Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
Öğr.Gör. Ömer ÇAKIR

BIL3008 Bilgisayar Grafikleri I
Bütünleme, 22.06.2017, 08:00, D1
Süre : **75** Dakika

NUMARA : ..... **AD SOYAD :**	**DEĞERLENDİRME**	
İMZA :	[.....]	.....

Öğrenciler, Mühendislik Fakültesi Sınav Uygulama Yönergesi'ndeki kurallara uymalıdırlar.
Sınav Soruları Bölüm Program Çıktıları'ndan 1,4,12 ile ilişkilidir.

1. Bakış noktası **(0,-26,143)** ve **(0,86,57)** olduğunda **N(0,0.8,0.6)** normaline sahip aşağıdaki üçgenin, arkayüz (backface) olup/olmadığını belirleyiniz. **(30P)**

U_0(0,60,100) U_1(60,0,180) U_2(-60,0,180)

(0,-26,143)-(0,60,100) = (0,-86,43)

(0,-86,43)*(0,0.8,0.6) = -43 (backface)

(0,86,57)- 0,60,100) = (0,26,-43)

(0,26,-43)*(0,0.8,0.6) = -5 (backface)

2. DirectX12'de **swap chain** nesnesinin görevi nedir? **Render()**'da swap chain nesnesinin hangi fonksiyonu koşar? **(20P)**

DirectX 12'de herhangi bir uygulamanın çıktısı görüntülenirken ekran kartındaki 2 bellek alanı kullanılır: **back buffer** ve **front buffer**
back buffer çizim yapılan; front buffer monitörde görüntülenen bellek alanıdır.
back buffer'daki çizim işlemleri (komutları) bittiğinde **swap chain** nesnesinin **Present()** fonksiyonu koşar ve bu bufferlar swap yapılır. Yani back buffer front buffer olur ve içeriği monitörde görüntülenir; front buffer da back buffer olur ve bir sonraki frame'a ait çizim komutları bu buffer için koşar.

3. DirectX12'de **constant buffer** tanımlanırken **.cpp**'deki herhangi bir constant bufferın **.hlsl**'deki eşdeğeri ile ilişkilendirilmesinde hangi setlemelere ihtiyaç vardır? **(20P)**

DirectX12'de 14'e kadar farklı constant buffer tanımlanabilir. Bu constant bufferların .cpp ve .hlsl programlarındaki eşdeğerleri arasındaki bağın kurulması için ilgili constant buffera (cbuffer) .hlsl'de **register(b0)** şeklinde bir etiket verilir. Burada b0'daki 0 değeri buffer için ekran kartındaki registerı (bellek alanını) gösterir. Bu değer 0..13 arası olabilir.

Ekran kartındaki 0. cbufferın kullanılacağını .cpp programında da setlemek için root signature nesnesine **InitAsConstantBufferView()** ile constant buffer, root parametresi olarak eklenirken ilk parametresi olan **shaderRegister** değişkeni de 0'a setlenir.

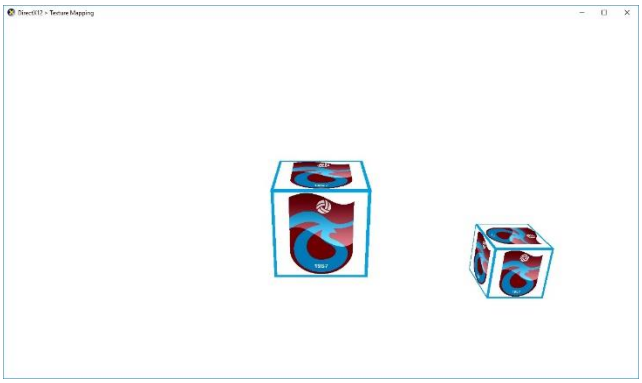
```
mRotate30  = XMMatrixRotationY(XM_PI / 6);          // 30° CW
mRotate45  = XMMatrixRotationY(XM_PI / 4);          // 45° CW
mRotate60  = XMMatrixRotationY(XM_PI / 3);          // 60° CW
mTranslate = XMMatrixTranslation(4.0f, 0.0f, 0.0f);
mScale     = XMMatrixScaling(0.5f, 0.5f, 0.5f);

g_World = mScale * mRotate30 * mTranslate * mRotate60;  // 01
g_World = mScale * mRotate45 * mTranslate * mRotate45;  // 02
g_World = mScale * mRotate60 * mTranslate * mRotate30;  // 03
g_World = mRotate30 * mScale * mTranslate * mRotate60;  // 04
g_World = mRotate45 * mScale * mTranslate * mRotate45;  // 05
g_World = mRotate60 * mScale * mTranslate * mRotate30;  // 06
g_World = mRotate30 * mTranslate * mScale * mRotate60;  // 07
g_World = mRotate45 * mTranslate * mScale * mRotate45;  // 08
g_World = mRotate60 * mTranslate * mScale * mRotate30;  // 09
g_World = mRotate30 * mTranslate * mRotate60  * mScale; // 10
g_World = mRotate45 * mTranslate * mRotate45  * mScale; // 11
g_World = mRotate60 * mTranslate * mRotate30  * mScale; // 12
```

