광등을 선분, 또는 길쭉한 직사각형으로 생각하고 그 위의 각 지점에서 발하는 빛의 효과를 계산한다. 실제로 우리가 사용하는 대부분의 광원은 점 광원이라고 하기보다는 분산 광원이라 할 수 있는데, 이 광원을 사용하면 매우 자연스러운 효과를 낼 수 있다(그림 3.9(e)). 반면에 길이, 면적, 또는 경우에 따라 체적을 가지는 물체의 여러 지점에서 출발하는 빛의 효과를 계산해주어야 하기 때문에 계산량이 상당히 많아 실시간 그래픽스에서는 직접 지원하지 않고 고급 기법을 사용하여 분산 광원 효과를 흉내내게 된다. 여기서 비단 렌더링뿐만 아니라 그래픽스 분야에서 사용되는 모델의 단순화 정도에 따른 장단점의 차이를 알 수 있는데, 점 광원이나 평행광원처럼 극도로 단순화 된 모델에서는 계산량이 적어 빠른 시간 안에 렌더링을 할 수 있으나, 분산 광원에 비해 상대적으로 생경한 이미지를 만들게 된다.

지금까지 설명을 한 내용을 종합해보면 앞에서 언급한 네 가지 문제 중 처음 두 가지에 대하여 답을 할 수가 있다. 렌더링 계산에서 사용되는 광원 모델로 점 광원, 평행 광원, 스폿 광원, 앰비언트 광원, 분산 광원 등이 있는데, 보통 실시간 렌더링 파이프라인에서는 앞의 네 가지 모델을 지원한다([문제 1]). 또한 실시간 파이프라인에서는 기본적으로 지역 조명 모델을 사용하기 때문에 광원에서 직접 들어오는 빛만 고려를 하고(경로), 방향은 광원의 기하 속성과 물체 표면의 위치에 의해 결정이 되며, 마지막으로 밝기나 색깔은 각 광원을 설정하는데 사용하는 색깔을 사용한다([문제 2]). 물론 밝기는 물체와 광원간의 거리를 사용하여 조절하기도 하는데 자세한 내용은 뒤에서 설명하겠다.

제 5 절 퐁의 조명 모델

이제 나머지 두 문제 [문제 3]과 [문제 4]에 관련된 사항에 대하여 알아보자. 앞 절의 내용은 물체 표면 상의 점으로 들어오는 빛을 어떻게 결정할 것인가에 관한 것

이었다. 이제 빛이 물체를 향해 들어 왔을 때, 우리가 바라보고 있는 시선의 방향으로 반사되는 빛의 색깔을 어떠한 방식으로 계산해야 할 지를 결정하여야 한다. 우리가 물체를 특정 색깔로 느낀다는 것은 광원으로부터의 입사 광선 중 일부는 물체에 흡수가 되고, 나머지 빛이 눈을 향하여 반사가 될 때 그러한 빛의 색깔로 느끼는 것인데, 여기에는 주로 광원의 밝기와 색깔, 빛이 들어오는 방향, 물체를 바라보는 방향, 그리고 물체의 기하 및 반사 성질 등이 영향을 미친다. 그래픽스 렌더링에서 사용되는 대부분의 라이팅 모델의 기반이 되는 퐁의 조명 모델(Phong's illumination model)은 바로 물체 표면에서 빛이 어떻게 반사가 되는가를 계산하는데 사용이 된다. 특히 이 모델은 주어진 물체의 빛의 반사 형태를 결정하는데 사용되므로 퐁의 반사 모델(Phong's reflection model)이라고 하기도 한다. 이 모델은 물리학적으로 정확한 모델은 아니나 비교적 계산량이 적고, 실험적으로 우수한 성능을 나타내기 때문에, 컴퓨터 그래픽스 렌더링 분야에서 거의 표준처럼 사용이 되고 있다. 실제로 퐁의 모델을 상황에 맞게 조금씩 변형을 하여 사용하나, 그 기본은 동일하다고 할 수 있다. 이 절에서는 퐁의 조명 모델 전반에 대하여 살펴보고, 다음절에서 OpenGL 시스템에서 실제로 사용되는 모델에 대하여 구체적으로 알아보도록 하겠다.

