Scissor Test

(glScissor)

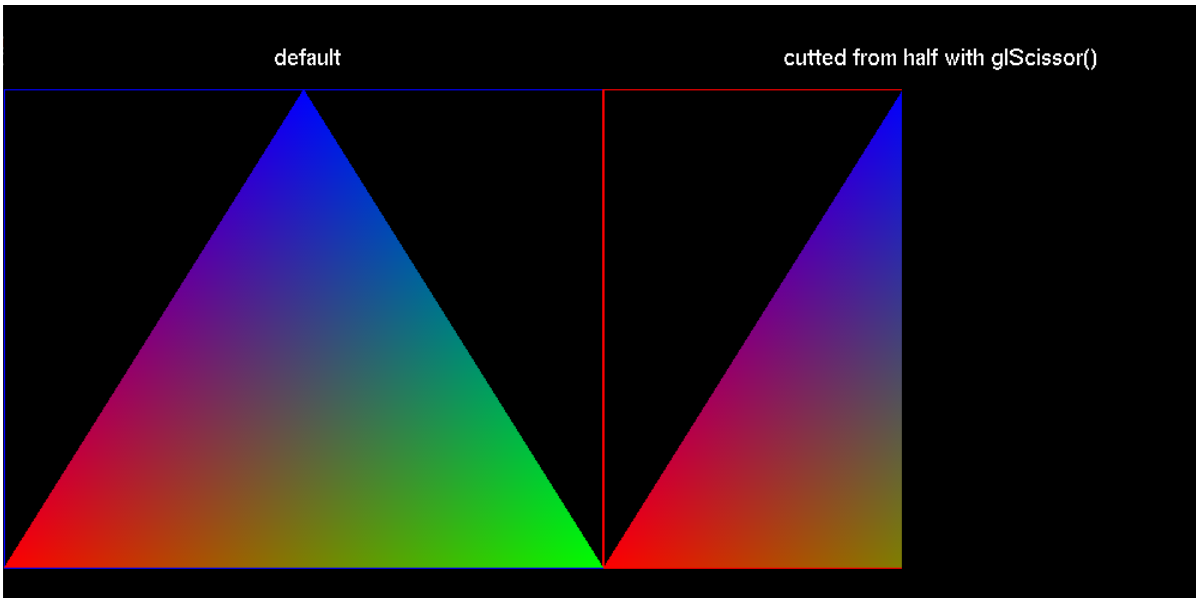
glScissor fonksiyonu, pencerenin belirli bir bölgesini kırpmak için kullanılır. Bu işlem, çizimlerin sadece belirli bir dikdörtgen bölge içinde görünmesini sağlar, bu dikdörtgen "scissor box" şeklinde isimlendirilir. Ekranda yalnızca belirli bir alanın güncellenmesi gereken durumlarda kullanışlıdır. Fonksiyonun aldığı parametreler şu şekildedir:

glScissor(GLint x, GLint y, GLsizei width, GLsizei height);

x: Scissor box'ın sol alt köşesinin x koordinatı
y: Scissor box'ın sol alt köşesinin y koordinatı
width: Scissor box'ın genişliği
height: Scissor box'ın yüksekliği

Scissor Test işlemini etkinleştirmek için ***glEnable(GL_SCISSOR_TEST)***, Scissor işlemi tamamlandıktan sonra tekrar kapatmak için ***glDisable(GL_SCISSOR_TEST)*** çağrısı yapılmalıdır. Örnek kullanımı aşağıdaki gibidir:

```
glEnable(GL_SCISSOR_TEST);
glScissor(-1.0, -0.5, 750, 600);
// İstenilen şekiller çizilir
glDisable(GL_SCISSOR_TEST);
```



Pencere Boyutu (1000px,800px)

Şekilde görülen ilk üçgen pencerede tam gözükürken yapılan Scissor Test işleminden dolayı ekran boyutu (750,600) boyutuna düşmüş ve ikinci üçgenin sadece yarısı görüntülenebilmektedir.

Sample Coverage

(glSampleCoverage)

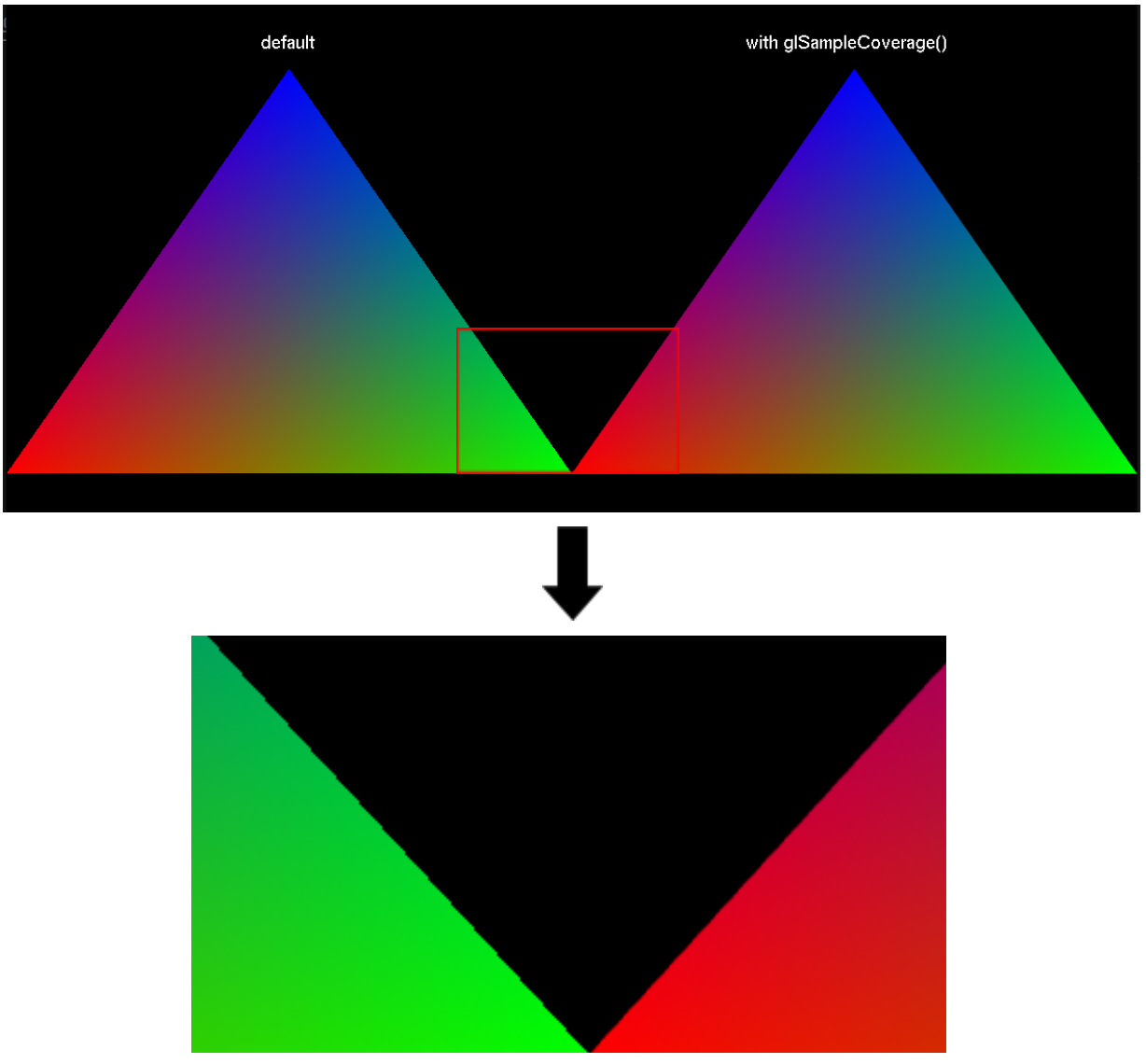
Grafik işlemcilerde anti-aliasing (kenar yumuşatma) tekniklerinden biri olan Multi-Sample Anti-Aliasing (MSAA) yöntemi için kullanılan bir terimdir. MSAA, kenarları daha pürüzsüz ve daha doğal görünmesini sağlamak için piksel renklerini yumuşatmak için kullanılır.

