

Ders Konu Kapsamı

4. OpenGL Kütüphanesi

4.1. Pipeline Adımları.

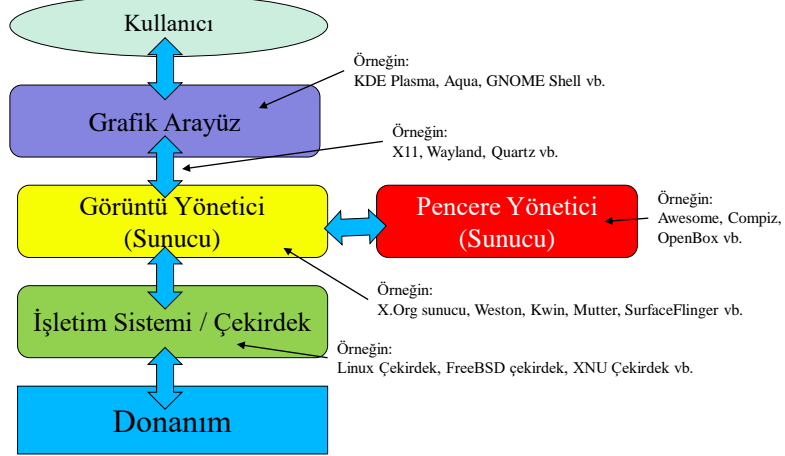
4.2. Çizme Fonksiyonlarının Kullanımı ve Örnek Uygulama.

Amaç

- OpenGL API'sinin geliştirilmesi
- OpenGL Mimarisi
 - Durum makinesi olarak OpenGL
 - Veri akışı makinesi olarak OpenGL
- Fonksiyonlar
 - Türleri
 - Biçimleri
- Basit bir program

Ön Bilgi

GRAFİK SİSTEMİ

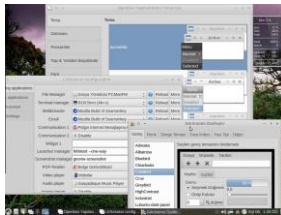


Dr. Ömer ÇETİN

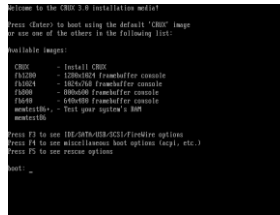
5

Ön Bilgi

- Bir yazılım ile kullanıcı arası iletişimi görsel olarak sağlamak için üretilmiş olan görsel öğeler bütününe **grafiksel kullanıcı arayüzü (Graphical User Interface - GUI)** denir.
- İşletim sistemlerine ait grafiksel arayüzde birbirine sıklıkla karışan çeşitli kavramları örneklerle açıklayalım:



Grafiksel Arayüz



Metin tabanlı kullanıcı arabirimi

Dr. Ömer ÇETİN

6

Ön Bilgi

GRAFİK KARTI SÜRÜCÜLERİ

- En yaygınları Intel, ATI, NVIDIA gibi üreticiler olan grafik kartı donanımlarını işletim sisteminin tanıyıp kullanabilmesi için gerekli olan yazılımdır.
- NVIDIA ve ATI üreticileri için **açık veya kapalı kaynak kodlu sürücü** seçenekleri bulunur.
 - Açık kaynak sürücüler yalnızca 2D desteği ile gelir. Ancak «**Mesa kütüphanesi**» yüklenerek 3D desteği de kazanılabilir.
- Grafik kartı sürücüsü konusunda karşılaşılabilecek diğer bir kavram da «**vesa**»dır.
 - Vesa 2D veya 3D desteği olmayan en temel grafik kartı sürücüsüdür. Dünya çapında standart olduğu için bütün grafik kartları tarafından desteklenir.

www.mesa3d.org

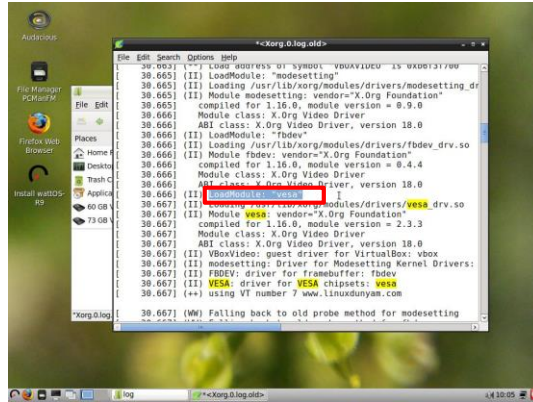
www.x.org/releases/current/doc/man/man4/vesa.4.xhtml

Dr. Ömer ÇETİN

7

Ön Bilgi

GRAFİK KARTI SÜRÜCÜLERİ



```
30.663] Load address 0: symbol: VESAVIDEO is undefined
30.665] [III] LoadModule: "modesetting"
30.665] [III] Loading /usr/lib/xorg/modules/drivers/modesetting_dr
30.665] [III] Module modesetting: vendor="X.Org Foundation"
30.665] compiled for 1.16.0, module version = 9.9.0
30.666] Module class: X.Org Video Driver
30.666] ABI class: X.Org Video Driver, version 18.0
30.666] [III] LoadModule: "fbdev"
30.666] [III] Loading /usr/lib/xorg/modules/drivers/fbdev_drv.so
30.666] [III] Module fbdev: vendor="X.Org Foundation"
30.666] compiled for 1.16.0, module version = 9.4.4
30.666] Module class: X.Org Video Driver
30.666] ABI class: X.Org Video Driver, version 18.0
30.666] [III] LoadModule: "vesa"
30.667] [III] Loading /usr/lib/xorg/modules/drivers/vesa_drv.so
30.667] [III] Module vesa: vendor="X.Org Foundation"
30.667] compiled for 1.16.0, module version = 2.3.3
30.667] Module class: X.Org Video Driver
30.667] ABI class: X.Org Video Driver, version 18.0
30.667] [III] VBoxVideo: guest driver for VirtualBox: vbox
30.667] [III] modesetting: Driver for Modesetting Kernel Drivers:
30.667] [III] FBDEV: driver for framebuffer: fbdev
30.667] [III] VESA: driver for VESA chipsets: vesa
30.667] [++] using VT number 7 www.linuxdynes.com
30.667] [W] Falling back to old probe method for modesetting
```