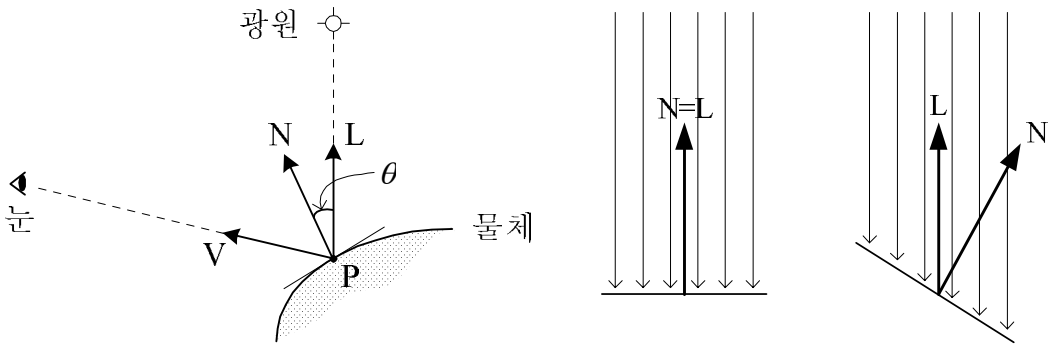
5.1 세 가지 종류의 반사

퐁의 조명 모델에서 고려하는 빛의 반사는 기본적으로 앰비언트 반사(ambient reflection), 난반사(diffuse reflection), 그리고 정반사(specular reflection) 등 세 가지 형태의 반사로 나누어진다.

5.1.1 앰비언트 반사

앰비언트 반사는 물체가 앰비언트 광원에 대하여 어떻게 반응을 할 것인가에 관한 것이다. 앰비언트 빛이란 광원에서 직접 들어오는 빛이 아니라 간접적으로 들어오는 빛을 모델링 하기 위하여 사방에 고르게 퍼져 있다고 가정하는 빛을 의미하는데, 광원의 색깔은 파장 λ의 가시 광선 영역에 대하여 $I_{a\lambda}$와 같이 설정할 수 있다. 광원의 밝기 또는 색깔을 정확하게 설정하려면 모든 파장 λ에 대하여 정의가 되어야 하나, 그래픽스 렌더링에서 사용되는 조명 모델에서는 RGBA 모델을 사용하여 $\lambda = R, G, B, A$에서 정의가 된다. 일단 이 절에서는 앞에서 언급한 바와 같이 편의상 RGB 색깔 모델을 사용한다고 가정하자. 이 경우 빛의 색깔은 $I_{a\lambda} = (I_{aR}\ I_{aG}\ I_{aB})^t$와 같이 세 개의 원소를 가지는 벡터로 표현된다.

앰비언트 빛이 물체 전체에 균일하게 들어올 때 물체를 이루는 물질의 성질에 따라 이를 반사하는 방식이 다르게 된다. 빛의 색깔의 각 채널에 대하여 물체가 각각 k_{aR}, k_{aG}, k_{aB} $(0 \leq k_{aR}, k_{aG}, k_{aB} \leq 1)$의 비율만큼만 반사를 한다면 결과적으로 반사되어 보이는 빛은 $k_{a\lambda} \equiv (k_{aR}\ k_{aG}\ k_{aB})^t$라 할 때, $I_\lambda = I_{a\lambda} \cdot k_{a\lambda}$와 같이 표현할 수 있는데, 여기서의 곱셈은 각 채널 값끼리의 곱셈을 의미한다. 이 때 세 개의 원소를 가지는 벡터 $k_{a\lambda}$를 앰비언트 반사 계수(ambient reflection coefficient)라고 하는데, 물체가 들어오는 앰비언트 빛을 어떻게 반사를 시킬 것인가를 결정하는 물질의 고유 성질이다. 앰비언트 반사의 경우 빛이 사방에서 고르게 들어온다고 가정을 하며, 반사 또한 사방으로 균일하게 반사된다고 가정을 하기 때문에 관찰자의 위치에 상관없이 같은 색깔로 반사되어 보이게 된다. 이렇게 하여 엉성하기는 하나 지역 조명 모델에서 전역 조명 모델의 흉내를 내게 되는 것이다.



