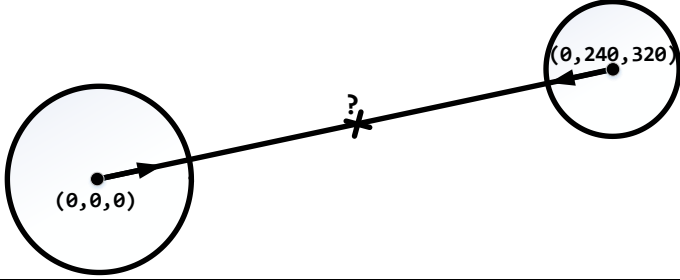




CEVAPLAR



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 çarpşıyorlar. Çarpşıma noktasının koordinatlarını hesaplayınız. (20P)

$$(0,0,0) + 50*(0,0.6,0.8) + 5t*(0,0.6,0.8) = (0,240,320) + 30*(0,-0.6,-0.8) + 3t*(0,-0.6,-0.8)$$

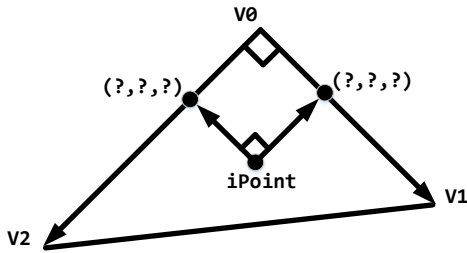
$$t(0,4.8,6.4) = (0,192,256)$$

$$t = 40 \text{ sn}$$

$$iPoint = (0,0,0) + 50*(0,0.6,0.8) + 5*40*(0,0.6,0.8) = (0,150,200)$$

veya

$$iPoint = (0,240,320) + 30*(0,-0.6,-0.8) + 3*40*(0,-0.6,-0.8) = (0,150,200)$$



$$V_0(0,12,60) \quad V_1(18,-12,60) \quad V_2(0,12,20)$$

2. Yukarıda köşe noktaları verilen V dik üçgeni üzerindeki $iPoint(9,0,48)$ noktasının $E_1=V_1-V_0$ ve $E_2=V_2-V_0$ kenarlarına dik izdüşüm koordinatlarını bulunuz. (20P)

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

$$iPoint_{E_1} = (9, 0, 60)$$

$$iPoint_{E_2} = (0, 12, 48)$$

İlan edilen programa bakınız.

```

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

```

1	5
4	6

2	8
7	11

3	9
10	12

```

[01] bool IntersectTriangle(XMVECTOR Ro, XMVECTOR Rd,
[02] SimpleVertex* verticesModel, int vertexCount, XMATRIX& g_World)
[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]         v0 = XMVector3Transform(v0, g_World);
[15]         v1 = XMVector3Transform(v1, g_World);
[16]         v2 = XMVector3Transform(v2, g_World);
[17]
[18]         normal = XMVector3Cross((v1 - v0), (v2 - v0));
[19]         float D = -XMVectorGetX(XMVector3Dot(v0, normal));
[20]         t = -(XMVectorGetX(XMVector3Dot(Ro, normal)) + D)
[21]             / XMVectorGetX(XMVector3Dot(normal, Rd));
[22]
[23]         if (t > 0)
[24]         {
[25]             R = Ro + t * Rd;
[26]
[27]             S = XMVector3Cross((v1 - v0), (v2 - v0));
[28]             S1 = XMVector3Cross((R - v0), (v2 - v0));
[29]             S2 = XMVector3Cross((v1 - v0), (R - v0));
[30]             S3 = XMVector3Cross((v1 - R), (v2 - R));
[31]
[32]             s = XMVectorGetX(XMVector3Length(S));
[33]             s1 = XMVectorGetX(XMVector3Length(S1));
[34]             s2 = XMVectorGetX(XMVector3Length(S2));
[35]             s3 = XMVectorGetX(XMVector3Length(S3));
[36]
[37]             float fark = (float)::abs(s - (s1 + s2 + s3));
[38]             float epsilon = 0.0001f;
[39]             if (fark <= epsilon) return true;
[40]         }
[41]     }
[42]
[43]     return false;
[44] }

```

3. Küçük küpün **g_World** matris setlemelerinden eşdeğer olanların satır numaralarını kutucuklara yazınız. (30P)

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

İpucu: Eşdeğer matrislerin hepsi ikişerli gruplar halindedir.

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 varsayalım. 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)