IŞIN İZLEME (RAY TRACING)

1. Giriş

Işın İzleme "Ray Tracing", bakış noktasından çıkan ve görüntü düzlemindeki piksellerin herbirinden geçen ışınlar ile 3D ortamdaki cisimler arasında yapılan kesişim testleri sonucu belirlenen en yakın cismin (eğer bu cisim ışını yansıtma özelliğine sahipse veya saydam olup ışın içinden doğrudan veya kırılarak geçiyorsa ondan yansıyıp/kırılıp çarptığı ilk cismin rengi de hesaba katılarak son) renginin bir boyama modeli (örneğin Phong boyama modeli) ile belirlenmesine dayanan 3D bilgisayar grafiği üretme yöntemidir.

Işının **Ro** başlangıç noktası, **Rd** doğrultsu ve `t` skaler uzaklık değerine bağlı parametrik ifadesini tanıtmadan önce, sıkça kullanılacak vektörel işlemler hakkında ön bilgiler verilecektir.

2. Vektörel İşlemlerle İlgili Temeller

Herhangi bir vektörün boyunu bulmak için (x,y,z) koordinatlarının karelerinin toplamının karekökü alınır. Örneğin **R=(0,6,8)** vektörünün boyu **|R|** = $\sqrt{0^2 + 6^2 + 8^2} = 10$.

Herhangi bir vektörün boyunu **1** birim yapma işlemine "normalizasyon" denir ve işlem için **(x,y,z)** koordinat bileşenlerinin her biri vektörün boyuna bölünür. Yukarıdaki **(0,6,8)** vektörü normalize edildiğinde $\left(\frac{0}{10}, \frac{6}{10}, \frac{8}{10}\right) = (\mathbf{0}, \mathbf{0.6}, \mathbf{0.8})$ bulunur. Normalize edilmiş vektörün boyu hesaplandığında $\sqrt{(0)^2 + (0.6)^2 + (0.8)^2} = 1$ olduğu görülür. Normalize edilmiş vektöre "birim vektör" denir. (Vektörel büyüklükler **koyu**, skaler büyüklükler normal font ile yazılacaktır).

Vektörler arasında skaler ve vektörel olmak üzere iki temel çarpım işlemi vardır. $\mathbf{R_1}$ ve $\mathbf{R_2}$ gibi iki vektörün sırasıyla skaler ve vektörel çarpımları aşağıdaki gibi yapılır:

$$\mathbf{R_1} * \mathbf{R_2} = \mathbf{R_{1x}R_{2x}} + \mathbf{R_{1y}R_{2y}} + \mathbf{R_{1z}R_{2z}} = |\mathbf{R1}|*|\mathbf{R2}|*\mathrm{Cos}(\beta)$$

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

İki vektörün skaler çarpımında **(x,y,z)** koordinatları ayrı ayrı çarpılıp toplanır veya vektörlerin boylarının aralarındaki açının kosinüsüyle çarpımı olarak da hesaplanabilir. Dolayısıyla birim vektörlerin skaler çarpımı aralarındaki açının kosinüsünü verir. Skaler çarpım Phong boyama modelinin diffuse ve specular renk bileşenlerinin hesaplanmasında kullanılır.

$\mathbf{R_1}$ ve $\mathbf{R_2}$ 'nin vektörel çarpımıyla onlara dik olan bir vektör elde edilir. Köşe noktalarının koordinatları **V0,V1,V2** şeklinde verilen bir üçgenin yüzeyine dik olan vektör yani başka bir deyişle üçgenin **N** normal vektörü **(V1-V0)** ve **(V2-V0)** fark vektörlerinin vektörel çarpımı ile hesaplanır. Normal, üçgenin hangi yüzeyine dik olur? Bu sorunun cevabı, **(X,Y,Z)** kartezyen koordinat eksenleri (özellikle **+Z** ekseni) ve köşe noktalarının sırasına (saat yönünde – ClockWise (CW), saat yönünün tersi – Counter ClockWise (CCW)) göre belirler.

(0,0,0) orjin noktasına bağlı olarak **+X** ekseni sağa, **+Y** ekseni yukarı ve **+Z** ekseni ileri doğru olan kartezyen koordinat sisteminde herhangi bir üçgenin normali köşe noktalarının sırasının saat yönünde – ClockWise (CW) olduğu yüzeyine dik olur. Örnek **V0,V1,V2** köşe noktaları **V0=(0,40,120), V1=(30,-40,60), V2=(-30,-40,60)** için **N** normali şöyle hesaplanır:

(V1-V0)=(30, -80, -60) (V2-V0)=(-30, -80, -60)

N = (V1-V0)x(V2-V0)=(-80*-60 - -60*-80, -60*-30 - 30*-60, 30*-80 - -80*-30)

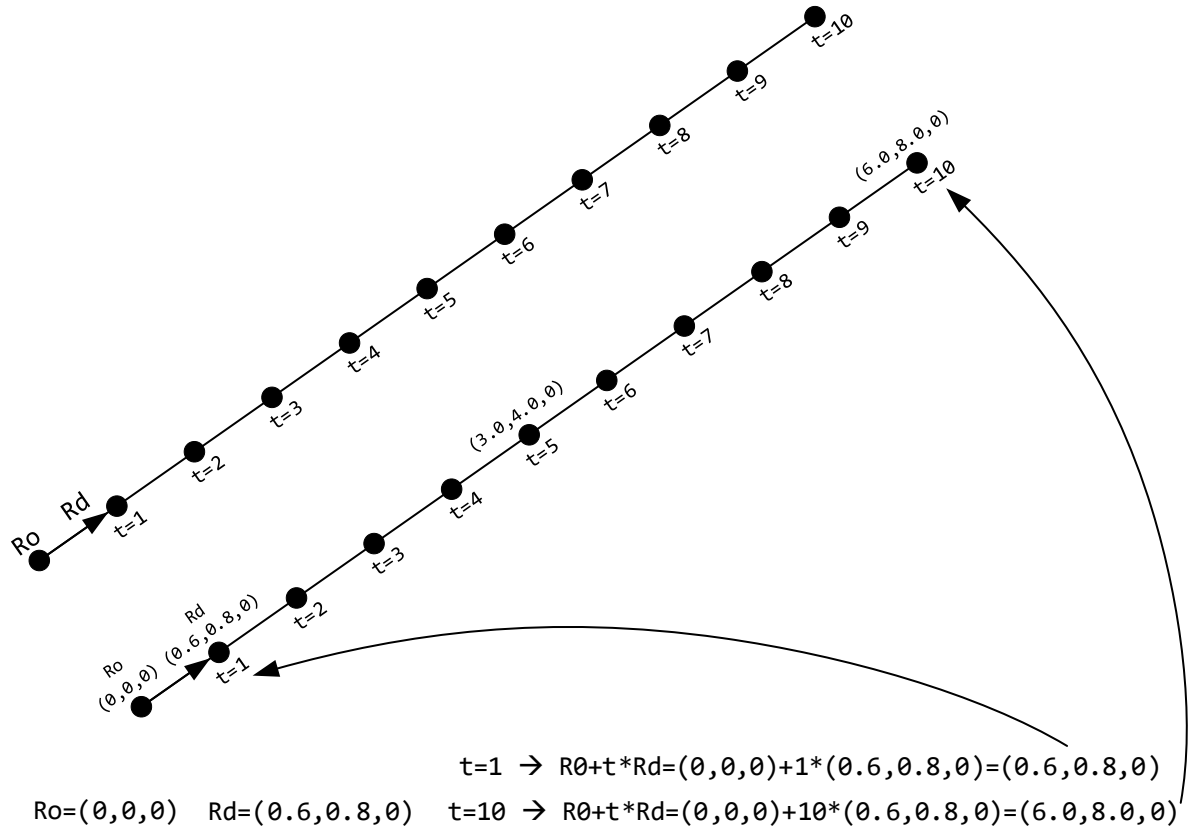
N = (0, 3600, -4800)