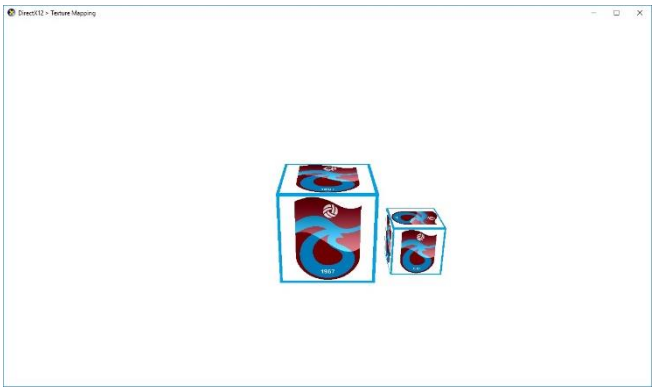
4. Küçük küpün **g_World** matris setlemelerini temsil eden **1-12** arası sayıları ilgili ekran görüntüsünün altına yazınız.
(30P)

Not: Bakış noktası **(0,4,-9)**'dadır. Büyük küpün merkezi **(0,0,0)** noktasındadır ve köşe noktaları **-1,+1** değerleri ile setlenmiştir. Dönme işlemleri saat yönündedir. (**C**lock**W**ise).

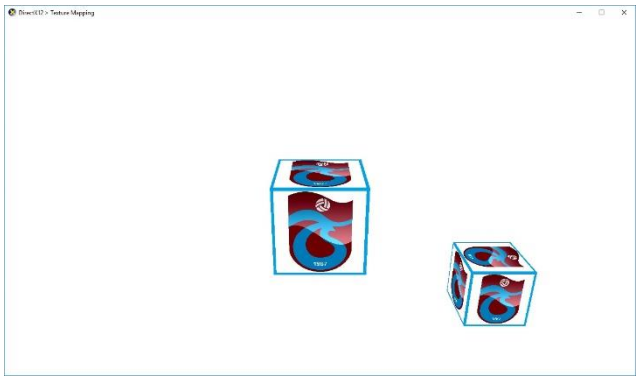
İpucu: Matrislerin hepsi ikişerli gruplar halindedir eşdeğerdir.



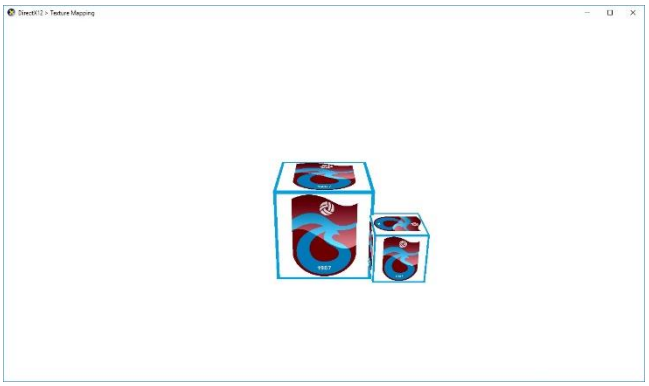
3	6



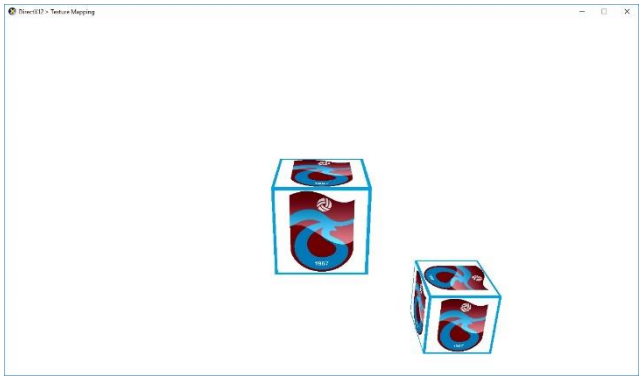
9	12



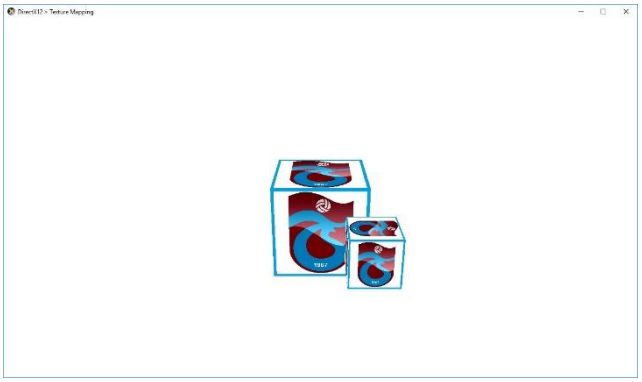
2	5



8	11



1	4



7	10