glSampleCoverage(GLclampf value, GLboolean invert);

value: Piksellerin örnekleme kaplaması değeri. 0.0 ile 1.0 arasında bir değer olmalıdır. Değer ne kadar yüksekse, piksellerin daha fazla örnekleneceği anlamına gelir.

invert: GL_TRUE olarak ayarlanırsa, kaplamaya dahil edilen örnekleme değeri tersine çevrilir (örneğin, 0.25 yerine 0.75 gibi). GL_FALSE olarak ayarlanırsa, kaplama değeri olduğu gibi kullanılır.

```
glEnable(GL_MULTISAMPLE);
glSampleCoverage(0.5, GL_FALSE);
glDisable(GL_MULTISAMPLE);
```



Stencil Test

Stencil testi, piksellerin render edilip edilmeyeceğini veya render sırasında nasıl etkileneceğini belirlemek için stencil tamponu kullanır. Stencil tamponu, ekran üzerindeki her pikselin bir değer tuttuğu ve stencil testinde kullanılan bir tampondur. Her piksel render edilirken stencil testi, stencil tamponundaki değeri belirli bir kriterle karşılaştırır ve sonuca göre pikselin görünürlüğünü değiştirir veya değiştirmeme kararını verir. Stencil Test işlemini etkinleştirmek için ***glEnable(GL_STENCIL_TEST)***, Scissor işlemi tamamlandıktan sonra tekrar kapatmak için ***glDisable(GL_STENCIL_TEST)*** çağrısı yapılmalıdır.

(glStencilFunc)

Bu fonksiyon stencil testini yapılandırmak için kullanılır. Aldığı parametreler şu şekildedir:

glStencilFunc(GLenum func, GLint ref, GLuint mask);

func: Karşılaştırma fonksiyonunu belirler. Bu, stencil tamponundaki değeri ref değeriyle karşılaştırmak için kullanılır. 8 çeşit sabiti vardır:

GL_NEVER: Stencil testi daima başarısız olduğu durumdur.

GL_LESS: Stencil tamponundaki değer, referans değerinden daha küçükse stencil testi başarılı olur.

GL_LEQUAL: Stencil tamponundaki değer, referans değerinden küçük veya eşitse stencil testi başarılı olur.

GL_GREATER: Stencil tamponundaki değer, referans değerinden daha büyükse stencil testi başarılı olur.

GL_GEQUAL: Stencil tamponundaki değer, referans değerinden büyük veya eşitse stencil testi başarılı olur.

GL_EQUAL: Stencil tamponundaki değer, referans değerine eşitse stencil testi başarılı olur.

GL_NOTEQUAL: Stencil tamponundaki değer, referans değerine eşit değilse stencil testi başarılı olur.

GL_ALWAYS: Bu seçenek, stencil testini her zaman geçerli yapar. (Başlangıç değeri GL_ALWAYS'dir.)

ref: Karşılaştırma için kullanılacak referans değeri. Bu değer, func ile belirtilen karşılaştırma işlemine tabi tutulacaktır.

mask: Test tamamlandığında hem referans değeri hem de saklanan kalıp değeri ile AND'lenen bir maske belirtir. İlk değerin tümü 1'dir.

```
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS, 1, 0xFF);
// bir başka kullanım örneği
glStencilFunc(GL_NOTEQUAL, 1, 0xFF);
glDisable(GL_STENCIL_TEST);
```

(glStencilFuncSeperate)

glStencilFunc işlevine benzerdir, ancak farklı yüzeylere (ön yüz ve arka yüz) ayrı ayrı stencil testi yapılandırmak için kullanılır. Bu işlev, bir nesnenin farklı yüzlerine farklı stencil testleri uygulamak istediğimiz durumlar için kullanışlıdır.

glStencilFunc(GLenum face, GLenum func, GLint ref, GLuint mask);

face: Stencil testini yapılandırmak için ön yüz (GL_FRONT), arka yüz (GL_BACK) veya hem ön hem arka yüz (GL_FRONT_AND_BACK) seçeneklerinden birini belirler.
func: Karşılaştırma fonksiyonunu belirler. Bu, stencil tamponundaki değeri ref değeriyle karşılaştırmak için kullanılır. Örneğin, GL_EQUAL, GL_NOTEQUAL, GL_LESS, GL_LEQUAL, GL_GREATER, GL_GEQUAL, vb.
ref: Karşılaştırma için kullanılacak referans değeri. Bu değer, func ile belirtilen karşılaştırma işlemine tabi tutulacaktır.
mask: ref değeriyle stencil tamponundaki değer arasında yapılacak AND işlemi için kullanılacak maske. Bu, ref değerinin belirli bitlerini kontrol etmek için kullanılabilir.

```
glEnable(GL_STENCIL_TEST);
// Ön yüzde stencil tamponu değeri 1'e eşit olan pikselleri geçir.
glStencilFuncSeparate(GL_FRONT, GL_EQUAL, 1, 0xFF);
// Arka yüzde her pikseli geçir.
glStencilFuncSeparate(GL_BACK, GL_ALWAYS, 0, 0xFF);
glDisable(GL_STENCIL_TEST);
```

(glStencilOp)

Stencil testleri sonrasında stencil tamponundaki değerlerin nasıl güncelleneceğini belirlemek için kullanılır. Özellikle, gölgelendirme efektleri, kesme (clipping) işlemleri ve seçim işlemleri gibi çeşitli grafik tekniklerde önemli bir rol oynar.

glStencilOp(GLenum sfail, GLenum dpfail, GLenum dppass);

sfail: Stencil testi başarısız olursa (yani stencil testi, stencil tamponundaki değeri referans değeri ile karşılaştırmada geçemezse), bu durumda stencil tamponunda ne tür bir işlem gerçekleşeceğini belirler.
dpfail: Stencil testi başarılı olur ancak derinlik (depth) testi başarısız olursa (yani stencil testi başarılı olsa da, derinlik tamponundaki değeri referans değeri ile karşılaştırmada geçemezse), bu durumda stencil tamponunda ne tür bir işlem gerçekleşeceğini belirler.
dppass: Stencil testi ve derinlik testi başarılı olursa (yani hem stencil testi hem de derinlik testi, stencil tamponundaki değeri referans değeri ile karşılaştırmada geçerse), bu durumda stencil tamponunda ne tür bir işlem gerçekleşeceğini belirler.