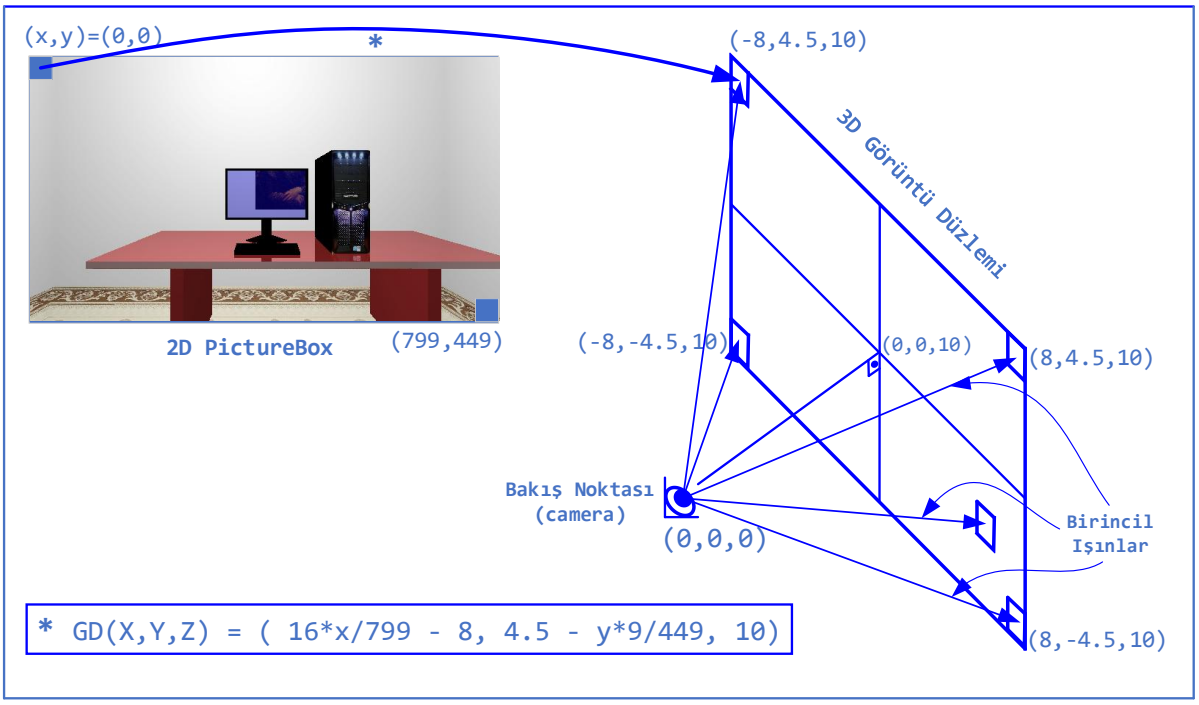
N vektörü normalize edilirse **(0,0.6,-0.8)** elde edilir.

3. Işının Tanımı

$\mathbf{R} = \mathbf{R_o} + t*\mathbf{R_d}$

R ışınına ait ifade yukarıda verilmiştir. Burada $\mathbf{R_o}$ ışının başlangıç noktası (**o**rigin), $\mathbf{R_d}$ doğrultusu (**d**irection), t ise ışının $\mathbf{R_d}$ doğrultusu boyunca t birim sonraki konumunu hesaplarken kullanılan skaler bir değerdir. Işın İzleme'de temelde, ışın ile 3D cisimler arasında kesişim testleri yapılır. Işın herhangi bir cisim ile kesişiyorsa, kesişim testi algoritması aradaki t uzaklığını döndürür. Bizim ışın izleme ile çizeceğimiz 3D cisimler, üçgenler ve küreler olacaktır. Dolayısıyla kesişim testi olarak Işın-Üçgen (4. sayfada) ve Işın-Küre (7. sayfada) kesişim testi algoritmalarını tanıtacağız.





Şekil 1: 2D Picture Box ve 3D Görüntü Düzlemi Arasındaki İlişki

Işın İzleme uygulaması pratikte bir Visual C++ Form Application'dır. Üretilecek 3D görüntü, Form'un üzerindeki 2D Picture Box'a çizilecektir. 2D Picture Box'taki herbir piksel için ışın izlemeye dair hesaplamalar 3D uzayda yapılmaktadır. O yüzden 2D Picture Box için 3D uzayda aynı çözünürlükte sanal bir Picture Box daha vardır. Buna **3D Görüntü Düzlemi** diyeceğiz. 3D ortamı gözlemleyen ve konumu **(0,0,0)** olarak setlenmiş **Bakış Noktası**, 3D Görüntü Düzleminin merkezine dik belli bir mesafe (10 birim) gerisine konumlandırılmıştır. Işın İzleme adımları şöyledir:

- Bakış Noktasından ışınlar yollanır. 3D Görüntü Düzlemindeki ilgili piksellerden geçen bu ışınlara "**Birincil Işınlar**" denir.
- Birincil Işınlarla 3D uzaydaki cisimler arasında kesişim testleri yapılır ve `t` uzaklıkları hesaplanır.
- Eğer herhangi bir birincil ışın 1'den fazla cisimle kesişmişse `t` uzaklığı en düşük olan cisim belirlenir. (Burada cisimlerin saydam olmadıkları varsayılmıştır)
- `t` uzaklığı en düşük cismin rengi, kesişim noktasının normali, bakış noktasının konumu, ışık kaynağının konumu ve rengi gibi parametrelere bağlı olarak birincil ışının geçtiği pikselin rengi hesaplanır.