Dr. Ömer ÇETİN

8

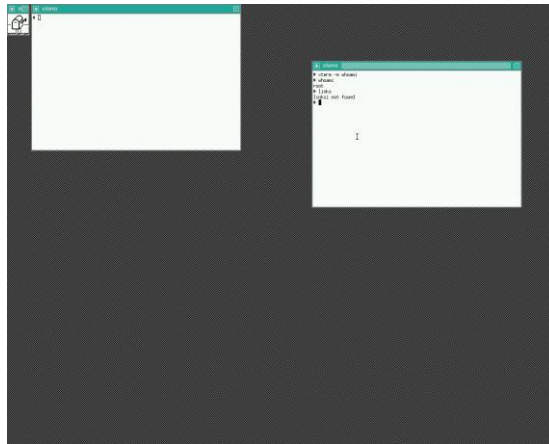
Ön Bilgi

Görüntü Yönetici (Sunucu)

- Örneğin bir görüntü yöneticisi olan Xorg yazılımı, bilgisayarınızın donanımının, çalıştırdığınız grafiksel arabirime sahip herhangi bir programa ait grafikleri size sunabilmesini sağlayacak olan ortamı oluşturabilmesini sağlar. Ayrıca klavye ve fare kullanımını da yönetir.
- Sunucu, istemci, kurallar ve kitaplıklar şeklinde ana bölümlere ayrılır. Sunucu size grafik arabirimini sunar. Uygulamaların istekleri Xorg'un istemcisi tarafından alınır. `xlsclients` komutu ile istekte bulunan uygulamalarınızı listeleyebilirsiniz. Kurallar istemci-sunucu iletişimini sağlar, kitaplıklar da gerekli olan bütün altyapıyı içerir.

Ön Bilgi

Xorg'un sunduğu temel X ortamı:



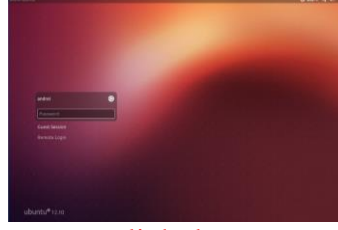
Ön Bilgi

GÖRÜNTÜ YÖNETİCİ (Display Manager)

- Size görsel arayüz sunacak olan grafik sunucusunun başlatılması, devam edebilmesi, kullanıcı yetkileri ve oturum açma gibi ayarları yapan küçük programlardır. Örneğin xdm, gdm, lightdm, mdm:



xdm



lightdm

Ön Bilgi

GÖRÜNTÜ YÖNETİCİ (Display Manager)

- MS Windows işletim sisteminde kullanılan görüntü yöneticisi uygulamasının adı Desktop Windows Manager (DWM) olarak adlandırılır.

Name	3% CPU	43% Memory	0% Disk	0% Network
Client Server Runtime Process	0%	1.1 MB	0 MB/s	0 Mbps
Console Window Host	0%	0.7 MB	0 MB/s	0 Mbps
Console Window Host	0%	0.8 MB	0 MB/s	0 Mbps
Desktop Window Manager	0.9%	50.1 MB	0 MB/s	0 Mbps

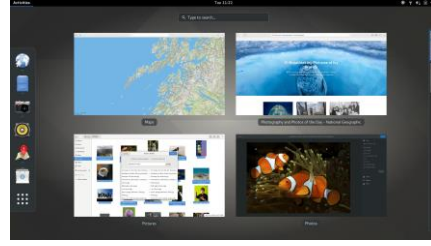
Ön Bilgi

MASAÜSTÜ ORTAMI (Desktop Manager)

- Kullandığınız işletim sisteminde size grafiksel arayüz olarak sunulan bütün resim, öge ve pencerelerin ortaklaşa kullandığı ortama denir. GNOME, KDE, Xfce, LXDE, Enlightenment birer masaüstü ortamıdır. KDE ve GNOME:



KDE

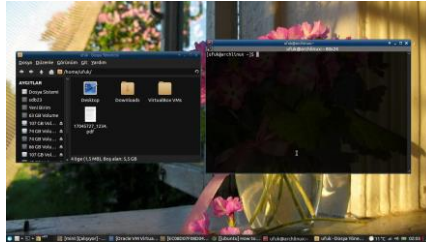


GNOME

Ön Bilgi

PENCERE YÖNETİCİSİ

- Masaüstü ortamında görüntülenen pencerelerin biçim, hareket ve yönetimini sağlayan uygulamadır. Pencere yöneticisi için masaüstü ortamı değişken olabilir. Örneğin Xfce ile ister xfwm4'ü ister marco'yu isterseniz metacity'i kullanabilirsiniz.