Bu parametrelerin değerleri şu şekildedir: GL_KEEP, GL_ZERO, GL_REPLACE, GL_INCR, GL_DECR, GL_INVERT

```
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_EQUAL, 1, 0xFF);
// stencil testi başarılı olursa değeri 1 artır.
glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);
// Burada çizim işlemleri gerçekleştirilir ve stencil tamponu değeri 1 olan pikseller üzerinde bir etki elde edilir.
glDisable(GL_STENCIL_TEST);
```

(glStencilOpSeperate)

Ön yüz ve arka yüz yüzeyler için ayrı ayrı stencil işlemlerini ayarlayan bir işlevdir. Bu işlev, stencil testi sırasında ön yüz ve arka yüz üçgenleri için farklı stencil işlemleri belirtmenizi sağlar. Özellikle gölgelerin oluşturulduğu veya kenar hatlarındaki üçgenlerin işlendiği gibi karmaşık render tekniklerinde, ön yüz ve arka yüzeyler için farklı stencil işlemleri uygulamak istediğinizde kullanışlıdır.

glStencilOpSeparate(GLenum face, GLenum sfail, GLenum dpfail, GLenum dppass)

face: Hangi yüzeyin stencil durumunu güncellemek istediğinizi belirtir. İki değer alabilir: GL_FRONT ve GL_BACK

sfail: Stencil testinin başarısız olduğu durumda alınacak işlemi belirtir.

dpfail: Stencil testinin başarılı olduğu ancak derinlik testinin başarısız olduğu durumda alınacak işlemi belirtir.

dppass: Hem stencil testinin hem de derinlik testinin başarılı olduğu durumda alınacak işlemi belirtir.

```
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS, 1, 0xFF);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
glStencilOpSeparate(GL_BACK, GL_KEEP, GL_INCR_WRAP, GL_KEEP);
glDisable(GL_STENCIL_TEST);
```

Depth Buffer Test

Depth buffer her pikselin ekran üzerindeki derinlik değerini tutan bir tampondur. Depth Buffer Test, her bir nesnenin derinlik değerini derinlik tamponu ile karşılaştırır ve eğer nesnenin derinliği, tampondaki değerden daha yakınsa, o piksel ekran üzerinde görünür hale gelir. Aksi takdirde, piksel görünmez olur ve arka taraftaki nesnelerin önünde kalır. Bu işlem, sahnede çeşitli nesnelerin katman katman render edilmesini sağlar ve sonuç olarak gerçekçi 3D görüntülerin elde edilmesine yardımcı olur. Depth Buffer Test, özellikle karmaşık sahnelerde ve nesnelerin birbirinin önünde olduğu durumlarda, doğru görsel sıralamayı sağlamak için önemlidir.

Depth Test işlemini etkinleştirmek için ***glEnable(GL_DEPTH_TEST)***, işlem tamamlandıktan sonra tekrar kapatmak için ***glDisable(GL_DEPTH_TEST)*** çağrısı yapılmalıdır.

(glDepthFunc)

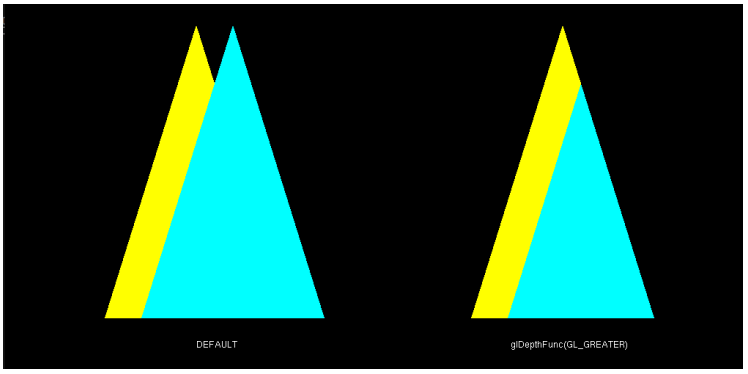
glDepthFunc(GLenum func);

func: derinlik tamponu testinin nasıl yapılandırılacağını belirler ve birçok parametre alır. Parametreler şu şekildedir:

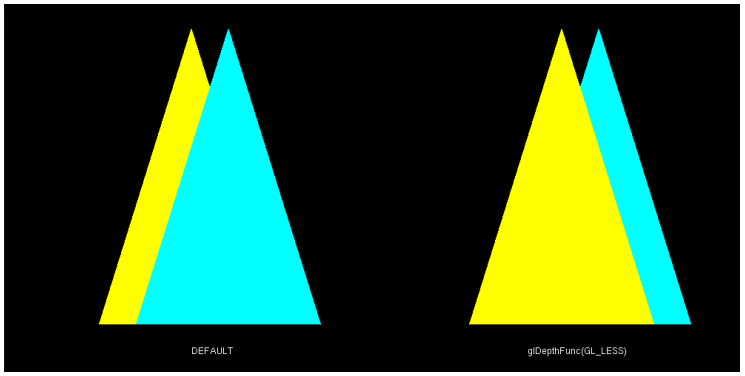
GL_NEVER, GL_ALWAYS, GL_LESS, GL_LEQUAL, GL_EQUAL, GL_GREATER, GL_GEQUAL, GL_NOTEQUAL

```
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_GREATER);
// bir başka kullanım örneği
glDepthFunc(GL_LESS);
glDisable(GL_DEPTH_TEST);
```

Yukarıdaki kodla yapılmış bir örnek aşağıda gösterilmiştir:



GL_GREATER



GL_LESS

Blending

Blending, çeşitli renk ve alfa kanallarını birbirine karıştırarak saydamlık ve gölgeleme efektleri oluşturmayı sağlayan bir grafik işleme tekniğidir. Bu teknik, nesneleri veya pikselleri ekran üzerinde birbirine karıştırmak için kullanılır ve özellikle saydam veya yarı saydam nesneleri görüntülemek için yaygın olarak kullanılır.
Etkinleştirmek için ***glEnable(GL_BLEND)***, işlem tamamlandıktan sonra tekrar kapatmak için ***glDisable(GL_BLEND)*** çağrısı yapılmalıdır.

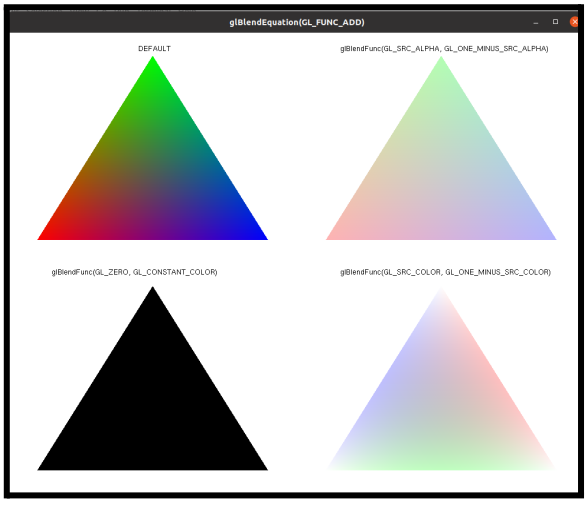
(glBlendEquation)

Blend Equations, yeni bir pikselin ("source" rengi) hali hazırda framebufferda bulunan bir pikselle ("destination" rengi) nasıl birleştirileceğini belirler. Bu işlev, hem RGB karışım denklemini hem de alfa karışım denklemini tek bir denkleme ayarlar.