Birincil Işınlar üretilirken öncelikle onların $\mathbf{R_d}$ doğrultuları belirlenir. Doğrultuların hesaplanabilmesi için iki noktaya ihtiyaç vardır. Bunlar $\mathbf{R_1}$ ve $\mathbf{R_2}$ olarak alınırsa $\mathbf{R_1}$'den $\mathbf{R_2}$'ye doğru olan doğrultu vektörü $\mathbf{R_d=R_2-R_1}$ ile hesaplanır. Benzer şekilde başlangıç noktasından ($\mathbf{R_1}$) çıkıp piksel koordinatlarından ($\mathbf{R_2}$) geçen ışının $\mathbf{R_d}$ doğrultusu da $\mathbf{R_2-R_1}$ ile yani piksel koordinatlarından başlangıç noktasının koordinatı çıkarılarak hesaplanır. Işın izlemede doğrultu vektörleri birim vektör olmalıdır. Dolayısıyla $\mathbf{R_d}$ normalize edilerek boyu **1** birim yapılır. Şekil 1'den görüldüğü gibi birincil ışınların 3D görüntü düzleminden geçtikleri pikseller ile 2D Picture Box'taki pikseller farklı koordinat sistemlerine sahiptir. Örneğin Picture Box'taki `800x450` çözünürlükteki bir görüntünün `(x,y)` koordinatları sol üst köşede `(0,0)` sağ alt köşede de `(799,449)`'dur. Dolayısıyla hep pozitif tam sayı (integer) değerler alırlar. 3D görüntü düzlemindeki `(x,y,z)` piksel koordinatları pozitif veya negatif değerler alabilen floating point sayılardır.

Işın İzleme uygulamasının pratikte bir Visual C++ Form Application olduğunu söylemiştik. Form'un üzerindeki örneğin 800x450 çözünürlükteki 2D Picture Box'ın herhangi bir **(x,y)** 2D piksel koordinatının 3D Görüntü Düzlemindeki **GD(X,Y,Z)** eşdeğeri şu ifade ile hesaplanır:

```
GD(X,Y,Z) = ( 16*x/799 - 8, 4.5 - y*9/449, 10)
```

Burada 3D Görüntü Düzlemi 16x9 birim boyunda seçilmiştir. Bakış Noktasının, 3D Görüntü Düzleminin merkezine dik uzaklığı 10 birimdir.

4. Işın-Üçgen Kesişim Testi

3D cisimler çoğunlukla üçgenler ile temsil edilirler. Burada anlatılacak olan ışın-üçgen kesişim testi iki aşamadan oluşmaktadır:

1. Işın ile üçgenin üzerinde olduğu düzlemsel yüzey arasında kesişim testi.
2. Işın yüzey ile kesişiyorsa kesişim noktasının üçgenin içinde olup olmadığını belirleme.

Birinci aşama için üçgenin tanımladığı yüzeyin denklemini çıkarmak gerekmedir. Bilindiği gibi yüzey denklemi Ax+By+Cz+D=0 'dır. Burada **(A,B,C)** yüzey normalidir. Yukarıda **V0,V1,V2** şeklinde verilen üçgenin yüzey denklemini çıkaralım. Daha önce üçgenin normali **N=(0,0.6,-0.8)** olarak hesaplanmıştı. Dolayısıyla **N=(A,B,C)** biliniyor. Yüzeyin üzerinde olduğu için yüzey denklemini sağlayacağından üçgenin köşe noktalarından herhangi biri D 'nin hesabı için kullanılabilir. Dolayısıyla Ax+By+Cz+D=0 'daki **(x,y,z)** yerine köşe noktalarından herhangi birinin mesela **V0** 'ın **(x,y,z)** 'sini yazıp sıfıra eşitlersek D 'yi :

```
Ax + By + Cz + D              = 0
0*0 + 0.6*40 + -0.8*120 + D   = 0
                          D = 72
```

olarak buluruz. Dolayısıyla yüzey denklemi 0.6y - 0.8z + 72 = 0 'dır. Işın yüzey ile kesişiyorsa üçgenin köşe noktaları gibi ışının yüzey üzerindeki koordinatları da yüzey denklemini sağlamalıdır. Dolayısıyla şöyle yazabiliriz:

$$A(R_{ox}+tR_{dx}) + B(R_{oy}+tR_{dy}) + C(R_{oz}+tR_{dz}) + D = 0$$

Yukarıdaki denklem t 'ye göre düzenlenirse:

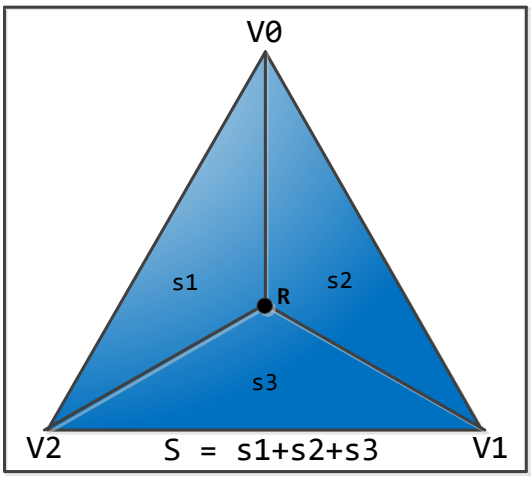
$$t = -\frac{AR_{ox} + BR_{oy} + CR_{oz} + D}{AR_{dx} + BR_{dy} + CR_{dz}} = -\frac{\boldsymbol{N} * \boldsymbol{R_o} + D}{\boldsymbol{N} * \boldsymbol{R_d}}$$

t>0 ise ışın yüzey ile kesişiyor demektir. t<0 ise ışın yüzey ile kesişmiyor demektir.
t=∞ ise yani $N * R_d = 0$ ise ışın yüzeye paralel demektir.

Örnek olarak $\mathbf{R_o=(0,0,0)}$ başlangıç noktasından $\mathbf{R_d=(0,0,1)}$ doğrultusu boyunca giden **R** ışınının bu yüzey ile kesişip kesişmediğini t hesabı ile belirleyelim:

$$t = -\frac{\boldsymbol{N} * \boldsymbol{R_o} + D}{\boldsymbol{N} * \boldsymbol{R_d}} = -\frac{72}{-0.8} = 90$$

Işının yüzey üzerindeki koordinatları $\mathbf{R=R_o+tR_d=(0,0,0)+}90\mathbf{(0,0,1)=(0,0,90)}$ olarak bulunur. Böylece kesişim testinin I. Aşaması tamamlanmış oldu.



Şekil 2: Alan Testi

II. aşamada **(0,0,90)** noktasının üçgenin içinde olup olmadığına karar verilmelidir. Bunun için değişik yöntemler denenebilir. Burada "alan testi" yöntemi kullanılacaktır. Buna göre Şekil 2'den de görüldüğü gibi yukarıda hesaplanan kesişim noktasından üçgenin köşelerine doğrular çizerek 3 alt üçgen oluşturulur. Bu alt üçgenlerin alanları toplamı büyük üçgenin alanına eşitse kesişim noktası üçgenin içinde demektir.

