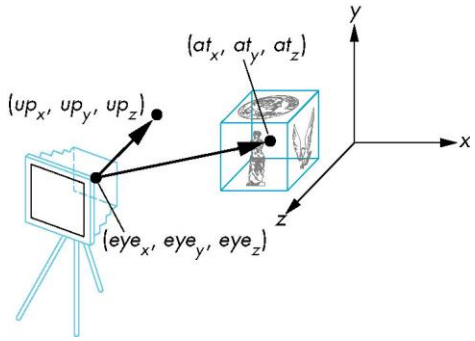
Şekil 6. Dönüşüm İşlem Sırası ve Farklı Çizim Sonuçları

4.9. OpenGL ve Kamera Görüntüsü

OpenGL kütüphanesi ile uzayda istenilen bir noktaya kamerayı koymak ve bu noktadan istenilen bir yöne istenilen açı ile bakmak mümkündür. Bu işlemin 3 öğesi bulunur:

1. Kameranın bulunduğu koordinatlar
2. Kameranın baktığı nokta
3. Kameranın bu eksen üzerindeki açısı

Bu durum kısaca aşağıdaki şekilde özetlenebilir.

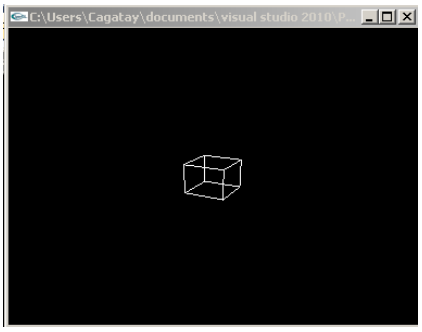


Şekil 7. OpenGL ile Kamera Konumu

Yukarıdaki şekilde de gösterildiği üzere kamera verilen eye_x, eye_y ve eye_z koordinatlarına yerleştirilmiş ve kameranın odak çizgisi verilen at_x, at_y ve at_z koordinatlarına yöneltilmiştir. Bu doğru üzerinde kamera istenildiği gibi döndürülebileceği için bu değeri belirlemek için kameranın bu eksenle yaptığı normal vektörü de up_x , up_y, up_z değerleri ile belirlenmiştir.

Aşağıdaki örnekte, bir küp çizdirilmiş ve küpe x=3, y=3 ve z=6 kordinatlarında bulunan bir kamaredan bakılmıştır. Ekran görüntüsü Şekil 4'deki gibidir:

```
#include <windows.h>
#include <GL/glut.h>

void init(void) {
 glClearColor(0.0, 0.0, 0.0, 0.0);
 glShadeModel(GL_FLAT);
}

void display(void) {
 glClear(GL_COLOR_BUFFER_BIT);
 glColor3f(1.0, 1.0, 1.0);
 glLoadIdentity();
   gluLookAt(3.0, 3.0, 6.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
 glutWireCube(1.0);
 glFlush();
}

void reshape(int w, int h) {
 glViewport(0, 0, (GLsizei) w, (GLsizei) h);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
 glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv) {
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
 glutInitWindowSize(500, 500);
 glutInitWindowPosition(100, 100);
 glutCreateWindow(argv[0]);
 init();
 glutDisplayFunc(display);
 glutReshapeFunc(reshape);
 glutMainLoop();

 return 0;
}
```



Şekil 8. OpenGL ile kamera görüntüsü

Tartışma Sorusu-4 : gluLookAt fonksiyonuna gönderilen parametreleri değiştirerek farklı açılarda çizim nesnesine bakınız. reshape fonksiyonun hangi nedenle kullanılmış olabileceğini tartışınız.

Tartışma Sorusu-5 : glTranslatef, glRotatef ve glScalef fonksiyonlarını işlevlerini

Tartışma Sorusu-6 : Daire, silindir, daire halkası gibi geometrik şekiller çizmek için kullanılabilecek yöntemleri tartışınız.

5. Deney Hazırlığı

Bu bölüm, deneye gelmeden önce her öğrenci tarafından yapılması gereken maddeleri içermektedir.