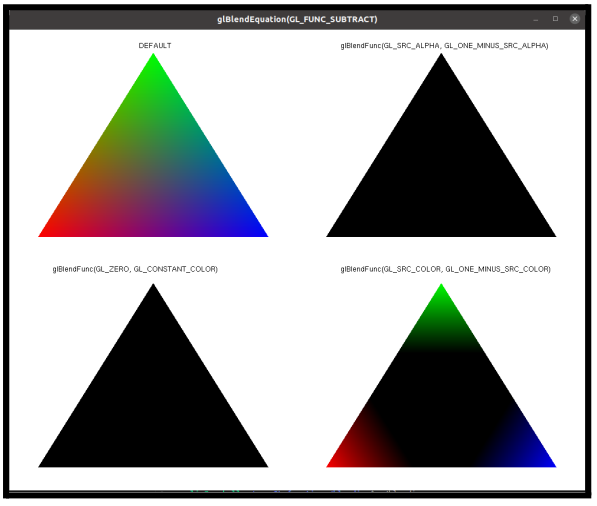
glBlendEquation(GLenum mode);

mode: blending işlemi sırasında kullanılacak denklemin türünü belirler. Bu parametre, aşağıdaki değerlerden birini alabilir:
GL_FUNC_ADD: Source ve destination renklerin doğrudan toplandığı varsayılan blending denklemidir.
GL_FUNC_SUBTRACT: source renklerin destination renklerden çıkarıldığı bir blending denklemidir.
GL_FUNC_REVERSE_SUBTRACT: Destination renklerin source renklerden çıkarıldığı bir blending denklemidir.
GL_MIN: Source ve destination renkler arasından en küçük olanını seçer.
GL_MAX: Source ve destination renkler arasından en büyük olanını seçer.
*(V*arsayılan olarak glBlendEquation(GL_FUNC_ADD) kullanılır, ancak ihtiyaç duyulduğunda diğer denklemler de kullanılabilir.)

```
glEnable(GL_BLEND);
glBlendEquation(GL_FUNC_ADD); // Resim 1 için kullanımı
glBlendEquation(GL_FUNC_SUBTRACT); // Resim 2 için kullanımı
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glDisable(GL_BLEND);
```



Resim 1

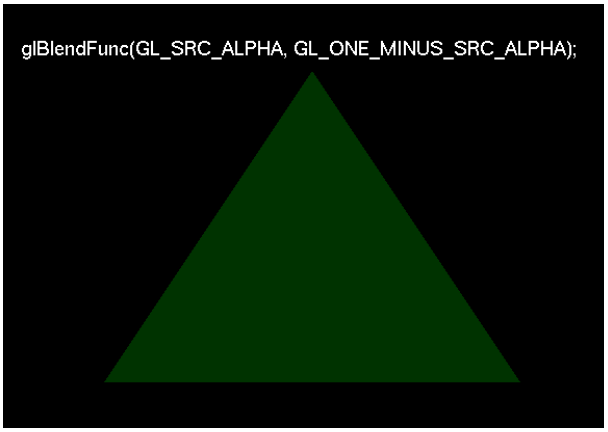


Resim 2

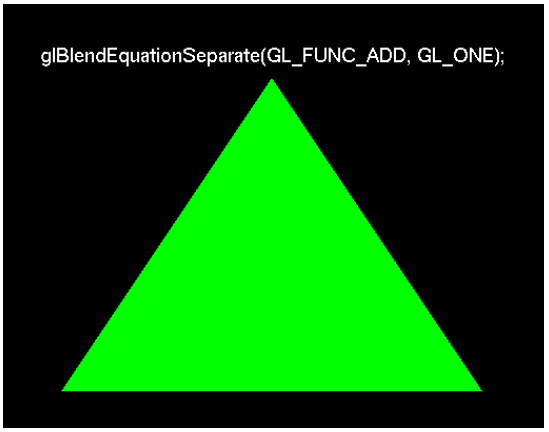
(glBlendEquationSeperate)
Birden fazla blend denklemi belirlemek için kullanılan bir işlevdir. RGB ve alfa blend denklemini ayrı ayrı tanımlamamızı sağlar.

glBlendEquationSeparate(GLenum modeRGB, GLenum modeAlpha);
modeRGB: RGB bileşenleri için blend denklemi türünü belirten bir GLenum değeri.
modeAlpha: Alfa bileşeni için blend denklemi türünü belirten bir GLenum değeri.

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
// Resim 3'deki üçgen çizdirilir
glBlendEquationSeparate(GL_FUNC_ADD, GL_ONE);
// Resim 4'deki üçgen çizdirilir
glDisable(GL_BLEND);
```



Resim 3



Resim 4

(glBlendFuncSeperate)

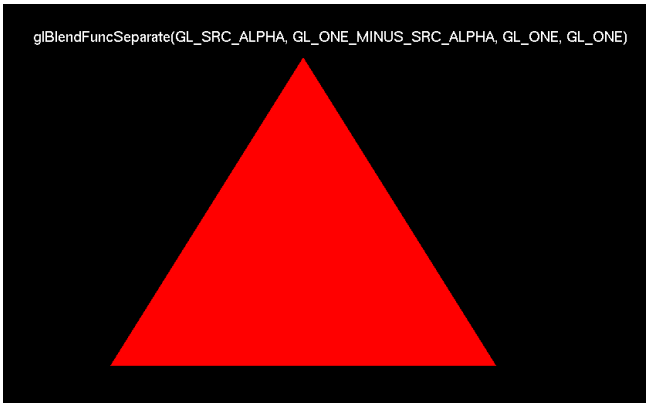
RGB ve alpha blend işlemlerinin ayrı ayrı blend fonksiyonlarını belirlemek için kullanılır. Bu işlev, ***glBlendFunc*** işleminin genişletilmiş bir versiyonudur ve birden fazla renk ve alfa bileşenleri için ayrı ayrı blend işlemi belirleme imkanı sunar.

glBlendFuncSeparate(GLenum srcRGB, GLenum dstRGB, GLenum srcAlpha, GLenum dstAlpha);

srcRGB: Source renk bileşeninin blend işlemindeki kaynağı.
dstRGB: Destination renk bileşeninin blend işlemindeki hedefi.
srcAlpha: Source alfa bileşeninin blend işlemindeki kaynağı.
dstAlpha: Destination alfa bileşeninin blend işlemindeki hedefi.

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
// Resim 5'deki üçgen çizdirilir
glBlendFuncSeparate(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA,GL_ONE,GL_ONE);
// Resim 6'daki üçgen çizdirilir
glDisable(GL_BLEND);
```