(a) 세 가지 중요한 방향 (b) 코사인 법칙의 원리

그림 3.10: 난반사의 원리

5.1.2 난반사

앰비언트 반사와는 달리 광원에서 직접 들어오는 빛은 크게 난반사와 정반사 등의 두 가지 종류의 반사를 한다. 우선 난반사는 입사 광선을 사방으로 고르게 동일한 밝기로 반사하는 형태의 반사를 시뮬레이션하는데 사용이 된다. 따라서 앰비언트 반사의 경우와 같이 만약 물체가 순수하게 난반사만 한다면, 관찰자의 시점이 어디에 있건 같은 밝기로 반사되어 보인다. 난반사는 물체 표면이 종이, 분필, 석고상 등과 같이 반짝거리지 않고 좀 둔탁해 보이는 물체를 렌더링 하는데 사용된다. 이러한 물체들은 시점을 이리 저리 옮겨가면서 보아도 비교적 보이는 색깔이 변하지 않는 성질을 가지고 있다. 그림 3.10(a)를 보자. 지금 물체 표면 상의 한 점 P에서 반사되는 빛의 색깔을 계산하려 한다. 광원에서 출발한 빛이 직접 물체로 들어오는 방향의 반대 방향을 L이라고 하고, 관찰자를 향해 반사되는 방향과 P에서 물체의 표면에 수직인 법선 벡터를 각각 V와 N이라는 벡터로 나타내자[9].

어떤 물체를 이루는 물질이 순수하게 난반사를 한다면 그 물체를 이상적인 난반사체(idea diffuse reflector)라고 하는데, 다른 말로 램버트 반사체(Lambertian re-

[9]편의상 이 벡터들은 길이가 1인 단위 벡터로 가정하겠다.

flector)라고 부르기도 한다. 이러한 물체들은 표면에서 빛을 반사시킬 때 램버트의 코사인 법칙(Lambert's cosine law)을 따른다. 이 법칙에 의하면 두 벡터 L과 N사이의 각도를 θ라 할 때 반사되는 빛 에너지의 양은 $\cos\theta$에 비례한다. 따라서 반사되는 빛의 색깔은 광원으로부터의 입사 각도와 바라보는 점에서의 법선 벡터에 영향을 받지만, 반면에 어디에서 바라보건(즉 V 벡터 방향이 어떻게 변하건) 동일하게 보인다. θ값이 커질수록 $\cos\theta$의 값이 작아지므로, 이 법칙에 따르면 빛이 수직으로 내려 쪼일 때 가장 세게 반사가 되고, 비스듬하게 빛을 비출 수록 반사되는 빛의 밝기가 약해지는 성질을 가진다. 램버트의 법칙은 그림 3.10(b)를 보면 직관적으로 이해할 수 있다. 즉 왼쪽에서처럼 수직으로 빛이 내려 쪼일 때에 비해 오른쪽에서와 같이 θ만큼의 입사 각도를 가지고 동일한 크기의 빛 에너지를 내려 쪼이면, 물체 표면의 단위 면적 당 빛의 입사량은 $\cos\theta$에 비례하여 줄어 들게 되고, 따라서 반사되는 빛의 양도 그에 따라 줄어 든다고 생각하면 된다. 순수한 난반사를 통하여 반사되는 빛의 색깔은 다음과 같은 공식에 의하여 계산된다.

$$I_\lambda = I_{l\lambda} \cdot k_{d\lambda} \cdot \cos\theta = I_{l\lambda} \cdot k_{d\lambda} \cdot (N \cdot L)$$

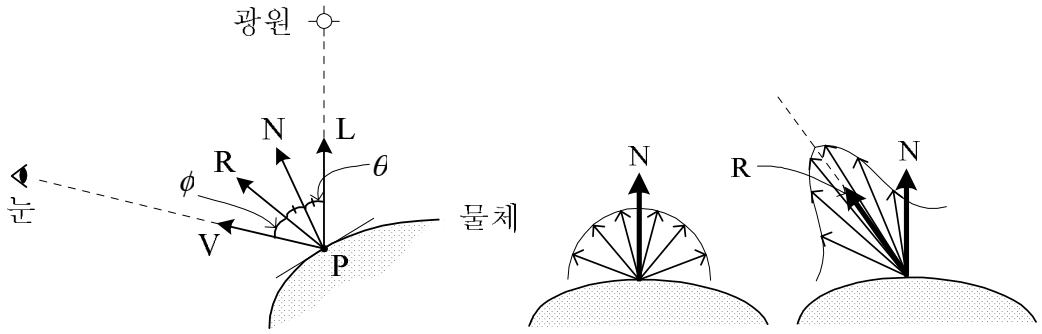
여기서 $I_{l\lambda}$는 광원의 색깔이고, $k_{d\lambda}$는 물체가 난반사를 할 때 각 채널의 빛을 어떠한 비율로 반사를 시킬 것인지를 결정하는 난반사 계수(diffuse reflection coefficient)로서, 앰비언트 반사 계수와 같이 세 개의 0과 1 사이의 값으로 구성된 벡터이다. 또한 여기서 두 방향 벡터 N과 L의 곱은 벡터의 내적을 의미하는데, 이 둘의 길이가 1이므로 두 방향의 각도에 대한 코사인 값은 내적 값과 같게 된다. 이 때 $I_{l\lambda}$와 $k_{d\lambda}$는 각 채널끼리 곱하고 상수 $\cos\theta$는 각 채널에 곱하게 된다. 난반사는 보통 θ가 0도와

