OPENGL

OpenGL Nedir?

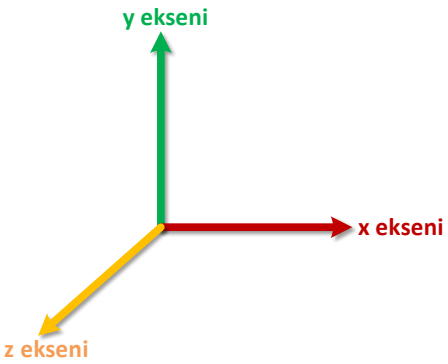
- İki veya üç boyutlu grafikleri ekrana çizmek için kullanılan grafik donanımı için geliştirilmiş bir arayüz programıdır. OpenGL ile karmaşık şekilleri komutlardan oluşturduğumuz noktalar, çizgiler ve çokgenler yardımıyla tanımlayabiliriz.

OpenGL İle Neler Yapılabilir?

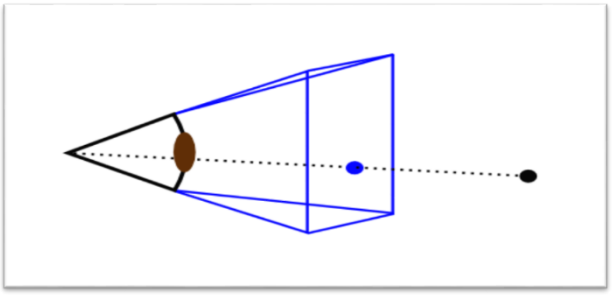
- OpenGL öncelikli olarak iki ve üç boyutlu çizimleri gerçekleştirebilmek ve bunların üzerinde animasyon özelliği uygulayabilmek amacıyla geliştirilmiştir. Bu görevleri gerçekleştirirken ayrıca şunları da destekler:
  - ⇒ Şekiller üçgen ve/veya çokgenler halinde oluşturulabilir,
  - ⇒ Şekillere derinlik verilerek atmosferik bir ortam görüntüsü yaratılabilir,
  - ⇒ Şekiller sivriltilerek yatay ve dikey çizgilerin görünürlüğü artırılarak boyut özelliği derinleştirilebilir,
  - ⇒ Işık özelliği kullanılmadan düz-gölge efekti ile renkler keskin çizgiler halinde belirginleştirilebilir,
  - ⇒ Işıklandırma ile daha gerçekçi ve üç-boyutlu belirgin çizimler yapılabilir,
  - ⇒ Şekiller üzerinde doku oluşturma (çizgiler veya resim, örneğin duvar kağıdı görünümü) yapılabilir,
  - ⇒ Bulanıklık verilerek hareketler simüle edilebilir,
  - ⇒ Görüş alanı değiştirilerek penceredeki farklı bir alan görüntülenebilir (uzaklaşma-yakınlaşma),
  - ⇒ Duman etkisi verilerek ortam efekti oluşturulabilir,
  - ⇒ Odaklanma efekti verilerek, odaklanılan cismin daha belirgin görünmesi sağlanabilir,
  - ⇒ Renkler arasında geçiş, renk akışı sağlanabilir,
  - ⇒ Işıklandırma açısı ve yönü her cisim için farklı ayarlanabilir,
  - ⇒ Hareket özelliği kazandırılan cisimler oluşturulabilir,
  - ⇒ Zamanlamaya bağlı hareketler oluşturulabilir,
  - ⇒ Hareketler kullanıcıya ve donanıma bağlı sağlanabilir(Mouse-klavye işlevleri),
  - ⇒ Ve daha sayamadığımız birçok özellikle birçok uygulama geliştirilebilir.

OpenGL Nasıl Çalışır?

- Basitçe üç boyutlu koordinat sistemini kullanarak çalışır. X ve Y koordinatları olan iki boyutlu ekrandan bize doğru uzanarak gelen birde Z ekseni mevcut gibi dşünüldüğünde zihnimizde canlanması gereken şekil yandaki gibidir.