Resim 5



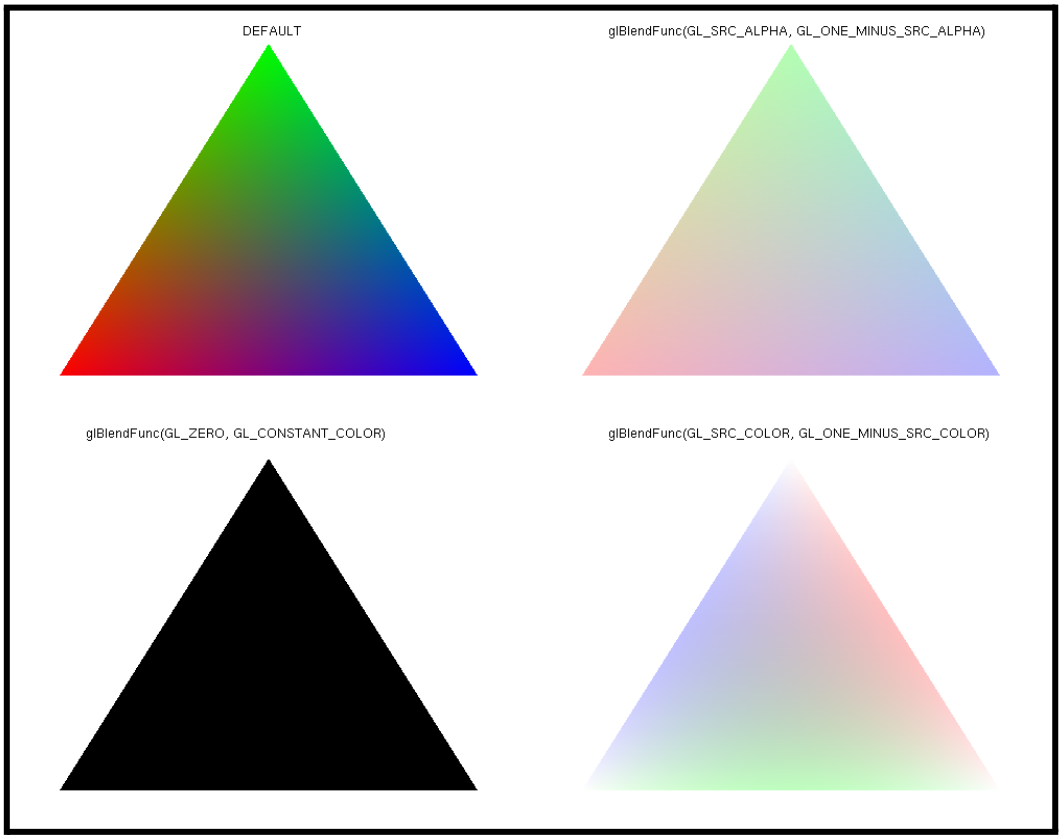
Resim 6

(glBlendFunc)
Renklerin karıştırılması (blending) işlemlerini yapılandırmak için kullanılır. Bu işlem, çizilen nesnelerin renklerini önceki piksellerin üzerine bindirerek veya çeşitli karışım yöntemleriyle birleştirerek gerçekleştirilir. Bu sayede, saydamlık ve gölgeleme gibi efektler oluşturulabilir.

glBlendFunc(GLenum sfactor, GLenum dfactor);

sfactor ve dfactor parametreleri karışım için source ve destination faktörleri belirtir. İki faktör arasındaki kombinasyon, çizilen pikselin rengini önceki piksellerle nasıl birleştireceğini belirler.
sfactor: çizilen pikselin kaynak faktörüdür. Yani çizilen nesnenin renklerinin nasıl etkileşimde bulunacağını belirler.
dfactor: hedef pikselin faktörüdür. Yani arka planda kalan pikselin renginin nasıl etkileşime gireceğini belirtir.

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glBlendFunc(GL_ZERO, GL_CONSTANT_COLOR);
glBlendFunc(GL_SRC_COLOR, GL_ONE_MINUS_SRC_COLOR);
glDisable(GL_BLEND);
```



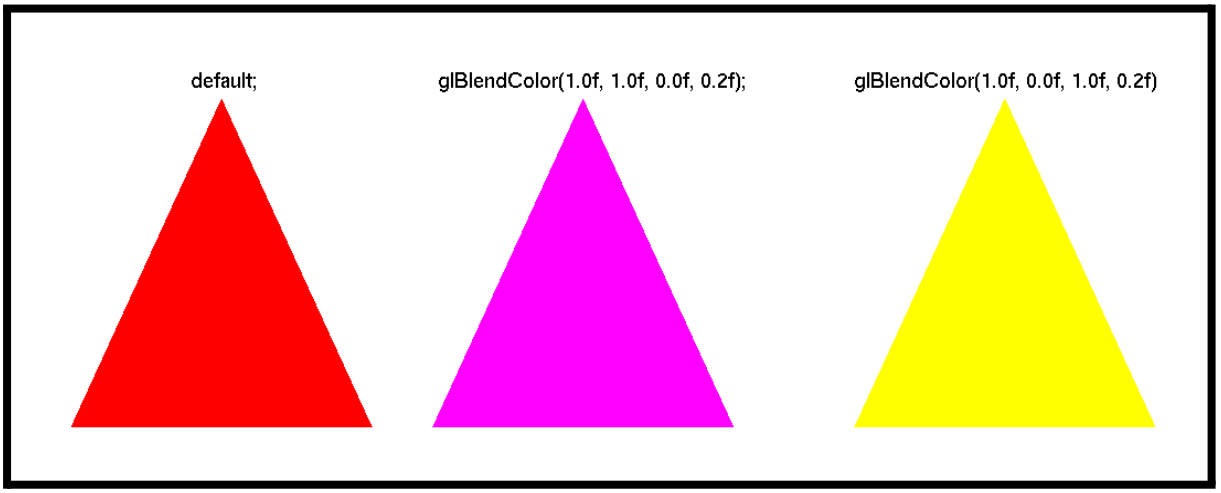
(glBlendColor)

Renk karıştırma (blending) işlemi için kullanılan karıştırma rengini belirlemek için kullanılan bir işlevdir. Bu işlev, bir karıştırma rengi belirlemek için kullanılır ve bu renk, glBlendFunc veya glBlendFuncSeparate işlevleriyle birleştirme işlemi sırasında kullanılır.

glBlendColor(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha);

Red, **green** ve **blue** parametreleri karıştırma renginin kırmızı, yeşil ve mavi bileşenlerini belirler. **Alpha** parametresi ise karıştırma renginin saydamlık bileşenini belirler.