API Tarihçesi

- International Federation for Information Processing (IFIPS-1973), **standart grafik API**'sı oluşturma için iki komite kurdu:
 - **Grafik Çekirdek Sistemi (GKS)**
 - 2B iş istasyonu modeli içeriyordu
 - **Çekirdek**
 - 2D ve 3D
 - **GKS, ISO** ve daha sonra **ANSI** standardı olarak kabul edildi (1980'ler).
- GKS kolayca 3D'ye uyarlanamadı (**GKS-3D**)
 - Donanım geliştirmenin çok gerisinde kaldı

PHIGS ve X

- Programmers Hierarchical Graphics System (PHIGS)
 - CAD topluluğundan ortaya çıktı
 - Korunmuş grafikleri olan **veritabanı modeli** (yapılar)
- X Pencere Sistemi
 - DEC / MIT çabası
 - Grafikler ile **istemci-sunucu** mimarisi
- PEX ikisini birleştirdi
 - Kullanımı kolay değil (her birinin kusurları)

SGI ve GL

- Silicon Graphics (SGI), **boru hattını** donanıma uygulayarak grafik iş istasyonunda devrim yarattı (1982).
- Sisteme erişmek için uygulama programcıları **Graphic Language (GL) adlı bir kütüphane** kullandılar.
- GL ile **üç boyutlu** etkileşimli uygulamaları programlamak nispeten kolaydı.

OpenGL

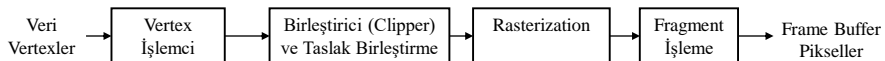
- GL'nin başarısı, **platformdan bağımsız** bir API olan **OpenGL**'ye geçiş sağladı (1992)
 - Kullanımı kolay
 - Mükemmel performans elde etmek için donanıma yeterince yakın
 - Oluşturmaya odaklanma sağlayan
 - Pencere sistemi bağımlılıklarını önleme

OpenGL Gelişimi

- Aslen bir Architectural Review Board (ARB) tarafından kontrol edilir.
 - Üyeleri SGI, Microsoft, Nvidia, HP, 3DLabs, IBM,
 - Şimdi adı: **Kronos Grubu**
 - Giderek kararlıdır (sürüm >2.5 ile)
 - Geriye uyumlu geliştirme
 - Yeni donanım yeteneklerini yansıtma
 - 3B doku eşleme ve doku nesneleri üretebilme
 - Vertex ve fragment programlarını gerçekleştirme
 - Shader programlama dili
 - Uzantılar üzerinden platforma özgü özelliklere izin verir...

Modern OpenGL

- CPU yerine **GPU** kullanılarak performans elde edilir.
- **Shaders** adı verilen programlar aracılığıyla GPU'yu denetleyin.
 - Varsayılan shader yok... (Özelleştirilebilir boru hattı sağlar)
 - Her uygulama hem bir vertex hem de bir fragment shader sağlamalıdır
- Uygulamanın işi GPU'ya **veri** göndermek.
- GPU tüm oluşturma (**rendering**) işlemlerini yapar.



Diğer Versiyonlar

OpenGL ES

Gömülü Sistemler

Version 1.0 basitleştirilmiş OpenGL 2.1

Version 2.0 basitleştirilmiş OpenGL 3.1

Shader tabanlı

WebGL

Javascript ES 2.0 uyarlaması

Yeni nesil tarayıcıların tamamı tarafından desteklendi

OpenGL 4.X ve son sürüm 4.6

Direct X

- **Avantajları**

- Kaynakların daha iyi kontrolü
- Üst düzey işlemlere erişim

- **Dezavantajları**

- Yeni sürümler geriye uyumlu değil
- Sadece Windows platformu desteklenir
- Shader yapısı ile ilgili son gelişmeler OpenGL ile yakınlaşmaya yol açmaktadır ...

OpenGL Kütüphaneleri

- **OpenGL çekirdek kütüphanesi**
 - Windows'ta OpenGL32
 - Çoğu unix / linux sistemlerinde GL (libGL.a)
- **OpenGL Hizmet Programı Kütüphanesi (GLU)**
 - OpenGL çekirdeğinde işlevsellik sağlar, ancak kodu yeniden yazmak zorunda kalmaz
 - Sadece eski kodla çalışacak
- **Pencere sistemli bağlantılar**
 - X pencere sistemleri için GLX
 - Windows için WGL
 - Macintosh için AGL

GLUT

OpenGL Yardımcı Araç Takımı (GLUT)

- Tüm pencere sistemlerinde ortak işlevsellik sağlar
- Bir pencere aç
- Harici aygıtlardan (Fare ve klavyeden vb.) giriş al
- Menüler oluştur
- Olay güdümlü çalışma sağla
- Kod taşınabilir ancak GLUT belirli bir platform için iyi bir araç setinin işlevselliğinden yoksundur...
- Örneğin kaydırma çubuğu yoktur.

freeglut

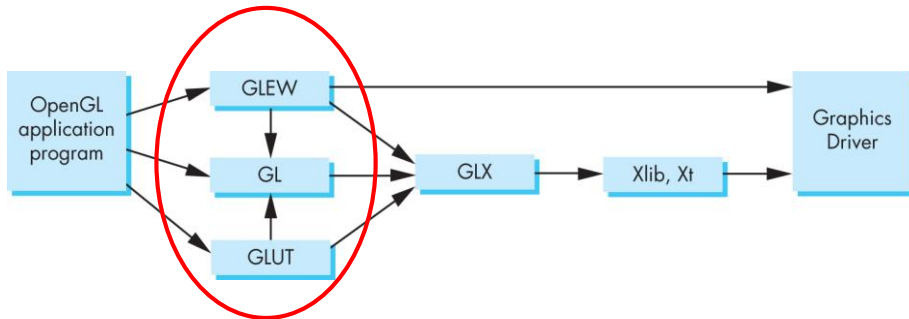
- GLUT uzun zaman önce geliştirildi ve değişmedi:
 - OpenGL 3.1 ile çalışması şaşırtıcı...
 - Kullanımdan kaldırılmış işlevler gerektirdiğinden bazı işlevleri çalışmıyor.
- **freeglut** GLUT ürününü güncelledi
 - Eklenen yetenekler geldi.
 - Bağlam kontrolü geliştirildi.