V0,V1,V2 üçgeninin alanı **|(V1-V0)x(V2-V1)|** ile hesaplanabilir. **R=(0,0,90)** noktası kullanılarak oluşturulan s1, s2 ve s3 alt üçgenlerin alanları ve **V0,V1,V2** üçgeninin S alanını hesaplandığında S=6000, s1=1500, s2=1500 ve s3=3000 çıkar. S=s1+s2+s3 olduğundan kesişim noktası üçgenin içindedir. Kesişim testi fonksiyonu `Intersect()`'in C++ kodu aşağıdadır:

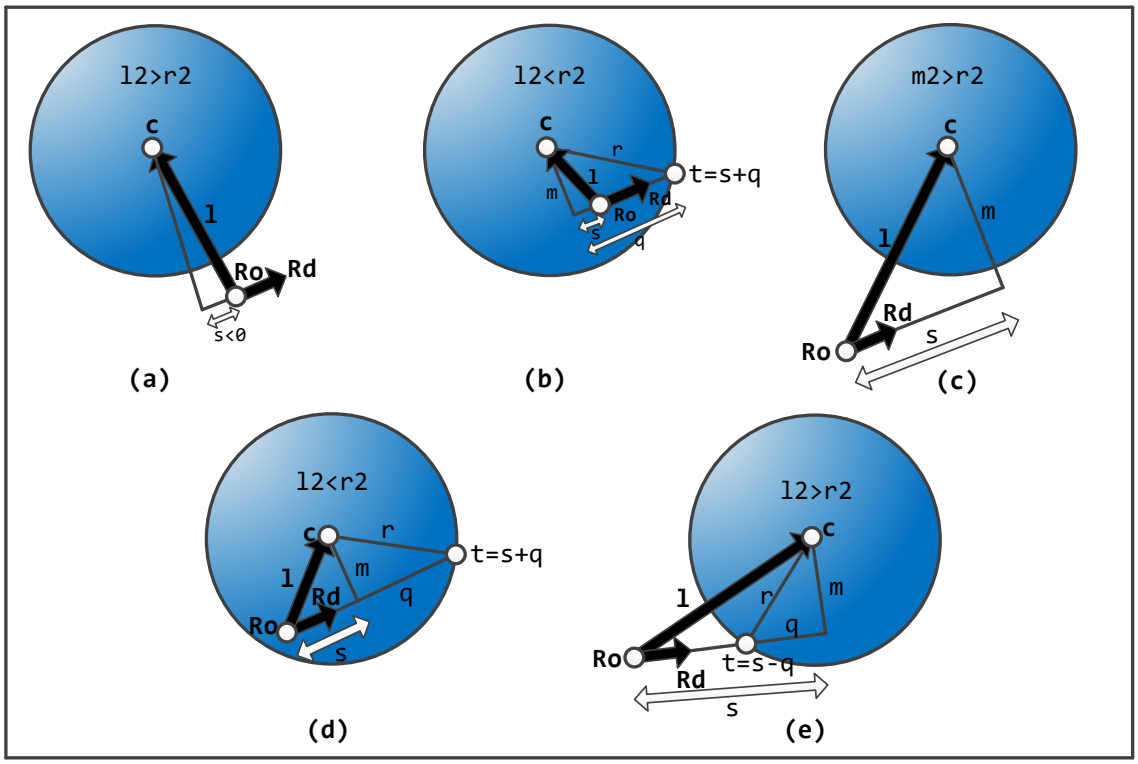
```
float Intersect(Vertex Ro, Vertex Rd)
{
	Vertex normal, R;
	float S, s1, s2, s3;

	normal  = (V1-V0).CrossProduct(V2-V0);
	float D = -(V0 * normal);
	float t = -(Ro * normal + D) / (normal * Rd);

	if (t > 0)
	{
		R = Ro + t * Rd;

		S  = (V1 - V0).CrossProduct(V2 - V0).Length();
		s1 = ( R - V0).CrossProduct(V2 - V0).Length();
		s2 = (V1 - V0).CrossProduct( R - V0).Length();
		s3 = (V1 -  R).CrossProduct(V2 -  R).Length();

		float fark = (float)Math::Abs(S - (s1 + s2 + s3));
		float epsilon = 0.005F;

		if (fark <= epsilon) return t; else return 0;
	}
	else return 0;
}
```

5. Işın-Üçgen Kesişim Testi Örneği

Görüntü düzleminden geçen ışınlardan herhangi biri gittiği doğrultu boyunca 1'den fazla cisimle kesişiyorsa ekrana bunlardan görüntü düzlemine en yakın olanı çizilmelidir. Bunun için kesişim testleri ile hesaplanan `t` uzaklıkları sıralanır ve en küçük `t` uzaklığına sahip cismin görüntüsü çizilir. Diğer cisimler görünmeyen yüzey olarak adlandırılırlar.

U0(0, 30, 40) **U1(40, -30, 120)** **U2 (-40, -30, 120)**
Y0(-50, 30, 124) **Y1(50, 30, 124)** **Y2 (0, -30, 44)**
Z0(-30, 0, 37) **Z1(30, 40, 117)** **Z2 (30, -40, 117)**

Yukarıda sırasıyla kırmızı, yeşil ve mavi renklere sahip **U0,U1,U2** üçgeni **Y0,Y1,Y2** üçgeni ve **Z0,Z1,Z2** üçgeninin köşe noktalarının koordinatları verilmiştir. Başlangıç noktası $\mathbf{R_o=(0,0,0)}$ 'dan çıkan ve görüntü düzleminde **(0,0,10)** noktasındaki pikselden geçen ışın ile bu üçgenler arasında ışın-üçgen kesişim testi yöntemine göre $t_U=80$, $t_V=84$, $t_Z=77$ uzaklık değerleri hesaplanır. Uzaklıklar sıralandığında mavi renkli Z üçgeninin bakış noktasına daha yakın olduğu görülür. Dolayısıyla ışının geçtiği piksel mavi renge boyanır. Kırmızı renkli U ve yeşil renkli Y üçgenleri görüntü düzleminde **(0,0,10)** noktasındaki pikselden görünmeyen üçgenlerdir.