```
glEnable(GL_BLEND);
// Kırmızı üçgen çizdirilir
glBlendFunc(GL_CONSTANT_COLOR, GL_ONE_MINUS_CONSTANT_COLOR);
glBlendEquation(GL_FUNC_ADD);
glBlendColor(1.0f, 1.0f, 0.0f, 0.2f);
// Kırmızı üçgen çizdirilir
glBlendColor(1.0f, 0.0f, 1.0f, 0.2f);
// Kırmızı üçgen çizdirilir
glDisable(GL_BLEND);
```

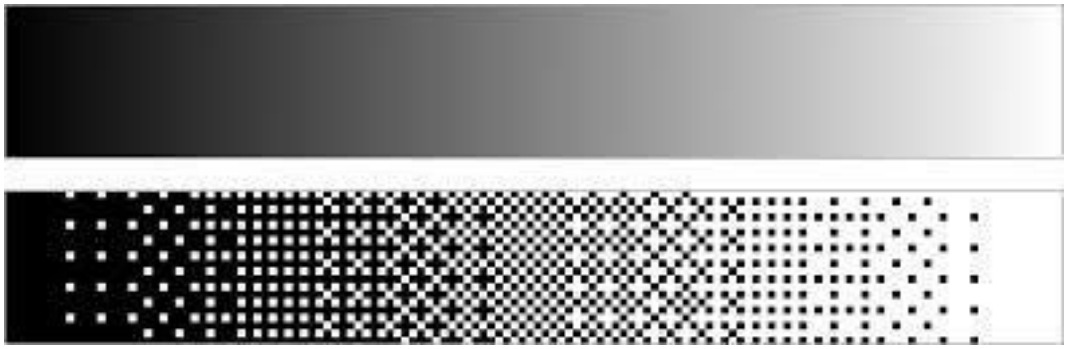


Dithering
(GL_DITHER)

Dithering, düşük renk çözünürlüğüne sahip ekranlarda veya renk paletleri ile çalışan grafik sistemlerinde daha fazla renk tonunu simüle etmek için kullanılan bir tekniktir. Bir ekran veya grafik sistemi düşük renk çözünürlüğüne sahip olduğunda, renkler arasındaki geçişler düzgün bir şekilde görüntülenemeyebilir ve bazı renk tonları kaybolabilir. Dithering, bu sorunu gidermek için renk tonları arasında hafif rastgele desenler veya parçacıklar ekleyerek optik yanılsama yaratır.

*Dithering efektini açmak ve kapatmak için **glEnable(GL_DITHER) glDisable(GL_DITHER)** çağrısl yapmamız gereklidir.*

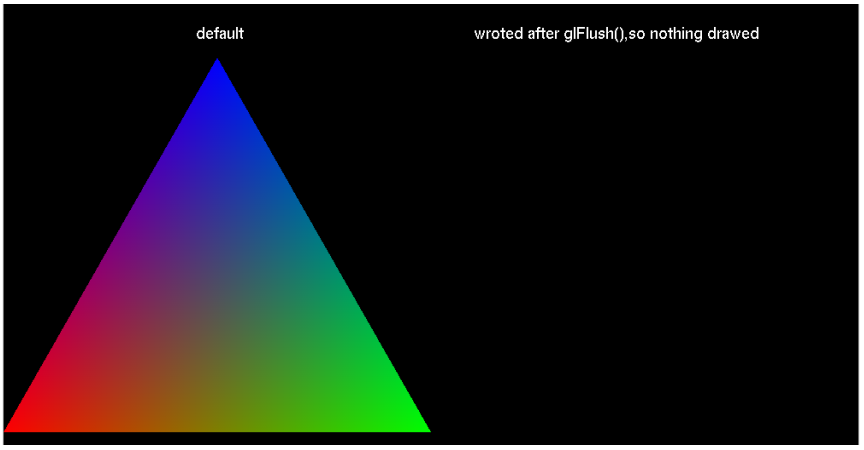
```
glEnable(GL_DITHER);
// Ekranda gözükecek şeyler
glDisable(GL_DITHER);
```



(glFlush)

Grafik verilerini hemen çizdirme komutlarının gerçekleştirilmesini sağlayan bir işlevdir. OpenGL, bir çizim işlemi gerçekleştirmek için çeşitli komutları bir komut havuzunda toplar ve bunları optimize etmek amacıyla bazen bekletebilir. glFlush fonksiyonu, çizim verilerinin tamamının işlemci üzerindeki komut havuzundan hemen çizdirilmesini ve ekrana aktarılmasını sağlar.

Aşağıdaki örnekte glFlush fonksiyonunu çağırmadan önce ve sonra çizdirilen bir üçgen görüntülenmektedir:



(glFinish)

Komut havuzunda bekleyen tüm komutların gerçekleştirilmesini sağlayan bir işlevdir. glFinish çağrısı yaptığınızda, OpenGL, komut havuzundaki tüm komutları sırayla ve tamamlayıcı bir şekilde işleyerek, bu komutların sonuçlarını beklemeye alır ve ardından glFinish işlevi tamamlandığında işlemi tamamlar.

glFinish fonksiyonu, glFlush fonksiyonundan farklıdır. glFlush, yalnızca komut havuzundaki işleri gerçekleştirir ve ekrana aktarırken, glFinish, komut havuzundaki tüm işleri tamamlayana kadar bekler. Bu nedenle, glFinish fonksiyonu, daha yüksek bir senkronizasyon seviyesi sağlar ve OpenGL işlemlerinin tamamlanmasını garantiler. Ancak, bu işlemin tamamlanması biraz zaman alabilir ve performans açısından maliyetli olabilir.

(glHint)

Bu işlev, OpenGL'ye belirli bir davranışın tercih edilen bir şekilde nasıl uygulanacağını belirtmek için kullanılır. glHint, uygulamaların OpenGL'in çeşitli optimizasyon ve işlemleme stratejilerini daha iyi anlamasına ve uygulamasına yardımcı olabilir.

glHint(GLenum target, GLenum mode);

target: Hint'in hangi özelliği veya davranışı etkileyeceğini belirtir. Alabileceği bazı parametreler:
GL_GENERATE_MIPMAP_HINT: Mipmap oluşturmanın optimizasyon hint.
GL_LINE_SMOOTH_HINT: Çizgi düzgünlüğü optimizasyon hint.
GL_POLYGON_SMOOTH_HINT: Poligon düzgünlüğü optimizasyon hint.
GL_TEXTURE_COMPRESSION_HINT: Texture sıkıştırma optimizasyon hint.

mode: Hint'in belirli bir hedef için nasıl uygulanacağını belirtir. Alabileceği bazı parametreler:
GL_FASTEST: En hızlı performansın hedeflendiği hint.
GL_NICEST: En yüksek kalitenin hedeflendiği hint.
GL_DONT_CARE: Performans veya kalite hedefi belirtilmeyen hint.

```
glEnable(GL_POINT_SMOOTH);
glHint(GL_POINT_SMOOTH_HINT, GL_NICEST);
glDisable(GL_POINT_SMOOTH);
```

State and State Requests

(glGetBooleanv)

Belirli bir özellik hakkında bir boolean değerini almak için kullanılır.

glGetBooleanv(GLenum pname, GLboolean* params);

pname: Bu parametre, sorgulanacak özelliği belirler. Bazı sorgulanabilir özellikler şunlardır:
GL_DEPTH_TEST: Derinlik testinin etkin olup olmadığını döndürür.
GL_BLEND: Karıştırma (blending) işleminin etkin olup olmadığını döndürür.
GL_CULL_FACE: Yüz çıkarma işleminin etkin olup olmadığını döndürür.
GL_LIGHTING: Aydınlatmanın etkin olup olmadığını döndürür.
params: Bu parametre, sorgulanacak değeri almak için bir işaretçiye işaret eder. Yani, işlev belirtilen özelliğin değerini bu işaretçiyle doldurur. Örneğin, GL_DEPTH_TEST özelliği etkinse, params işaretçisi GL_TRUE değeriyle doldurulur. Eğer özellik etkin değilse, GL_FALSE değeriyle doldurulur.

```
GLboolean depthTestEnabled;
glGetBooleanv(GL_DEPTH_TEST, &depthTestEnabled);
if (depthTestEnabled == GL_TRUE) {
    std::cout << "Depth testing is enabled." << std::endl;
}
```

(glGetIntegerv)

Belirli bir özellik hakkında bir tamsayı değerini almak için kullanılır.

glGetIntegerv(GLenum pname, GLint* params);

pname: Bu parametre, sorgulanacak özelliği belirler. Aldığı bazı parametreler:
GL_MAX_TEXTURE_SIZE: Maksimum texture boyutu.
GL_MAX_TEXTURE_UNITS: Desteklenen maksimum texture birimi sayısı.
GL_MAX_TEXTURE_COORDS: Maksimum texture koordinat sayısı.
GL_MAX_LIGHTS: Maksimum ışık sayısı.
params: Bu parametre, sorgulanacak değeri almak için bir işaretçiye işaret eder. Yani, işlev belirtilen özelliğin değerini bu işaretçiyle doldurur.