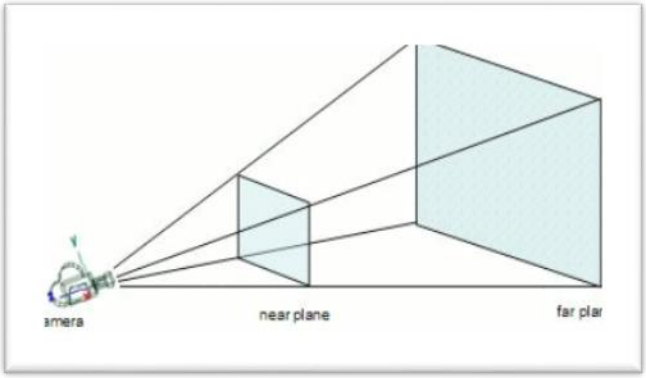


- Ekranımızda görünenler izdüşüm dönüşümü yapılmış iki boyutlu resimlerdir. Bakış açımızı taklit eden OpenGL böylece üç boyutluluk izlenimi verir.



- İzdüşüm, yani projeksiyon dönüşümü şekildeki gibidir:

Yakın düzlemi ve uzak düzlemi, OpenGL'in sunduğu görme alanının sınırlarını belirler. Yakın düzleminden önceki alan çok yakın varsayıldığı, uzak düzleminden sonraki alan da çok uzak varsayıldığı için görünmez.



GLUT Nedir?

- GLUT ile işletim sistemi penceresi değil platformdan bağımsız kendi penceremiz oluşturulabildiği bir OpenGL kütüphanesidir. Böylece her işletim sisteminde derlenerek çalıştırılabilir.

GLUT'u Visual Studio 2017'de Ayarlama:

- GLUT indirilir
- **glut32.dll** dosyasını [c://windows/System32/] klasörünün içine, eğer 64 bit kullanılıyorsa ayrıca [c://windows/sysWOW64/] klasörüne atılır.
- **glut32.lib** dosyası kullanılmakta olan derleyicinin lib ([C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\VC\Auxiliary\VS\lib]) klasörüne atılır.
- **glut.h**, dosyası derleyicinin başlık dosyalarının olduğu klasöre ([C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\VC\Auxiliary\VS\include]) ya da projenin dosyalarının olduğu yere atılır.

OpenGL ile Temel Çizim Fonksiyonları:

- **glutInit (&argc, argv):** glut fonksiyonlarını başlatmak için kullanılır.
- **glutInitDisplayMode():** GLUT penceresinin nasıl bir renk kipine, tek veya çift arabellekli olup olmayacağınaveya pencereye derinlik sağlanıp sağlanmayacağı belirlenir.

Bu fonksiyonun aldığı parametrelerden bazıları:

GLUT_RGB : Kırmızı, yeşil, mavi renk modunda pencere açar.

GLUT_RGBA : Kırmızı, yeşil, mavi ve alfa(geçirgenlik) renk modunda pencere açar.

GLUT_INDEX : Renk sıralama kipini seçer.

GLUT_SINGLE : Tek ara bellekli bir pencere oluşturur.

GLUT_DOUBLE : Çift ara bellekli bir pencere oluşturur.

GLUT_DEPTH : Derinlik ara bellekli bir pencere oluşturur.

NOT: Parametre alan fonksiyonlar (örneğin glutInitDisplayMode) birden fazla parametreyi aralarına | işareti konularak alabilirler.

```
glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE)
```

Yukarıdaki komut ile RGB renk modunu kullanan ve çift ara bellekli bir pencere oluşturulması istenir.

NOT: Tek arabellekte çizim komutlarının işlevi hemen uygulanır. Arkasından başka bir komut geldiğinde önceki silinir ve yerine yeni olanın işlevi görüntülenir. Bu durum görüntü oluşturmada ekran titremelerine sebep olabilir. Bu yüzden genellikle çift arabellek kullanılır. Böylece titreşim engellenir. Çizimler ayrı bir çizim belleğinde gerçekleştirilir tamamlandığında ekrana bastırılır.