Şekil 3: Işın-Küre Kesişim Testi

6. Işın-Küre Kesişim Testi

Kesişim testine ışının başlangıç noktası $\mathbf{R_o}$'dan kürenin merkezine $\mathbf{l=c-R_o}$ vektörü hesaplanarak başlanır. Sonra üç tane skaler değer $\mathbf{s=l*R_d}$, $\mathbf{l2=l*l}$ ve $\mathbf{r2=r*r}$ şeklinde hesaplanır. Burada $\mathbf{s}$, $\mathbf{l}$ vektörünün boyunun $\mathbf{R_d}$ üzerine izdüşümüdür. Çünkü $\mathbf{R_d}$ birim vektör olduğundan $\mathbf{l*R_d}$ aynı zamanda $\mathbf{|l|*Cos(\beta)}$ demektir. $\mathbf{l2}$, $\mathbf{l}$ vektörünün r2 de yarıçapın boyunun karesidir. Eğer `(s<0 && l2>r2)` şartı sağlanıyorsa Şekil 3-(a)'daki durum söz konusudur yani kesişim yok demektir. `s<0` olsa bile `l2<r2` ise yani ışının başlangıç noktası kürenin içinde ise Şekil 3-(b) 'deki gibi her zaman kesişim vardır. `s>0` iken hangi durumlarda kesişim olduğunu belirlemek üzere `m2=l2-s2` hesaplanır. `m2>r2` ise Şekil 3-(c)'deki gibi kesişim yoktur. `m2<=r2` ise kesişim vardır ve `l2<r2` ise ışın Şekil 3-(d)'deki gibi içten, `l2>r2` ise de Şekil 3-(e)'deki gibi dıştan küreyle kesişiyor demektir. `q=sqrt(r2-m2)` hesaplanır. Dıştan kesiştiğinde ışının küreye uzaklığı `t=s-q` ile içten kesiştiğinde `t=s+q` ile hesaplanır. Işın-küre kesişim fonksiyonu `Intersect()` aşağıda verilmiştir:

```
float Intersect(Vertex Ro, Vertex Rd)
{
    Vertex l = Center - Ro;
    float  s = l * Rd;
    float l2 = l * l;
    float r2 = Radius * Radius;
    if (s < 0 && l2 > r2) return 0;
    float s2 = s * s;
    float m2 = l2 - s2;
    if (m2 > r2) return 0;
    float  q = (float)Math::Sqrt(r2 - m2);
    if (l2 > r2) return s - q;
    else return s + q;
}
```

7. Phong Boyama Modeli

Işınlar ile üçgenler/kürelerle kesişim testleri yapılıp görüntü düzlemine en yakın olan üçgen/küre belirlendiğinde bu üçgenin/kürenin rengi doğrudan ilgili piksele setlenmez. Çünkü ışık kaynağının o noktayı ne oranda aydınlattığı dikkate alınmalıdır. Phong boyama modeline göre ışık kaynağına bağlı olarak ambient, diffuse ve specular olmak üzere üç renk bileşeni, toplamları 1 olan katsayılarla çarpılıp toplanarak pikselin son rengi belirlenir.

7.1. Diffuse Renk Bileşeni

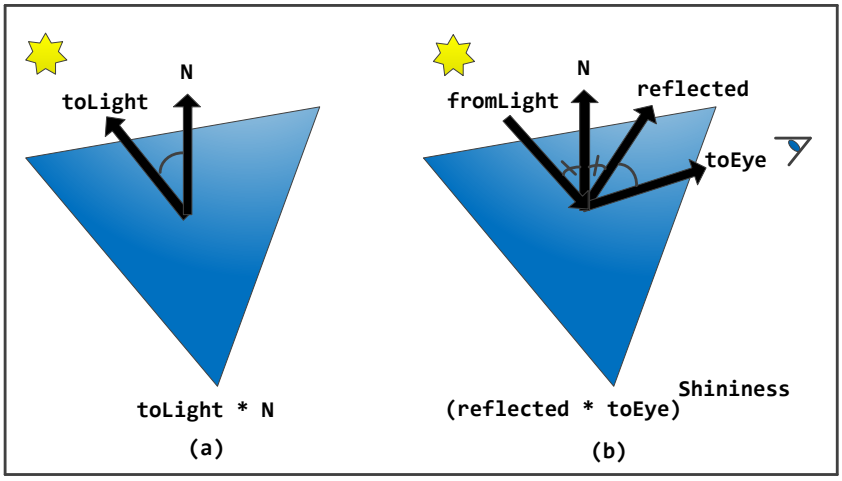
Diffuse katsayı Şekil 4-(a)'daki gibi yüzey normali ile ışık kaynağına doğru olan vektör skaler çarpılarak belirlenir. Her ikisinin de birim vektör olduğu varsayıldığında skaler çarpımları aralarındaki açının kosinüsünü verir. Negatif değerler için yüzey ışık kaynağı tarafından aydınlatılmıyor demektir. Işık kaynağı yüzeye tam dik vuruyorsa yani yüzey normali ile aralarındaki açı 0 derece ise `Cos(0)=1` olduğundan maksimum aydınlatma; açı 90 derece ise de `Cos(90)=0` minimum (sıfır) aydınlatma söz konusudur. Hesaplanan diffuse katsayı ile yüzeyin rengi çarpılarak Phong boyama modelinin diffuse renk bileşeni belirlenir.

Işık kaynağının koordinatları **`(0,40,0)`** olduğu durumda yüzey normali **`(0,1,0)`** olan kırmızı renkli bir üçgen üzerindeki **`(0,0,30)`** noktasının diffuse renk bileşenini hesaplayalım:

Işık kaynağına doğru olan vektör **`(0,40,0)-(0,0,30)=(0,40,-30)`**'dur. Normalize edildiğinde **`(0,0.8,-0.6)`** olur. Diffuse katsayı **`(0,0.8,-0.6)*(0,1,0)`**`=0.8` çıkar. Kırmızı renk `(255,0,0)`, hesaplanan diffuse katsayı ile çarpılırsa diffuse renk bileşeni `(204,0,0)` olarak bulunur.

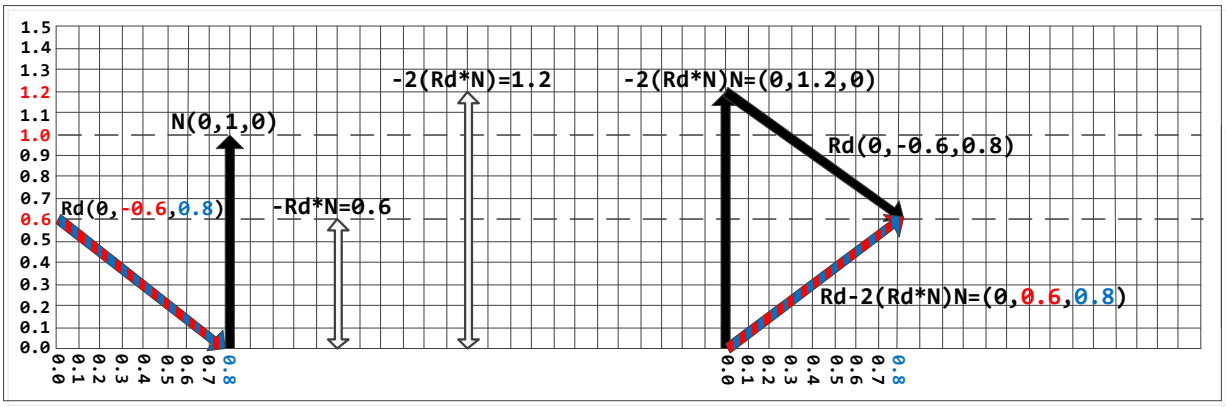
7.2. Specular Renk Bileşeni

Specular katsayı Şekil 4-(b)'deki gibi ışık kaynağından yüzeye doğru olan vektörün yüzey normalinden yansıma vektörü ile bakış noktasına doğru olan vektör arasındaki açının belli bir skaler değer kadar kuvveti alınarak hesaplanır. Hesaplanan specular katsayı ile ışık kaynağının rengi çarpılarak Phong boyama modelinin specular renk bileşeni belirlenir.