```
GLint maxTextureSize;
glGetIntegerv(GL_MAX_TEXTURE_SIZE, &maxTextureSize);

std::cout << "Max texture size: " << maxTextureSize << std::endl;
```

(glGetFloatv)

Belirli bir özellik hakkında bir float değerini almak için kullanılır.

glGetFloatv(GLenum pname, GLfloat* params);

pname: Bu parametre, sorgulanacak özelliği belirler. Aldığı bazı parametreler:
GL_CURRENT_COLOR: Geçerli renk değeri.
GL_LINE_WIDTH: Geçerli çizgi kalınlığı.
GL_DEPTH_CLEAR_VALUE: Derinlik tamamlayıcı değeri.
GL_POINT_SIZE: Geçerli nokta boyutu.
params: Bu parametre, sorgulanacak değeri almak için bir işaretçiye işaret eder. Yani, işlev belirtilen özelliğin değerini bu işaretçiyle doldurur.

```
GLfloat currentColor[4];
glGetFloatv(GL_CURRENT_COLOR, currentColor);

std::cout << "Current color: (" << currentColor[0] << ", " << currentColor[1] << ", " <<
currentColor[2] << ", " << currentColor[3] << ")" << std::endl;
```

(glIsEnabled)

Belirli bir özelliğin (GL capability) etkin olup olmadığını sorgulamak için kullanılan bir işlevdir.

glIsEnabled(GLenum cap);

cap: Sorgulanacak özelliği belirten sabit. Aldığı bazı parametreler:
GL_DEPTH_TEST: Derinlik testinin etkin olup olmadığını sorgular.
GL_BLEND: Karıştırma (blending) işleminin etkin olup olmadığını sorgular.
GL_TEXTURE_2D: 2 boyutlu doku haritalarının etkin olup olmadığını sorgular.
GL_CULL_FACE: Arka yüzeyleri gizleme (culling) işleminin etkin olup olmadığını sorgular.

```
bool depthTestEnabled = glIsEnabled(GL_DEPTH_TEST);
if (depthTestEnabled) {
    std::cout << "Depth test is enabled." << std::endl;
}
```

(glGetString)
Farklı bilgileri almak için kullanılır.

glGetString(GLenum name)

name: Alınmak istenen bilgiyi belirten sabit. Bu sabit, GLenum türünden bir değerdir ve farklı bilgileri almak için kullanılır. Örneğin:
GL_VENDOR: OpenGL uygulamasının üreticisinin adını döndürür.
GL_RENDERER: OpenGL uygulamasının mevcut render cihazının adını döndürür
GL_VERSION: OpenGL uygulamasının sürüm bilgisini döndürür.
GL_SHADING_LANGUAGE_VERSION: Shader dili (GLSL) sürüm bilgisini döndürür.

```
// OpenGL sürüm bilgisini al
const GLubyte* version = glGetString(GL_VERSION);
std::cout << "OpenGL Version: " << version << std::endl;

// Grafik kartı bilgisini al
const GLubyte* renderer = glGetString(GL_RENDERER);
std::cout << "Renderer: " << renderer << std::endl;

// Üretici bilgisini al
const GLubyte* vendor = glGetString(GL_VENDOR);
std::cout << "Vendor: " << vendor << std::endl;

// GLSL sürüm bilgisini al
const GLubyte* glslVersion = glGetString(GL_SHADING_LANGUAGE_VERSION);
std::cout << "GLSL Version: " << glslVersion << std::endl;
```

Whole Framebuffer Operations

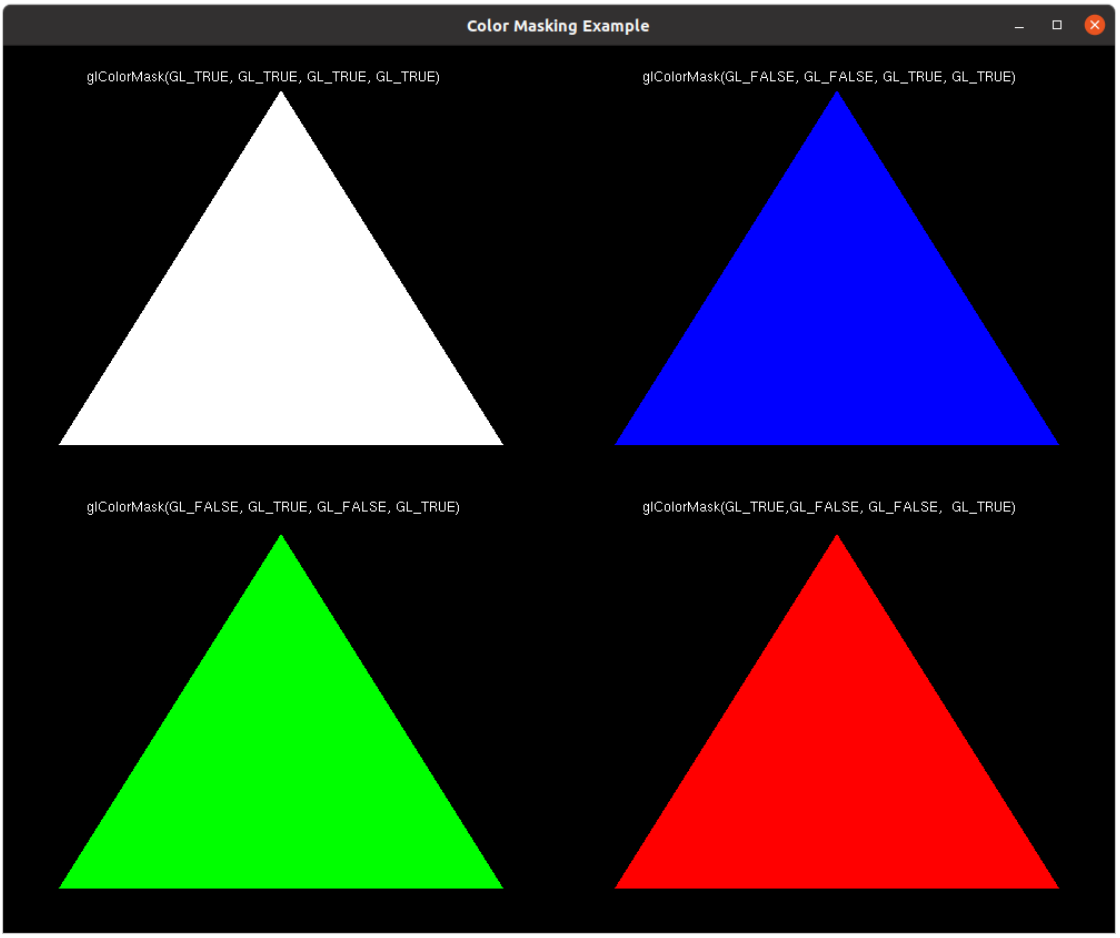
(glColorMask)

Renk kanallarının yazılmasını kontrol etmek için kullanılan bir işlevidir. Bu işlev, belirtilen renk kanallarını etkinleştirip veya devre dışı bırakarak, hangi renk kanallarının yazılacağını belirlemenize olanak tanır.

glColorMask(GLboolean red, GLboolean green, GLboolean blue, GLboolean alpha);

red: Kırmızı kanalının yazılmasını kontrol etmek için kullanılan boolean türünden bir değer.
green: Yeşil kanalının yazılmasını kontrol etmek için kullanılan boolean türünden bir değer.
blue: Mavi kanalının yazılmasını kontrol etmek için kullanılan boolean türünden bir değer.
alpha: Alfa (saydamlık) kanalının yazılmasını kontrol etmek için kullanılan -boolean türünden bir değer.