90도 사이의 값을 가질 경우에만 고려한다. 만약 이 값이 90도보다 크다면 이는 물체 표면의 뒤쪽에서 빛이 들어오는 것을 의미하므로, 대개의 경우 이 코사인 값을 0으로 처리하는데, 퐁의 모델을 어떻게 변형하는가에 따라 각도 $180-\theta$에 대한 코사인 값을 적용하기도 한다.

난반사에 대하여 요약을 하면 물체가 순수하게 난반사를 할 경우 관찰자의 시점에는 상관없이 빛의 입사 각도에 대한 코사인 값, 광원의 밝기, 그리고 물체의 난반사 계수에 비례하여 빛을 반사한다. 한 가지 중요한 사실은 우리가 어떤 물체에 대한 반사 성질을 지정할 때 물체의 전반적인 색깔을 난반사 계수 $k_{d\lambda} = (I_{dR}\ I_{dG}\ I_{dB})^t$를 통하여 지정을 한다는 점이다. 즉 어떤 물체를 푸른색으로 하고 싶다면 바로 그 색깔을 난반사 계수로 사용을 하면 되는데, 따라서 이 계수 값이 다른 조명 인자들과 비교하여 물체의 전반적인 색깔에 가장 영향을 많이 미치는 인자라 할 수가 있다.

5.1.3 정반사

입사 광선을 사방으로 고르게 반사시키는 난반사와는 달리 정반사는 물체 표면으로 들어오는 빛을 특정 방향을 중심으로 집중적으로 반사를 하는 성질을 가진다. 정반사는 주로 자동차 표면이나 보석과 같은 금속, 또는 광택을 낸 사과와 같이 반짝거리는 물체를 표현하는데 사용이 된다. 이러한 물체들의 반사에 대한 특징중의 하나는 표면에 상대적으로 특별히 반짝거리는 부분이 생기게 되는데, 이러한 부분을 하이라이트(hightlight)라 한다. 거울은 정반사를 하는 극단적인 예라고 할 수 있는데 입사 광선을 입사각과 대칭이 되는 반사각 방향으로만 빛을 반사를 한다. 일반적으로 정반사를 많이 하는 반짝거리는 물체들은 거울과 같이 한 방향으로만 빛을 반사시키지는 않고, 특정 방향 둘레로 어느 정도의 범위를 가지고 집중적으로 빛을 반사시킨다.



(a) 네 가지 중요한 방향 (b) 난반사 및 정반사의 반사 형태

그림 3.11: 정반사 및 빛의 반사 형태

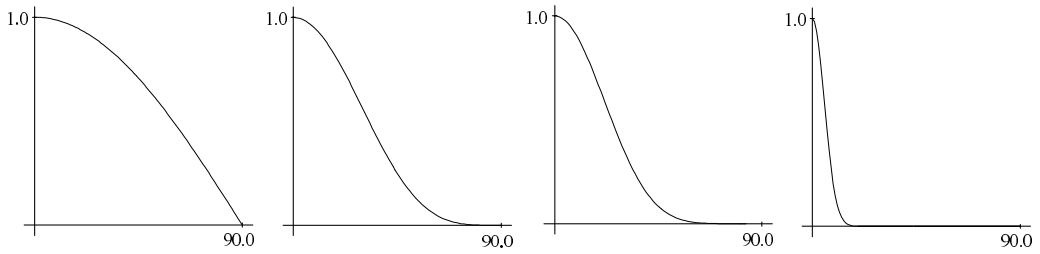
그림 3.11(a)를 보면 정반사에 영향을 미치는 네 개의 방향이 도시되어 있는데, 단위 벡터 L, V, 그리고 N의 의미는 앞에서와 같고, R 벡터는 L을 N 벡터를 중심으로 하여 반대 방향으로 반사를 시킨 방향을 나타낸다. 이 방향을 정반사 방향(specular reflection direction)이라고 하는데, 정반사의 경우 한 지점으로 들어오는 빛을 R 방향으로 가장 강하게 반사를 시키고, 방향이 R에서 벗어날 수록 반사되는 빛의 세기가 급격히 약해지게 된다. 그림 3.11(b)를 보면 난반사와 정반사의 반사 형태의 차이를 쉽게 이해할 수 있다. 물체와 광원을 고정시키고 관찰자 시점을 움직이면서 한 지점을 바라볼 때, 순수하게 난반사를 하는 물체의 경우에는 항상 같은 밝기의 색깔로 보이나, 정반사를 하는 물체의 경우에는 보는 시점에 따라 밝아졌다 어두워졌다 하게 된다. 특히 R 벡터 방향에서 바라볼 때 가장 밝게 보이게 되는데, 이를 거꾸로 생각을 해보면 우리가 정반사를 하는 물체를 바라볼 때, 표면의 각 지점에서의 R 벡터 방향과 관찰자를 향한 V 벡터 방향이 서로 비슷한 지역에 하이라이트가 생김을 알 수 있다.