Şekil 4: Diffuse ve Specular Katsayı Hesabı

Şekil 4-(b)'deki specular katsayı hesabındaki **`reflected`** yansıma vektörünün hesaplanışı Şekil 5'te gösterilmiştir. Burada gelen ışının doğrultusu $\mathbf{R_d}$, yüzey normali **N**, olduğu durumda yansıma vektörü $\mathbf{R_d - 2R_d * N * N}$ ile hesaplanır.



Şekil 5: Yansıma Vektörü Hesabı

7.3. Ambient Renk Bileşeni

Bir odada masa sehpa gibi eşyaların altları gibi ışık kaynağı tarafından doğrudan aydınlatılmayan yüzeyler vardır. Buralara duvarlardan veya diğer eşyalardan yansıyan ışıkların vurması nedeniyle az da olsa aydınlatılmaktadırlar. Phong modelinin ambient renk bileşeni bu dolaylı aydınlatmayı modeller ve cismin kendi rengi `0..1` arası bir katsayı ile çarpılarak ambient renk bileşeni hesaplanır.

8. Yansıma (Reflection), Saydamlık (Transparency) ve Kırılma (Refraction)

Işını yansıtan yüzeyler için yansıyan ışının doğrultusunun (reflected direction) $\mathbf{R_d-2R_d*N*N}$ ile hesaplanacağından specular bileşene ait **`reflected`** vektörü hesaplanırken bahsedilmişti. Işını yansıtma özelliğine sahip cisim için Şekil-5'teki gibi yansıma doğrultusu hesaplandıktan sonra yeni doğrultu boyunca yollanan ışın ile cisimler üzerinde kesişim testleri yapılarak en yakın cisim belirlenir. Onun rengi o..1 arası bir katsayı ile çarpılarak yansıma ile görülen cismin rengi hesaplanır.

Saydam cisimlerde ışın cismin içinden geçerken doğrultusu değişmez. Yani cismin içinden geçen ışının doğrultusu (transmitted direction) gelen ışının doğrultusu $\mathbf{R_d}$ ile aynı alınıp yukarıdaki işlemler tekrarlanır.

Işın saydam bir küreden geçerken (ışının başlangış noktasının kürenin dışında olduğu varsayıldığından) küreyle önce dıştan sonra da içten kesişerek yoluna devam eder. Işın küreyle dıştan kesiştikten sonra aynı doğrultu boyunca içten kesişecek şekilde tekrar yollandığında ışın-küre kesişim testi `t` uzaklığı hesabında:

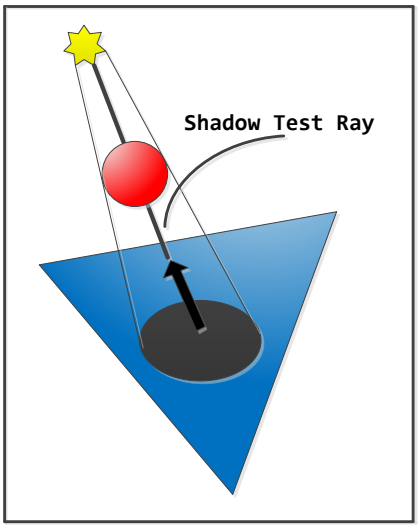
```
if (l2 > r2) return s - q;
else return s + q;
```

ışının başlangıç noktası küre üzerinde yani `l2 = r2` olduğundan normalde `else` kısmının koşması beklenir ama ışın-üçgen alan testindeki gibi `l2` ve `r2` `float` sayılar olduğundan çoğu zaman birebir aynı olmazlar. Bazen `l2>r2` bile olabilir ve `t=s-q` olarak hatalı hesaplanacağından küre üzerinde gürültülere sebep olur. Yukarıdaki kodun `else` kısmının koşmasını garanti etmek için `if (l2 > r2 + 0.001)` olarak güncellenir. `if ((int)l2 > (int)r2)` de aynı işlevi görür.

Kırılma (Refraction) ile ilgili belgeye **buradan** erişebilirsiniz.

9. Gölge Testi

Herhangi bir yüzeyin başka bir yüzeyin gölgesinde kalıp kalmadığını belirlemek için Şekil 6'da gösterildiği gibi yüzey üzerindeki kesişim noktasından ışık kaynağına doğru gölge test etme ışını yollanır. Bu ışın ile üçgenler/küreler arasında kesişim testleri yapılarak en yakın üçgen/küre belirlenir. Hesaplanan uzaklık değeri ışık kaynağına olan uzaklıktan küçük ise yüzey bu üçgenin/kürenin gölgesinde kalıyor demektir.



Şekil 6 : Gölge Test Etme Işını

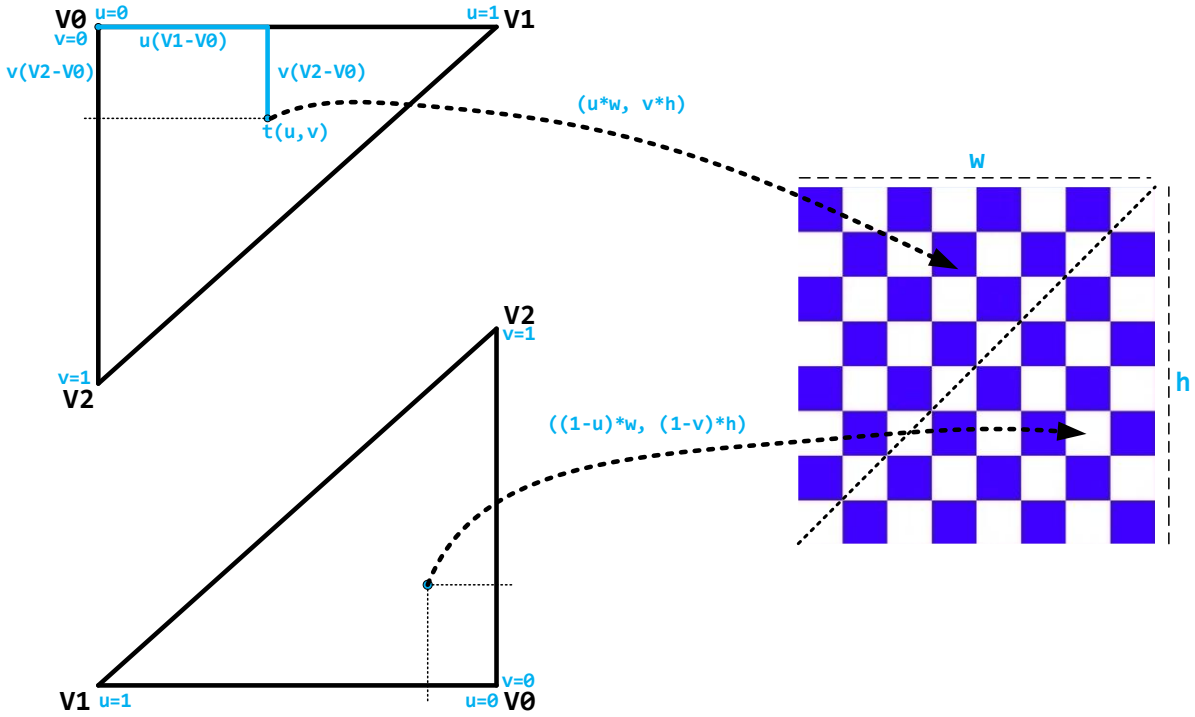
10. Düzlemsel Yüzey Üzerine Doku Kaplama