GLEW

OpenGL Eklenti Wrangler Kütüphanesi

- Belirli bir sistemde bulunan OpenGL uzantılarına erişmeyi kolaylaştırır.
- Windows kodunda belirli giriş noktalarına sahip olmaktan kaçınır.
- Uygulama sadece **glew.h**'yi içermeli ve bir **glewInit ()** çalıştırmalıdır.

Yazılım Organizasyonu



Ödev-1 OpenGL Kurulumu

Dr. Ömer ÇETİN

27

OpenGL Fonksiyonları

- İlkeller (Primitives)
 - Noktalar
 - Doğru parçaları
 - Üçgenler
- Öznitelikler
- Dönüşümler
 - İzlenimi
 - Modelleme
- Denetim (GLUT)
- Giriş (GLUT)
- Sorgu

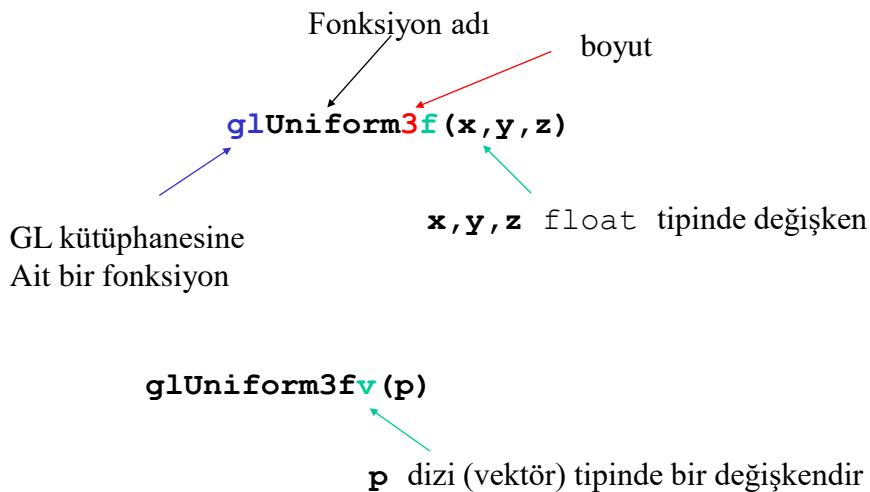
Dr. Ömer ÇETİN

28

OpenGL Durum Makinesi

- OpenGL bir durum makinesidir.
- OpenGL fonksiyonları iki tiptir:
 - **İlkel üretim fonksiyonları**
 - İlkel görünür durumdaysa çıkışa neden olabilir
 - Köşeler nasıl işlenir ve ilkel görünüm durum makinası tarafından kontrol edilir
 - **Durum değiştirme fonksiyonları**
 - Dönüşüm fonksiyonları
 - Özellik fonksiyonları
 - Versiyon 3.1 altında çoğu durum değişkeni uygulama tarafından tanımlanır ve shader'lara gönderilir

OpenGL fonksiyon formatı



OpenGL #defines

- Çoğu sabit include dosyaları içeriğinde yer almaktadır:
- **gl.h**, **glu.h** ve **glut.h**
 - `#include <GL/glut.h>` diğerlerini otomatik olarak içermelidir.
 - Örneğin;
 - `glEnable(GL_DEPTH_TEST)`
 - `glClear(GL_COLOR_BUFFER_BIT)`
- include dosyaları ayrıca OpenGL veri tiplerini de içerir:
GLfloat, **GLdouble**,....

GLSL

OpenGL Shading Dili (GLSL)

- C benzeri
 - Matris ve vektör çeşitleri (2, 3, 4 boyutlu)
 - Tanımlanmış (overloaded) operatörler
 - C ++ benzeri yapılar
- Nvidia'nın Cg ve Microsoft HLSL'ine benzerdir.
- Shader'lara kaynak kodu olarak gönderilen koddur.
- OpenGL fonksiyonları, shader'lar ile derleme, bağlantı kurma ve bilgi alma işlevlerini yerine getirir.

OpenGL ve GLSL

- Shader tabanlı OpenGL, veri akışı modelinden ziyade **durum makinesi** modeline dayanır...
- Çoğu durum değişkeni, öznitelik ve versiyon 3.1 öncesi OpenGL işlevleri kullanım dışı kalmıştır...
- Eylemler shader içinde gerçekleşir...
- İş, uygulama GPU'ya veri almaktır...

Örnek OpenGL Uygulaması

- İlk adım **programlama dilini** seçmek. OpenGL için **binding** işlemi, C# ve Java'dan Python ve Lua'ya kadar birçok dilde mevcuttur. Bazı dillerde, hiçbir resmi olmayan birden fazla OpenGL bağlama grubu vardır. Sonuçta hepsi C veya C ++ bağlarına dayanır.
- C / C ++ kullanmıyorsanız, seçtiğiniz dil için OpenGL bağlayıcılarını içeren bir paket veya kütüphane indirmeli ve yüklemelisiniz. Bazıları önceden yüklenmiş olarak gelir, ancak bazılarında ayrı indirmeler vardır.
- C / C ++ kullanıyorsanız, önce OpenGL'e bağlanabilecek bir derleme ortamı (Visual Studio projesi, GNU makefile, CMake dosyası, vb.) oluşturmalsınız. Windows altında, **OpenGL32.lib** adlı bir kitaplığa statik olarak bağlanmanız gerekir.
- Linux'ta, libGL'ye bağlanmanız gerekir. Bu, "-lGL" komut satırı parametresiyle yapılır.