```
// Bütün kanalları çalıştır (Beyaz üçgen)
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
// Mavi üçgen
glColorMask(GL_FALSE, GL_FALSE, GL_TRUE, GL_TRUE);
// Yeşil üçgen
glColorMask(GL_FALSE, GL_TRUE, GL_FALSE, GL_TRUE);
// Kırmızı üçgen
glColorMask(GL_TRUE,GL_FALSE, GL_FALSE,  GL_TRUE);
```



(glDepthMask)

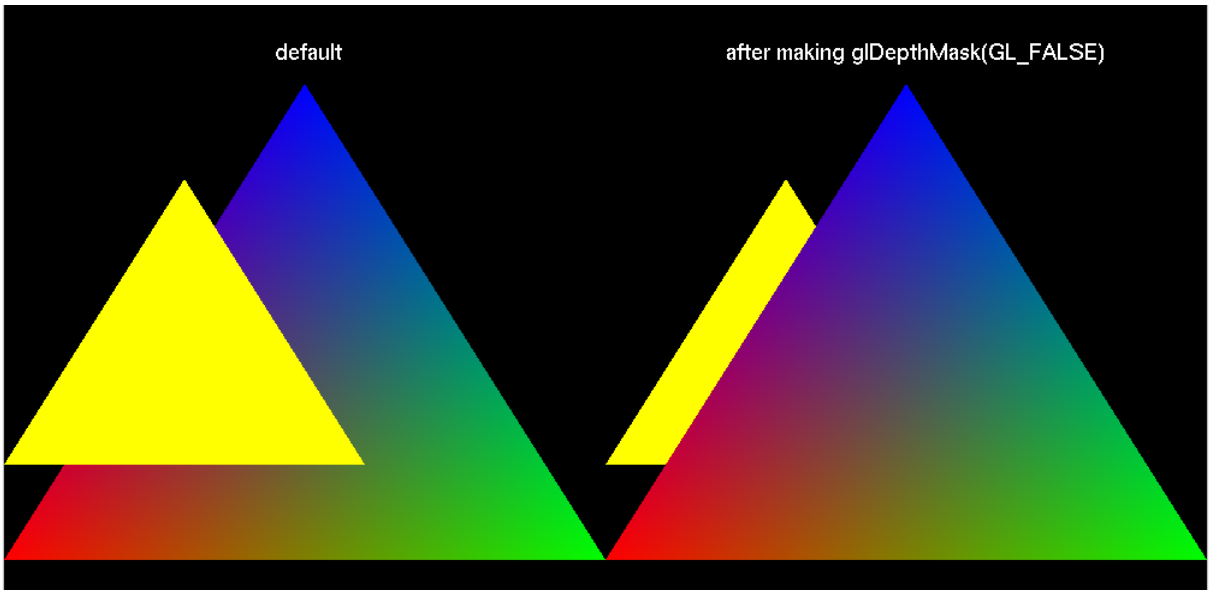
Derinlik bufferı yazma işlemini kontrol etmek için kullanılan bir işlevdir. Derinlik tamponu, sahnedeki nesnelerin uzaklıklarını saklamak için kullanılan bir tampondur ve genellikle 0.0 (en yakın) ile 1.0 (en uzak) arasında değerlere sahip olan bir piksel tablosudur.

glDepthMask(GLboolean flag);

flag: Derinlik bufferına yazma işlemini etkinleştirmek veya devre dışı bırakmak için kullanılan GLboolean türünden bir değer. Eğer GL_TRUE olarak ayarlanırsa, derinlik tamponu yazma işlemi etkinleştirilir ve yeni değerler derinlik tamponuna yazılır. Eğer GL_FALSE olarak ayarlanırsa, derinlik tamponu yazma işlemi devre dışı bırakılır ve derinlik tamponu güncellenmez.

(Parametrenin default hali ***GL_TRUE***' dur.)

```
glEnable(GL_DEPTH_TEST);
glDepthMask(GL_TRUE);
// Sol taraftaki üçgeni çizdir (default)
glDepthMask(GL_FALSE);
// Sağ taraftaki üçgeni çizdir
```



(glStencilMask)
Stencil bufferına yazma işlemini kontrol etmek için kullanılan bir işlevdir. Stencil tamponu, görüntü üzerindeki piksellerin stencil değerlerini saklamak için kullanılan bir tampondur.

glStencilMask(GLuint mask);

mask: Stencil bufferına yazma işlemini etkinleştirmek veya devre dışı bırakmak için kullanılan bit maskesidir. Bu maske, stencil tamponuna yazılacak değerleri belirler. İşlev, bu maske ile belirtilen bitleri stencil tamponuna yazmaya izin verir ve diğer bitleri değiştirmez. Maskeler, GLuint türünden olmalıdır ve 32 bit uzunluğundadır. Bazı maskeler şu şekildedir:
0xFFFFFFFF (tüm bitler 1): Tüm bitlere izin verir, herhangi bir değer stencil tamponuna yazılır.
0x00000000 (tüm bitler 0): Hiçbir bitlere izin vermez, hiçbir değer stencil tamponuna yazılmaz.
0x00000001 (sadece en düşük bit 1): Yalnızca en düşük bit değerine izin verir, yalnızca tek sayılar stencil tamponuna yazılır.
0xAAAAAAAA (1010 1010 1010 1010 1010 1010 1010 1010): Alternatif bitlere izin verir, örneğin 2'nin katları stencil tamponuna yazılır.
0x55555555 (0101 0101 0101 0101 0101 0101 0101 0101): Alternatif bitlere izin verir, örneğin 2'nin katları olmayanlar stencil tamponuna yazılır.

```
glEnable(GL_STENCIL_TEST);
glStencilMask(0x00000001);
glStencilFunc(GL_ALWAYS, 1, 0xFF)
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
glDisable(GL_STENCIL_TEST);
```

(glStencilMaskSeperate)
Stencil bufferına yazma işlemini ayrı ayrı kontrol etmeyi sağlar. Bu işlev, iki ayrı stencil buffer kullanıldığında kullanışlıdır.

glStencilMaskSeparate(GLenum face, GLuint mask);

face: Bu parametre, hangi stencil tamponunun (ön veya arka) işlemi etkileyeceğini belirler. Değer olarak GL_FRONT, ön stencil tamponunu, GL_BACK ise arka stencil tamponunu belirtir.
mask: Bu parametre, stencil tamponuna yazma işlemini etkileyecek maskeyi belirtir. Bitleri 1 olanlar stencil tamponuna yazılırken, bitleri 0 olanlar yazılmaz.

```
glEnable(GL_STENCIL_TEST);
glStencilMaskSeparate(GL_FRONT, 0xFF);
glClearStencil(0);
glClear(GL_STENCIL_BUFFER_BIT);
glStencilMaskSeparate(GL_FRONT, 0x00);
glDisable(GL_STENCIL_TEST);
```

(glClear)
Bellekte tutulan framebufferı temizlemek için kullanılır. Framebuffer, renk, derinlik ve stencil tamponlarından oluşur. Bu işlev, belirtilen tamponların içeriğini belirtilen değerlere sıfırlar veya doldurur.

(glClearColor)

(glClearDepthf)
(glClearStencil)