Işın izleme ile doku kaplama için farklı yöntemler kullanılabilir. Burada Tomas Möler'in ışın-üçgen kesişim testinden faydalanılarak doku kaplama anlatılacaktır. Möller'in yöntemine göre kesişim testi yapıldığında yalnızca kesişim noktasına uzaklık olan t değeri değil aynı zamanda kesişim noktasının barisentrik koordinatları da hesaplanabilir. Barisentrik koordinatlar ve üçgenin köşe noktaları kullanılarak üçgen üzerinde herhangi bir noktaya gidebilmek için aşağıdaki ifade kullanılır:

$$t(u,v) = V_0 + u(V_1 - V_0) + v(V_2 - V_0)$$

Üçgenin köşe noktaları uygun sırada tutulursa barisentrik koordinatlarla doku kaplama için kullanılan resim dosyasındaki doğru piksel koordinatları rahatlıkla bulunabilir. Bunun için bazı varsayımlar yapılmıştır. Birincisi üzerine doku kaplama yapılacak yüzey düzlemseldir. İkincisi yüzey iki dik üçgenden oluşmaktadır. Dik üçgenlerin V_0, V_1 ve V_2 köşe noktalarından dik kenarları birleştiren V_0 olmalıdır. Bu varsayımlar kullanarak doku kaplama şu şekilde yapılır: Doku kaplamada kullanılacak resmin genişliği `w`, yüksekliği `h` olduğunda, üzerine doku kaplama yapılacak olan yüzeye ait dik üçgenlerden birincisi için resim dosyasındaki koordinatları bulmak için `(u*w,v*h)` formülü kullanılır. İkinci dik üçgen için de `((1-u)*w,(1-v)*h)` formülü kullanılır. Şekil 7'de barisentrik koordinatlarla doku kaplamanın nasıl yapıldığı gösterilmektedir.

Diğer doku kaplama yöntemleriyle ilgili belgeye **buradan** erişebilirsiniz.



Şekil 7: Barisentrik koordinatlarla doku kaplama

11. Arkayüz Kaldırma (Backface Culling)

Arkayüz olan yüzey (veya üçgen) bakış noktası ile arasında başka yüzeyler olmasa bile ters durduğu için görünmeyen yüzeydir. Yüzeyin veya üçgenin ters durması ne demektir? Y=0 yüzeyi üzerinde aşağıda köşe noktaları **U0,U1,U2** olarak verilmiş üçgeni düşünelim.

U0(0, 0, 40) U1(40, 0, -40) U2 (-40, 0, -40)

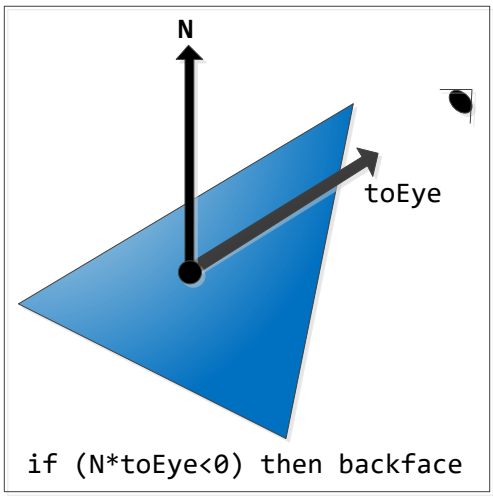
Bu üçgenin +Y ve –Y eksenlerine bakan iki farklı yüzü vardır. Üçgenin yüzey normali hesaplanırsa **(0, 6400, 0)** bulunur. Normalize edilirse **(0,1,0)** olur. Şimdi köşe noktalarının koordinatlarını farklı sırada yazalım:

U0(40, 0, -40) U1(0, 0, 40) U2 (-40, 0, -40)

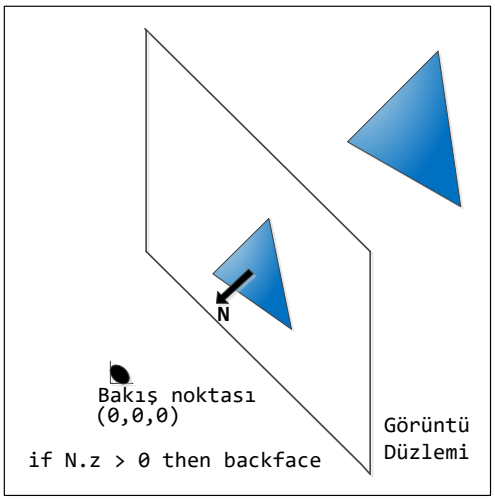
Tekrar yüzey normali hesaplanırsa **(0,-6400,0)** bulunur. Normalize edilirse bu sefer **(0,-1,0)** olur. İlk hesaplanan yüzey normali **(0,1,0)** +Y eksenine doğru şimdiki ise **(0,-1,0)** –Y eksenine doğru çıktı. Dolayısıyla köşe noktalarının sırası değişince yüzey normalinin doğrultusu da değişmektedir. 3D kartezyen koordinat sisteminde Z ekseni (0,0,0) 'dan ileriye doğru pozitif artıyor ise (+Z) köşe noktaları saat yönünde (ClockWise-CW) negatif olarak artıyor ise (-Z) saat yönünün tersi sırada (CounterClockWise-CCW) tanımlanmalıdır. Bu kurallara sırasıyla sol el ve sağ el kuralı denir. Sol el kuralında sol elin 4 uzun parmağı +X eksenini gösterirken +Y eksenini gösterecek şekilde katlandığında baş parmağın doğrultusu +Z eksenini gösterir. Aynı işlem sağ el ile yapıldığında baş parmak yine +Z 'i gösterir. DirectX 3D kartezyen koordinatları sol el; OpenGL de sağ el kuralına göre belirler.

Yüzey normalinin doğrultusunun köşe noktalarının sırasına bağlı olarak değişmesi ile arkayüz olması arasında ne ilişki var? 3D cisimlerin görüntüleri çizilirken onların renkleri doğrudan ekrana basılmaz. Son renk değeri belirlenirken ışık kaynağının konumu da dikkate alınır. O cisim ışık kaynağından az miktarda veya çok miktarda ışık almasına bağlı olarak rengi belli katsayılarla çarpılarak değiştirilip ekrana basılır. Bu katsayılardan biri diffuse diğeri de specular katsayılardır. Her ikisi de belirlenirken yüzey normali kullanılır. Örneğin diffuse katsayı yüzey normali ile bakış noktasına doğru olan vektörlerin skaler çarpımı ile bulunur.