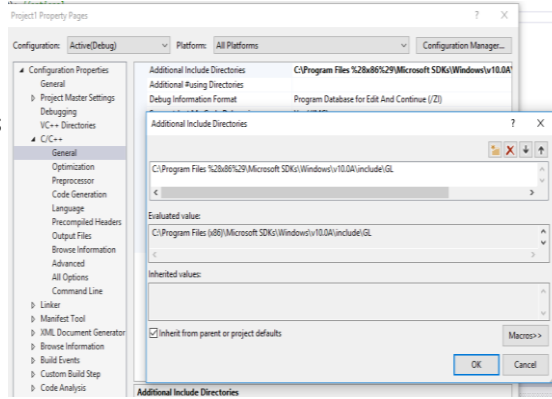
Örnek OpenGL Uygulaması

```
#include "glew.h"
#include "freeglut.h"
#include <iostream>

void display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glPointSize(5.0f);

    glBegin(GL_QUADS);
        glVertex2i(10, 10);
        glVertex2i(10, 210);
        glVertex2i(210, 210);
        glVertex2i(210, 10);
    glEnd();

    glFlush();
}
```

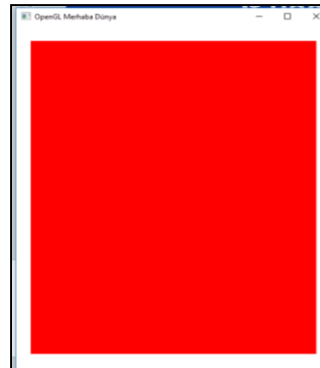


Örnek OpenGL Uygulaması

```
int main(int argc, char *argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
    glutInitWindowPosition(500, 200);
    glutInitWindowSize(500, 600);
    glutCreateWindow("OpenGL Merhaba Dünya");

    glClearColor(1.0, 1.0, 1.0, 1.0);
    gluOrtho2D(0.0, 220.0, 0.0, 220.0);

    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```



GLUT Fonksiyonları

```
glutInit(&argc, argv);
```

- Bu çağrı **GLUT'u başlatır**. GLUT kütüphanesi kullanılmaya başlanmadan önce mutlaka bu fonksiyon çağrılmalıdır. Parametreler doğrudan komut satırından sağlanabilir ve X'in eş zamansız yapısını devre dışı bırakan ve otomatik olarak GL hatalarını denetleyen ve bunları görüntüleyen (sırasıyla) '-sync' ve '-gldebug' gibi kullanışlı seçenekler içerir.

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
```

- Burada bazı GLUT seçeneklerini yapılandırıyoruz. **Renk paleti ve derinliği gibi ayarlamalar yapılır**. Örneğin, **GLUT_DOUBLE**, çift buffer ile çalışmayı (başka bir buffer gösterilirken arka plan bufferına çizim) ve çoğu işlemin sona erdiği (yani ekran) renk tamponunu etkinleştirir.

<https://www.opengl.org/resources/libraries/glut/spec3/node12.html>

GLUT Fonksiyonları

```
glutInitWindowPosition(500, 200);
```

```
glutInitWindowSize(500, 600);
```

```
glutCreateWindow("OpenGL Merhaba Dünya");
```

- Bu çağrılar **pencere parametrelerini** belirler ve yaratır. Ayrıca pencere başlığını belirleme seçeneğiniz de vardır.



GLUT Fonksiyonları

```
glutDisplayFunc(fonksiyon_adı);
```

- Pencereleme sisteminde çalıştığımız için, çalışan programla etkileşimin çoğu olay geri çağırma (callback) işlevleriyle gerçekleşir. GLUT, temeldeki pencereleme sistemi ile etkileşime girmeye özen gösterir ve bize birkaç geri çağırma seçeneği sunar. **Burada tek bir çerçevenin tüm işlemlerini yapmak için sadece bir "ana" geri çağrı kullanıyoruz. Bu fonksiyon sürekli olarak GLUT iç döngüsü tarafından çağrılır.**
- Oluşturulan pencere üzerinde **çizim işlemlerini yapan fonksiyonu parametre olarak alır** ve pencere içeriği yeniden çizildiğinde veya oluşturulduğunda bu fonksiyonu çalıştırır.

Örnek OpenGL Uygulaması

```
glClearColor(1.0, 1.0, 1.0, 1.0);
```

- Yukarıdaki çağrı, **çerçeve içeriğini temizlerken kullanılacak rengi** ayarlar. Renk dört kanala (RGBA) sahiptir ve 0,0 ile 1,0 arasında normalize edilmiş bir değer olarak belirtilir.

GLUT Fonksiyonları

```
glutMainLoop ();
```

- Bu çağrı ile kontrol GLUT'a geçer ve **kendi iç döngüsüne başlar**. Bu döngüde, pencereleme sistemindeki olayları dinler ve yapılandırdığımız geri çağrılar yoluyla bunları iletir.

```
glClearColor (GL_COLOR_BUFFER_BIT);
```

```
glutSwapBuffers ();
```

- Render fonksiyonumuzda yaptığımız tek şey çerçeveyi temizlemek (yukarıda belirtilen rengi kullanarak - değiştirmeyi deneyin). İkinci çağrı GLUT'a arka tamponun ve ön tamponun rollerini değiştirmesini söyler. Oluşturma geri çağırısı boyunca bir sonraki turda mevcut karelerin ön tamponuna işleyeceğiz ve mevcut geri arabellek görüntülenecektir.

Örnek OpenGL Uygulaması

```
gluOrtho2D(0.0, 220.0, 0.0, 220.0);
```

void gluOrtho2D (Gldouble sol, Gldouble sag, Gdouble alt, GLdouble üst);

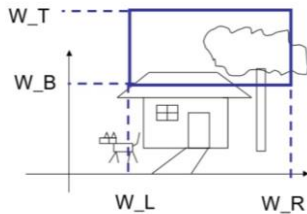
- **Parametreler**

Sol ve sağ dikey kırpma düzlemleri için koordinatları belirtin.

Alt ve üst yatay kırpma düzlemleri için koordinatları belirtin.

