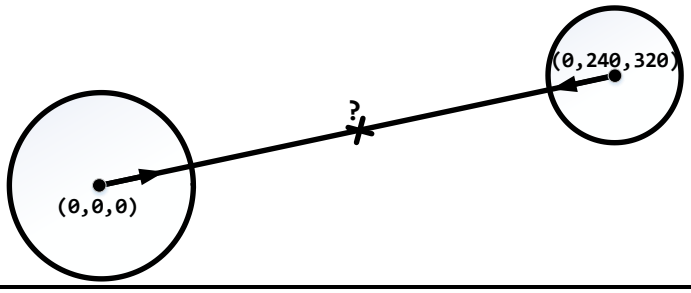
Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
Öğr.Gör. Ömer ÇAKIR

BIL320 Bilgisayar Grafikleri I
Final Sınavı, 31.5.2016, 13:00, D1-2
Süre : **100** Dakika

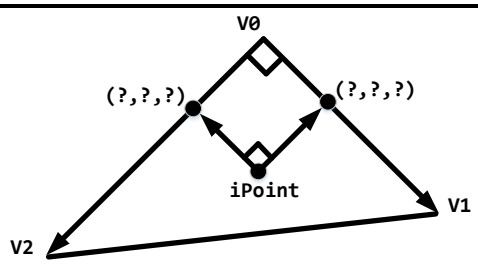
Sınavda Uyulması Gereken Kurallar

1. Cep telefonlarının, hesap makinesi, saate bakmak gibi herhangi bir amaçla kullanılması yasaktır. Telefon **kapalı** ve **cepte** olmalıdır.
2. Sınavın başında öğrenciler anlamadıkları noktaları **sesli olarak sorup**, Hoca cevapladıktan sonra **sınav boyunca soru sormak yasaktır**.

NUMARA :	AD SOYAD :	İMZA :	DEĞERLENDİRME :



1. Yarıçapı r_M=50br (birim) merkezi **c_M(0,0,0)** noktasındaki mavi renkli bir küre ile yarıçapı r_K=30br merkezi **c_K(0,240,320)** noktasındaki kırmızı renkli küre merkezlerinden geçen doğru boyunca birbirlerine sırasıyla **v_M=5br/sn**, **v_K=3br/sn** hızlarla yaklaşıyor ve çarpışıyorlar. Çarpışma noktasının koordinatlarını hesaplayınız. **(20P)**



V0(0,12,60) V1(18,-12,60) V2(0,12,20)

2. Yukarıda köşe noktaları verilen **V** dik üçgeni üzerindeki **iPoint(9,0,48)** noktasının **E_1=V1-V0** ve **E_2=V2-V0** kenarlarına dik izdüşüm koordinatlarını bulunuz. **(20P)**

$R_1 x R_2 = (R_{1y}R_{2z}-R_{1z}R_{2y},\ R_{1z}R_{2x}-R_{1x}R_{2z},\ R_{1x}R_{2y}-R_{1y}R_{2x})$

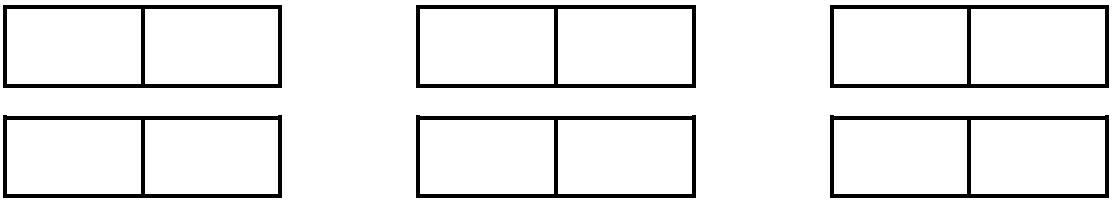
```
mRotate30  = XMMatrixRotationY(XM_PI / 6);          // 30° CW
mRotate60  = XMMatrixRotationY(XM_PI / 3);          // 60° CW
mTranslate = XMMatrixTranslation(4.0f, 0.0f, 0.0f);
mScale     = XMMatrixScaling(0.5f, 0.5f, 0.5f);

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

3. Küçük küpün **g_World** matris setlemelerinden eşdeğer olanların satır numarlarını kutucuklara 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: Eşdeğer matrislerin hepsi ikişerli gruplar halindedir.



```
[01]  bool IntersectTriangle(XMVECTOR Ro, XMVECTOR Rd,
[02]                SimpleVertex* verticesModel, int vertexCount)
[03]  {
[04]     float t = 0.0, s, s1, s2, s3;
[05]     XMVECTOR normal, R, S, S1, S2, S3;
[06]
[07]     for (int i = 0; i < vertexCount; i += 3)
[08]     {
[09]
[10]        XMVECTOR v0 = XMLoadFloat3(&verticesModel[i    ].Pos);
[11]        XMVECTOR v1 = XMLoadFloat3(&verticesModel[i + 1].Pos);
[12]        XMVECTOR v2 = XMLoadFloat3(&verticesModel[i + 2].Pos);
[13]
[14]        normal  = XMVector3Cross((v1 - v0), (v2 - v0));
[15]        float D = -XMVectorGetX(XMVector3Dot(v0, normal));
[16]        t = -(XMVectorGetX(XMVector3Dot(Ro, normal)) + D)
[17]           / XMVectorGetX(XMVector3Dot(normal, Rd));
[18]
[19]        if (t > 0)
[20]        {
[21]           R = Ro + t * Rd;
[22]
[23]           S  = XMVector3Cross((v1 - v0), (v2 - v0));
[24]           S1 = XMVector3Cross((R  - v0), (v2 - v0));
[25]           S2 = XMVector3Cross((v1 - v0), (R  - v0));
[26]           S3 = XMVector3Cross((v1 -  R), (v2 -  R));
[27]
[28]           s  = XMVectorGetX(XMVector3Length (S));
[29]           s1 = XMVectorGetX(XMVector3Length(S1));
[30]           s2 = XMVectorGetX(XMVector3Length(S2));
[31]           s3 = XMVectorGetX(XMVector3Length(S3));
[32]
[33]           float fark = (float)::abs(s - (s1 + s2 + s3));
[34]           float epsilon = 0.0001F;
[35]           if (fark <= epsilon) return true;
[36]        }
[37]     }
[38]
[39]     return false;
[40]  }
```

4. **IntersectTriangle()** başlangıç noktası (**Ro**) ve doğrultusu (**Rd**) verilen ışınla, üçgenlerinin köşe noktalarının koordinatları **verticesModel**, köşe noktalarının sayısı **vertexCount** ile temsil edilen cisim arasında kesişim testi yapar. Kesişim varsa **true** yoksa **false** döndürür.

Fonksiyonunun bir Tank Savaşı oyununda, ateş edildiğinde düşmanın vurulup vurulmadığının test edilmesinde kullanıldığı varsayılsın. Düşman sabitken fonksiyon isabet testini doğru yapıyor yalnız düşman hareketli olduğunda çalışmıyor. Fonksiyonun hareketli cisimler için de kullanılabilmesine yönelik olarak hangi satır(lar), nasıl güncellenmelidir? **(30P)**