Yukarıda ilk tanımlanan U üçgeni içindeki **(0,0,0)** noktasının diffuse katsayısını hesaplayalım. Bunun için ışık kaynağının koordinatları bilinmelidir. Işık kaynağı **(0,60,80)** noktasında olsun. Işık kaynağına doğru olan vektör **(0,60,80)-(0,0,0)=(0,60,80)** olur. Normalize edilirse **(0,0.6,0.8)** bulunur. Diffuse kaysayıyı bulmak için bu vektör ile yüzey normali skaler çarpılırsa 0.6 bulunur. Aynı işlemler köşe noktalarının sırası değiştirilerek normali **(0,-1,0)** olan üçgen için yapılırsa bu sefer -0.6 bulunur. Renk değerinin negatif olması imkansız olduğundan -0.6 diffuse katsayı olarak kullanılamaz. Başka bir deyişle bu yüzey ışık kaynağı tarafından aydınlatılmıyor demektir. Demek ki yüzey normalinin doğrultusuna bağlı olarak yüzey ışık alır veya almaz. Benzeri şekilde yüzey normali ile bakış noktasına doğru olan vektörler skaler çarpılınca sonuç negatif çıkıyorsa yüzeyin bakış noktasına göre görülmesi imkansız demektir. Yani arkayüzdür. Böylece arkayüz belirlemede kullanılan birinci yöntem açıklanmış oldu. Demek ki herhangi bir yüzeyin arkayüz olup olmadığını belirlemek için o yüzey üzerindeki herhangi bir noktadan (üçgen için köşe noktalarından herhangi biri) bakış noktasını doğru olan vektör ile yüzey normali skaler çarpılır. Sonuç sıfırdan küçükse bu yüzey arkayüzdür. Mesela yukarıda ikinci olarak tanımlanan U üçgeni bakış noktası **(0,60,80)** alındığında arkayüz olmaktadır. Bakış noktasına doğru olan vektör ile yüzey normalini **skaler çarparak arkayüz kaldırma** Şekil 8'de gösterilmiştir.

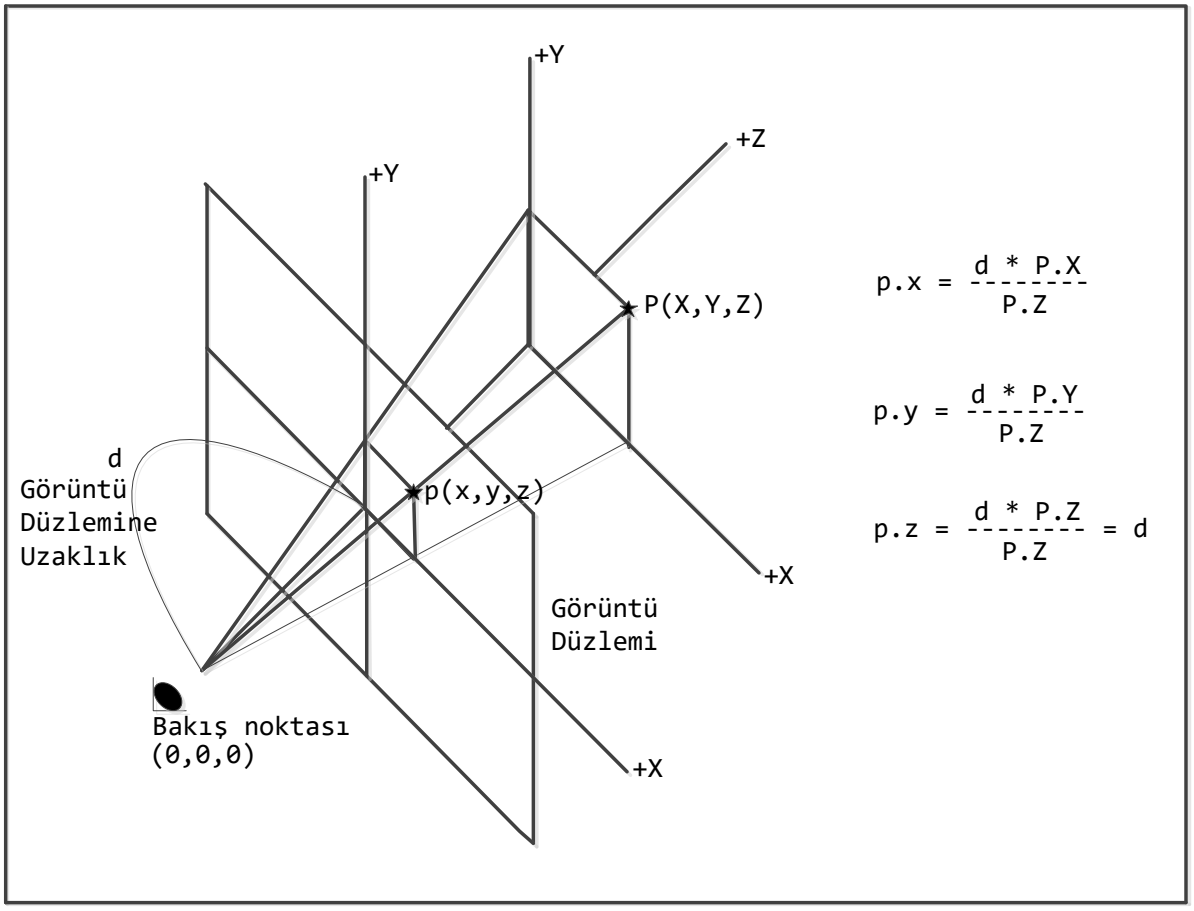


Şekil 8: Skaler çarpımla arkayüz kaldırma



Şekil 9: Vektörel çarpımla arkayüz kaldırma

Arkayüz kaldırmada kullanılan diğer bir yöntemde üçgenin görüntü düzlemine izdüşümü alınır ve normali hesaplanır. Eğer normalin Z bileşeni 0'dan büyük ise arkayüzdür. İzdüşüm sonrası vektörel çarpımla normal hesaplandığından bu yöntem **vektörel çarpımla arkayüz kaldırma** olarak isimlendirilecektir. Vektörel çarpımla arkayüz kaldırma Şekil 9'da gösterilmiştir. Herhangi bir noktanın benzer üçgenler yardımıyla görüntü düzlemine perspektif dönüşüm ile izdüşümünün nasıl yapıldığı Şekil 10'da gösterilmiştir.



Şekil 10: Perspektif Dönüşüm (Projeksiyon) ile İzdüşüm

12. Etkileşimli Işın İzleme

Etkileşimli Işın İzleme W,A,S,D tuşlarıyla 3D ortamda gezinme olarak özetlenebilir. Etkileşimli Işın İzleme ile ilgili belgeye **buradan** erişebilirsiniz.