- **Açıklama**

gluOrtho2D iki boyutlu bir ortografik görüntüleme bölgesi oluşturur.



```
gluOrtho2D(W_L, W_R, W_B, W_T);
```

Örnek OpenGL Uygulaması

```
void display(void) {  
    glClearColor(GL_COLOR_BUFFER_BIT);  
    glColor3f(1.0, 0.0, 0.0);  
    glPointSize(5.0f);  
  
    glBegin(GL_QUADS);  
        glVertex2i(10, 10);  
        glVertex2i(10, 210);  
        glVertex2i(210, 210);  
        glVertex2i(210, 10);  
    glEnd();  
  
    glFlush();  
}
```

Örnek OpenGL Uygulaması

```
void glBegin(enum kip);
```

Bir çizime başlandığını belirtir. Parametresi çizilen geometrik şekli tanımlar. Bu parametre yerine geçilmesi gereken sembolik sabitler şunlardır :

- **GL_POINTS** : Nokta çizileceğini belirtir.
- **GL_LINES** : Verilen noktaların birleştirilerek bir çizgi çizileceğini belirtir.
- **GL_POLYGON** : Verilen noktaların birleştirilerek doğrular oluşturulacağını ve oluşan şeklin alansal bir şekil olacağını, içinin renklendirileceğini belirtir.
- **GL_QUADS** : Verilen dört noktadan içi renklendirilmiş bir dörtgen oluşturulacağını belirtir.
- **GL_TRIANGLES** : Verilen üç noktadan içi renklendirilmiş üçgen oluşturulacağını belirtir.
- **GL_TRIANGLES_STRIP** : Verilen noktaların üçer üçer sırasıyla birleştirilerek üçgenler oluşturulacağını belirtir.
- **GL_QUAD_STRIP** : Verilen noktaların dörder dörder sırasıyla birleştirilerek dörtgenler oluşturulacağını belirtir.
- **GL_TRIANGLE_FAN** : Verilen noktaların ilk nokta ikişer ikişer alınıp her adımda ilk noktayı üçüncü nokta kabul ederek birleştirileceğini ve yelpazemsi bir şekil oluşturulacağını belirtir.

Örnek OpenGL Uygulaması

```
void glEnd(glVoid);
```

glBegin() fonksiyonu ile başlatılmış çizim işleminin bittiğini belirtir. Çizdirilen şekil ekrana bastırılmak üzere saklanır. Saklanmış bu şeklin çizdirilmesi için başka bir fonksiyon kullanılır.

```
void glFlush(glVoid);
```

Tampon bellekteki tüm şekillerin ekrana çizdirilmesini sağlar.

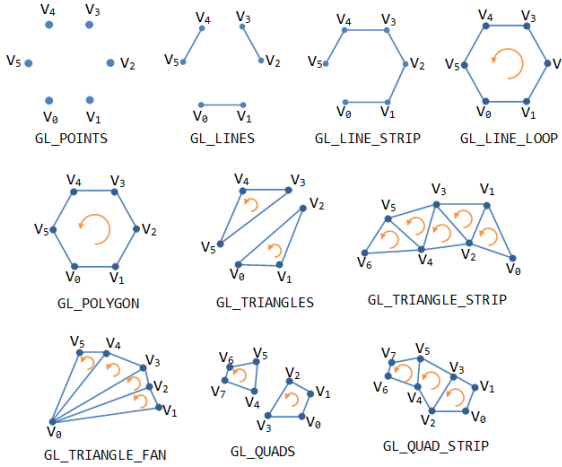
Örnek OpenGL Uygulaması

```
void glVertex2f(float x,float y);
```

Bir geometrik şekle ait kontrol noktasının koordinat değerlerini belirtir. Geometrik şeklin ne olacağı **glBegin** fonksiyonunun parametresi ile belirlenir.

```
glBegin(GL_POLYGON);  
glColor3f(1.0f, 1.0f, 0.0f);  
glVertex2f(0.4f, 0.2f);  
glVertex2f(0.6f, 0.2f);  
glVertex2f(0.7f, 0.4f);  
glVertex2f(0.6f, 0.6f);  
glVertex2f(0.4f, 0.6f);  
glVertex2f(0.3f, 0.4f);  
glEnd();
```

Örnek OpenGL Uygulaması



```
glBegin(GL_POLYGON);
glColor3f(1.0f, 1.0f, 0.0f);
glVertex2f(0.4f, 0.2f);
glVertex2f(0.6f, 0.2f);
glVertex2f(0.7f, 0.4f);
glVertex2f(0.6f, 0.6f);
glVertex2f(0.4f, 0.6f);
glVertex2f(0.3f, 0.4f);
glEnd();
```

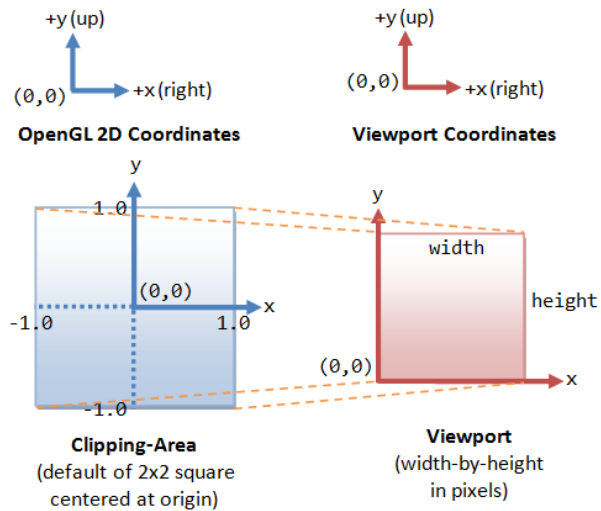
Örnek OpenGL Uygulaması

```
void glVertex2f(float x, float y);
```