- **glutInitWindowSize(int weight, int height):** Bu fonksiyon ile pencere boyutları belirlenir.
- **glutInitWindowPosition(int x, int y):** Penceremizin konumlanacağı sol üst köşenin ekran koordinatını belirtir.
- **glutInitCreateWindow("Örnek"):** "Örnek" başlıklı bir pencere oluşturulur.
- **glutGlutDisplay(ciz):** Görüntüleme yapmak için 'ciz' isimli bir pencere oluşturulur.
- **glutReshapeFunc(yinele):** Çizidirilen pencere yeniden boyutlandırılırsa bu fonksiyonun geri bildirim fonksiyonu olan 'yinele' çağrılır. Geri bildirim fonksiyonu parametre olarak genişlik ve yükseklik değerlerini alır.
- **glutIdleFunc(idle):** Hiçbir olay oluşturulmadığı durumda ilgili geri bildirim fonksiyonunu (idle) çağırır.
- **glutMainLoop():** OpenGL penceresi oluşturmak için main() bloğu içerisinde yazdığımız son komuttur. OpenGL penceresinin olayları sürekli olarak döndürmesini, yani görüntünün sürekli tazelenmesini sağlayan komuttur.
- OpenGL çizim komutları **glBegin()** ile başlar ve **glEnd()** ile biter. **glBegin()** aldığı çizim parametresine ve kendinden önce aktif edilen komutlara başlı olarak çizimin başlayacağını, **glEnd()** ise ilgili çizimin bittiğini belirtir.

GL_POINTS : Tanımlanan her bir vertexi tek bir nokta olarak alır.

GL_LINES : Arka arkaya tanımlanan her vertex çiftini bir doğru kabul eder.

GL_TRIANGLES : Tanımlanan üç vertex ile bir üçgen oluşturur.

GL_QUADS : Tanımlanan dört vertex ile bir dörtgen oluşturur..

GL_POLYGON : Tanımlanan tüm vertexleri kullanarak bir çokgen oluşturur

```
glBegin(GL_QUADS);
glVertex3f(-0.5, 0.0, -0.5);
glVertex3f(-0.5, 0.0, 0.5);
glVertex3f(0.5, 0.0, -0.5);
glVertex3f(0.5, 0.0, 0.5);
glEnd();
```

Çizimler **glutDisplay** komutu tarafından gönderilen geri bildirim fonksiyonunun içerisinde gerçekleştirilir.

- **glFlush(): glBegin()**ve **glEnd()** blokları arasında belirtilen ve çizdirilmek üzere saklanan çizim özelliklerini ekrana yazdırır.

➡ **glutSwapBuffers:** Çizimin çift tamponlama (GLUT_DOUBLE) seçilerek, yani arka planda şeklin tamamı çizildikten sonra ön plana aktarılarak yapılması durumda **glBegin()**ve **glEnd()** blokları arasında belirtilen çizim özelliklerini ekrana yazdırır.

➡ **glClear:** Tamponların içeriğini belirlenen değerlerle temizler.

```
glClear(GL_COLOR_BUFFER_BIT);

glClearColor(0.0, 0.0, 0.0, 0.0);
```

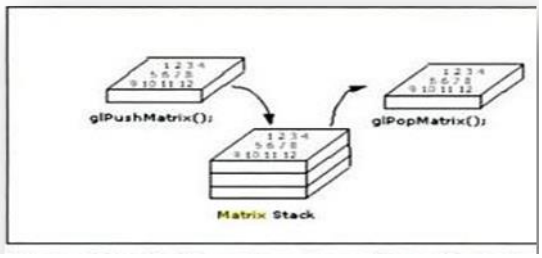
Renk tamponunun içeriğini temizler ve başlangıç değerini siyah olarak ayarlar

➡ OpenGL'de mevcut matris yığını **glPushMatrix()** ve **glPopMatrix()** komutları arasında gerçekleştirilir. Yığın yapısı kullanan OpenGL'de üzerinde işlem (döndürme, öteleme, vs.) yapılacak olan çizimi alır ve yığının en üstüne taşır, daha sonra da ilgili matrisi hafızaya yükler.

```
glPushMatrix();
glScalef(0.5, 1, 1);
glTranslatef(0.5, 0.5, 0.0);

glutWireCube(0.8);

glPopMatrix();
```



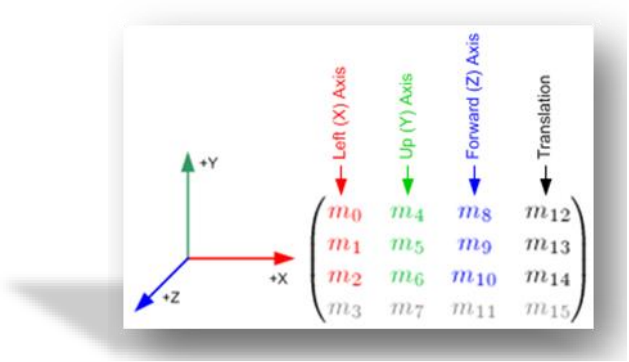
➡ **glTranslatef(x, y, z):** Mevcut matrisin öteleme matrisi ile çarpımıdır. *x*, *y*, *z* parametreleri öteleme vektörünün koordinatlarıdır.