1. Deneye gelmeden önce C++ derleyicisi içeren herhangi bir sürümde Visual Studio IDe'si kurulmalıdır. *http://www.microsoft.com/visualstudio/tur/downloads#d-2010-expres* adresinden veya bölümümüzün DreamSpark Premium sayfasından indirebilirsiniz.
2. Ek-1'de verilen adımlar takip edilerek Microsoft Visual C++ 2010 Express veya diğerleri için (VS 2013 vs.) gerekli GLUT kütüphaneler eklenmelidir. (Farklı olarak VS 2013 için C:\Program Files (x86)\Microsoft Visual Studio 12.0)
3. Deneyde verilen uygulama kodları Ek-1'de anlatıldığı gibi ide üzerinde çalıştırılmalıdır.
4. Deney föyü dikkatlice okunmalı ve deneye hazırlık soruları cevaplanmalıdır. Deney uygulama yönergesinde gerekli açıklamalar bulunmaktadır.

6. Deney Tasarım ve Uygulaması

1. Deneye hazırlıksız gelen ve deney sırasında ilgisiz olan öğrenciler deneyden çıkarılacaktır. Deneye OpenGL uygulamalarının nasıl çalıştırılacağı, Visual Studio IDE'si ile kod düzenleme ve çalıştırmanın nasıl yapılacağını bilmeniz gerekmektedir. Deney ortamında VS 2013 kurulu olacaktır.
2. Deney sayfasında yer alan uygulamalar ve kaynak kodlar çalıştırılır ve incelenir. Deney sırasında uygulamalar üzerinde değişiklik yapar farklı grafik ve animasyon çıktıları elde etmeniz istenecektir.
3. OpenGL hakkında temel bilgiler soru-cevap şeklinde sorgulanır. Uygulama alanları ve güncel bazı OpenGL uygulamalarına örnekler verilir.
4. GLUT kütüphanesi nedir ve ne için kullanılır soruları cevaplanır, kod üzerinde uygulama yapmanız istenir.
5. Basit bir dörtgen şekli çizen program incelenir. Kullanılan fonksiyonların işlevleri soru-cevap şeklinde sorgulanır.
6. OpenGL ile şekil değiştirme işlemlerinin (taşıma, döndürme ve ölçeklendirme) nasıl yapıldığı, işlem önceliği ve farklı çizim sonuçları tartışılır. OpenGL 'de kullanılan koordinat uzayları ve dönüşümleri incelenir.
7. OpenGL ile kamera görüntüsü uygulaması incelenir ve gerekli tartışma soruları cevaplanır. Parametreler değiştirilerek farklı açılardan kamera görüntüsü incelenir. İlgili fonksiyonların işlevleri soru-cevap şeklinde sorgulanır.

7. Deney Soruları

1. OpenGL nedir? Ne için kullanılır? Kullanım avantajları nelerdir? OpenGL kullanılarak geliştirilen uygulamaları araştırınız.
2. OpenGL ES veya WebGL ile geliştirilmiş güncel bir kaç örnek uygulama araştırınız.
3. GLUT nedir? Ne için kullanılır?
4. 2 boyutlu dörtgen şekli çizen OpenGL komutlarını açıklayınız.
5. Taşıma, döndürme ve ölçeklendirme işlemlerinin koordinat sisteminde nasıl gerçekleştirildiğini kağıt üzerinde basitçe çizerek anlatınız.
6. Dönüşüm işlemleri farklı sırada çağrıldığında farklı çizim sonuçları üretir. Örnek veriniz ve çizerek anlatınız.
7. OpenGL ile kamera görüntüsü uygulamasında kamera görüntüsü almayı sağlayan kodları açıklayınız.

8. Deney Raporu

Deney rapor şablonu deney sayfasındadır. Gerekli açıklamalar ve sorular rapor kapağında verilmiştir. Raporda istenen dışında deneyle ilgili herhangi bir şey yazmayınız.

9. Kaynaklar

- An Interactive Introduction to OpenGL Programming - Dave Shreiner,Ed Angel, Vicki Shreiner
- Addison Wesley, "OpenGL Programming Guide", 6th Edition, 2008.
- http://www.opengl.org/
- http://www.khronos.org/
- http://www.opengl.org/wiki/Language_bindings
- http://www.opengl.org/documentation/implementations/#os
- http://www.bilgisayarkavramlari.com/
- http://www.lighthouse3d.com/opengl/glut/
- http://nehe.gamedev.net/