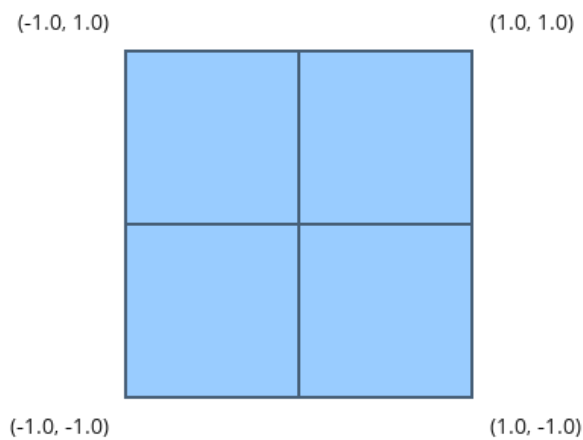
Bu fonksiyon iki boyutlu çizim yapılacağı zaman kullanılır. Koordinatın sadece x ve y değerleri verilir. Bu fonksiyon ile aynı işi yapan `glVertex2s`, `glVertex2i`, `glVertex2d` fonksiyonları sırasıyla `short`, `integer`, `double` türden parametre değişkenleri alırlar.

```
typedef unsigned char GLboolean;
typedef unsigned int GLbitfield;
typedef void GLvoid;
typedef signed char GLbyte; /* 1-byte signed */
typedef short GLshort; /* 2-byte signed */
typedef int GLint; /* 4-byte signed */
typedef unsigned char GLubyte; /* 1-byte unsigned */
typedef unsigned short GLushort; /* 2-byte unsigned */
typedef unsigned int GLuint; /* 4-byte unsigned */
typedef int GLsizei; /* 4-byte signed */
typedef float GLfloat; /* single precision float */
typedef float GLclampf; /* single precision float in [0,1] */
typedef double GLdouble; /* double precision float */
typedef double GLclampd; /* double precision float in [0,1] */
```

Örnek OpenGL Uygulaması



Örnek OpenGL Uygulaması



Örnek OpenGL Uygulaması

```
void glVertex3f(float x,float y,float z);
```

Bir geometrik şekle ait kontrol noktasının koordinat değerlerini belirtir. Geometrik şeklin ne olacağı glBegin fonksiyonunun parametresi ile belirlenir. Bu fonksiyon üç boyutlu çizim yapılacağı zaman kullanılır.

Koordinatın x ve y değerlerine ilaveten z değeri de verilir.

Bu fonksiyon ile aynı işi yapan glVertex3s, glVertex3i, glVertex3d fonksiyonları sırasıyla short, integer, double türden parametre değişkenleri alırlar.

Örnek OpenGL Uygulaması

```
void glColor3f(float kirmizi,float yesil,float mavi);
```

Çizilecek şeklin rengini belirler. Ön tanımlı değerler 0'dır.

```
void glRectf(float x1,float y1,float x2,float y2);
```

Parametre olarak geçilen koordinat değerlerini sol alt köşe ve sağ üst köşe olarak kabul ederek içi dolu bir dikdörtgen çizer.

Derleme

Unix/linux

Genellikle ../include/GL dizinindeki dosyaları dahil et

–lglut –lgl yükleyici bayraklarıyla derleyin

X kütüphaneleri için –L bayrağı eklemeniz gerekebilir

Mesa uygulaması çoğu linux dağıtımına dahildir

Mesa ve glut'un en son sürümleri için web'e bakın

Derleme

Visual C++

Web'den glut.h, glut32.lib ve glut32.dll dosyasını edinin

Karşılık gelen OpenGL dosyalarıyla aynı yerlere kurun

Boş bir uygulama oluşturun

Proje ayarlarına glut32.lib ekleyin (bağlantı sekmesi altında)