➡ **glRotatef(angle, x, y, z):** Mevcut matrisin döndürme matrisi ile çarpımıdır. *angle* döndürme açısı, *x*, *y*, *z* parametreleri öteleme vektörünün koordinatlarıdır. Dönme yönü sağ el kuralına göre belirlenir.

➡ **glScalef(x, y, z):** Mevcut matris ile ölçekleme matrisinin çarpımıdır. *x*, *y*, *z* parametreleri sırasıyla eksenlerin ölçekleme oranlarıdır.

➡ **glLoadIdentitiy():** Herhangi bir anca OpenGL yığınında bulunan matrisin temizlenmesi, yani birim matrisi dönüştürülmesi için kullanılır.

➡ **glMatrixMode ():** OpenGL'in kullandığı dönüşüm işlemleri için kullandığı 4x4'lük matrislerden (GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE, GL_COLOR) hangisinin kullanılacağını belirten komuttur.

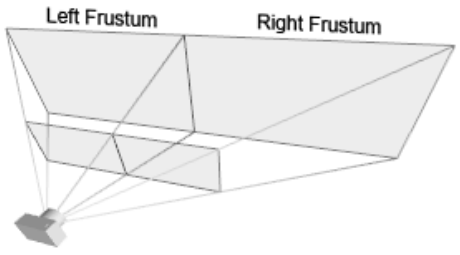



```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

gluPerspective(60.0, (float)w / (float)h, 1.5, 20.0);
```

➡ **gluPerspective(fovY, aspect, zNear, zFar):** Perspektif görüş alanımızı belirler. Görüş alanımızın y eksenine olan açısını *fovY*, pencere boyutunun genişlik/yükseklik oranını *aspect*, yakın ve uzak görüş mesafeleri de *zNear* ve *zFar* bileşenleri ile belirlenir.

➡ **glFrustum(-fW, fW, -fH, fH, zNear, zFar):** Perspektif görüş alanımızı belirlemek için kullanılabilecek OpenGL komutudur. Burada ekran boyutunun sol ve sağ bileşenleri $fH = \tan\left(fovY \middle/ 360 * pi \right) * zNear$, üst ve alt bileşenleri $fW = fH * aspect$, yakın ve uzak görüş mesafeleri de *zNear* ve *zFar* bileşenleri ile belirlenir. Perspektif görüş alanı belirlerken, özellikle de alanımızı iki farklı görüş alanına bölmek istersek bu komutu kullanırız.



➡ **glOrtho(left, right, bottom, top, zNear, zFar):** Paralel görüş alanımızı belirlemek için kullanılabilecek OpenGL komutudur. Burada ekran boyutunun sol ve sağ bileşenleri *left* ve *right*, üst ve alt bileşenleri *bottom* ve *top*, , yakın ve uzak görüş mesafeleri de *zNear* ve *zFar* bileşenleri ile belirlenir.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

gluLookAt(0.0, 0.0, 1.0,
    0.0, 0.0, 0.0,
    0.0, 1.0, 0.0);
```

- **gluLookAt(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ):** Kameranın bulunduğu noktadaki görüş matrisini oluşturur. Parametrelerden eyeX, eyeY, eyeZ gözün bulunduğu koordinatları, centerX, centerY, centerZ kameranın bulunduğu referans noktasınının koordinatları, upX, upY, upZ ise kameranın duruşuna bağlı yukarı vektörünün yönünü gösterir.

- **glViewPort(x, y, width, height):** Gerçekleştirilecek olan çizimin görüş alanını belirler. *x*, *y* görüş alanı dikdörtgeninin sol alt köşesi, *width* ve *height* ise genişlik ve yükseklik parametreleridir.

Boş OpenGL Penceresi Oluşturmak:

- Aşağıda örnek boş bir OpenGL penceresi oluşturmak için gerekli kodlar verilmiştir.

```
#include <iostream>
#include <glut.h>

using namespace std;

void render()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
}
void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(500, 500);
    glutCreateWindow("İlk OpenGL Penceremiz...");
    glutDisplayFunc(render);
    glutMainLoop();
}
```