Freeglut ve GLEW için aynı

OpenGL İlkelerin Çizimi

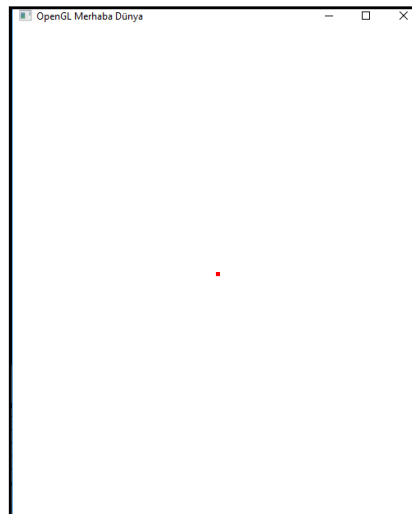
```
#include "glew.h"
#include "freeglut.h"
#include <iostream>
void display(void) {
    //...
}
int main(int argc, char *argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
    glutInitWindowPosition(500, 200);
    glutInitWindowSize(500, 600);
    glutCreateWindow("OpenGL Merhaba Dünya");
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Dr. Ömer ÇETİN

55

OpenGL İlkelerin Çizimi

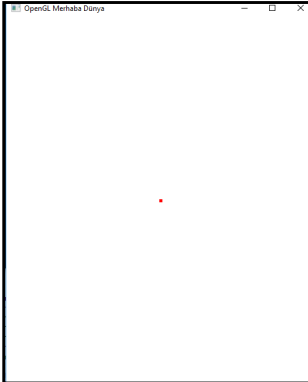
```
void display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glPointSize(5.0f);
    glColor3f(1, 0, 0);
    glBegin(GL_POINTS);
        glVertex2f(0, 0);
    glEnd();
    glFlush();
}
```



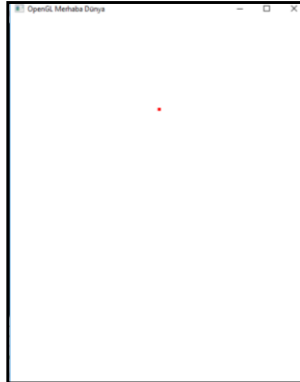
Dr. Ömer ÇETİN

56

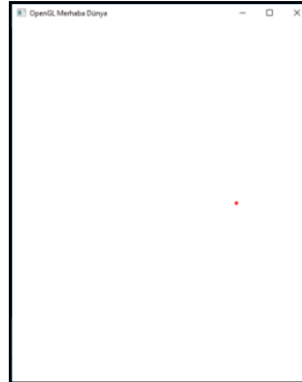
OpenGL İlkelerin Çizimi



`glVertex2f(0, 0);`



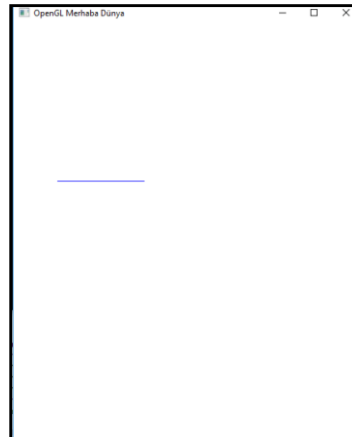
`glVertex2f(0.5, 0);`



`glVertex2f(0, 0.5);`

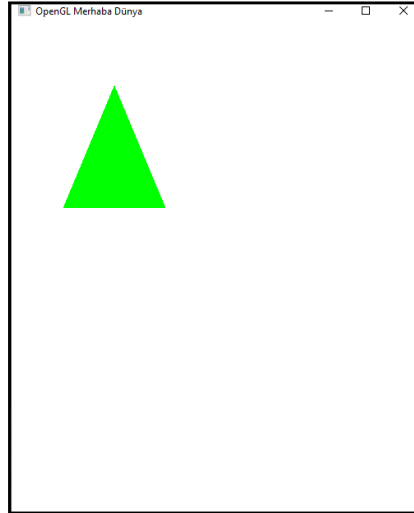
OpenGL İlkelerin Çizimi

```
void display(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glColor3f(1.0, 0.0, 0.0);  
    glPointSize(5.0f);  
    glColor3f(0, 0, 1);  
    glBegin(GL_LINES);  
        glVertex2f(-0.25, 0.25);  
        glVertex2f(-0.75, 0.25);  
    glEnd();  
    glFlush();  
}
```



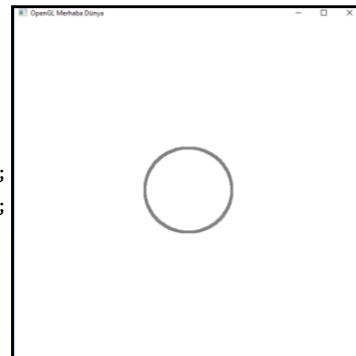
OpenGL İlkelerin Çizimi

```
void display(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glColor3f(1.0, 0.0, 0.0);  
    glPointSize(5.0f);  
    glColor3f(0, 1, 0);  
    glBegin(GL_POLYGON);  
        glVertex2f(-0.25, 0.25);  
        glVertex2f(-0.75, 0.25);  
        glVertex2f(-0.5, 0.75);  
    glEnd();  
    glFlush();  
}
```



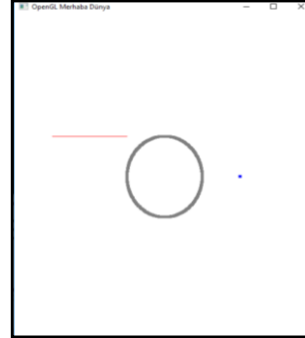
OpenGL İlkeler Kullanarak Çizim

```
void display(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glColor3f(0.5, 0.5, 0.5);  
    glPointSize(5.0);  
    glBegin(GL_POINTS);  
        for (int i = 1; i < 360; i++) {  
            float x = 0.25 * sin(((float)i) * 3.14 / 180);  
            float y = 0.25 * cos(((float)i) * 3.14 / 180);  
            glVertex2f(x, y);  
        }  
    glEnd();  
    glFlush();  
}
```

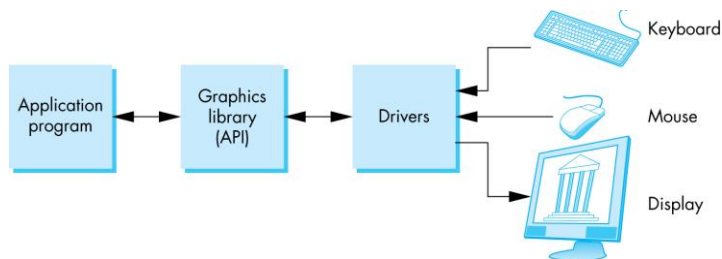


Şekillerin Bir Arada Çizimi

```
void display(void) {  
    glClearColor(GL_COLOR_BUFFER_BIT);  
    glPointSize(5.0f);  
  
    glColor3f(0, 0, 1);  
    glBegin(GL_POINTS);  
        glVertex2f(0.5, 0);  
    glEnd();  
  
    glColor3f(1, 0, 0);  
    glBegin(GL_LINES);  
        glVertex2f(-0.25, 0.25);  
        glVertex2f(-0.75, 0.25);  
    glEnd();  
  
    glColor3f(0.5, 0.5, 0.5);  
    glBegin(GL_POINTS);  
        for (int i = 1; i < 360; i++) {  
            float x = 0.25 * sin(((float)i) * 3.14 / 180);  
            float y = 0.25 * cos(((float)i) * 3.14 / 180);  
            glVertex2f(x, y);  
        }  
    glEnd();  
    glFlush();  
}
```



OpenGL Mimarisi



Ödev

OpenGL - klavye ile etkileşim...