이러한 정반사의 형태를 퐁이라는 사람이 코사인 함수를 사용하여 다음과 같은 실험적인 모델을 설정하였다.

$$\begin{aligned} I_\lambda &= I_{l\lambda} \cdot w(\theta, \lambda) \cdot \cos^n \phi \\ &\approx I_{l\lambda} \cdot k_{s\lambda} \cdot \cos^n \phi \\ &= I_{l\lambda} \cdot k_{s\lambda} \cdot (R \cdot V)^n \end{aligned}$$

여기서 $I_{l\lambda}$는 앞에서와 같이 광원의 색깔이고 $w(\theta, \lambda)$는 정반사 계수(specular reflection coefficient)로서 빛의 입사각 θ와 파장 λ의 함수로 표현이 된다. 정반사 계수는 물질의 고유한 성질로서 들어오는 빛을 어떤 비율로 반사를 시킬 것인가를 결정하는데, 입사각이 90도일 경우 들어오는 빛을 전부 반사시키는 반면 90도보다 작을 경우에는 물질의 성질에 따라 반사되는 빛의 비율이 달라진다. 일반적으로 대부분의 불투명한 물체들에 대하여 이 값이 각도에 따라 크게 변하지 않기 때문에 상수 벡터인 $k_{s\lambda} = (I_{sR}\ I_{sG}\ I_{sB})^t$ 값으로 근사화한 모델을 사용한다. ϕ는 정반사 방향 R과 시선의 방향 V와의 각도인데, R과 V를 단위 벡터라 할 경우 $\cos\phi$는 R과 V의 내적과 같은 값을 가지게 되므로 마지막에 있는 공식을 얻게 된다. 난반사의 경우와 다른 점은 코사인 값을 그냥 사용하는 것이 아니라 주어진 값 n에 대하여 n 제곱을 한다는 사실이다. 이러한 값 n을 정반사 지수(specular reflection exponent)라 하는데, 이는 물체의 고유 성질로서 보통 1부터 수백 사이의 값을 가진다. 과연 정반사 모델에서 정반사 지수의 역할은 무엇일까?

앞에서도 강조한 바와 같이 물체가 정반사를 할 때 정반사 방향으로 빛을 가장 강하게 반사를 하고, 그 방향에서 벗어날수록 반사되는 빛의 세기가 약해진다고 하였다. 이러한 성질은 정반사를 위한 공식에서 $\cos\phi$에 의하여 표현이 되고 있는데,



(a) $n = 1$ (b) $n = 4$ (c) $n = 8$ (d) $n = 128$

그림 3.12: 함수 $\cos^n \phi$의 그래프

이 값에다 n 제곱을 함으로써 약해지는 속도를 조절할 수 있게 된다. 그림 3.12를 보면 $\cos^n \phi$ 값이 n 값에 따라 어떻게 변하는가를 알 수 있다. 잘 알다시피 코사인 값은 ϕ 값이 0도에서 90로 증가함에 따라 그 값이 1에서 0으로 점점 작아지므로, n 값이 커질수록 $\cos^n \phi$ 값은 더 빠른 속도로 작아진다. 따라서 정반사 지수 n을 크게 하면 바라보는 방향이 정반사 방향에서 조금만 벗어나도 반사되는 빛의 밝기가 급격히 줄어들고, 그 결과 시점을 고정시키고 물체를 바라볼 때 밝게 보이는 하이라이트 지역의 크기가 작아진다. 다시 말해서 하이라이트의 크기를 크게 하려면 n값을 작은 값으로 설정하고, 반대로 작게 하려면 이 값을 크게 하면 되는데, 물론 여기에는 물리학적으로 정확한 어떤 수치가 있는 것이 아니라 프로그래머의 판단에 따라 정반사 지수 n 값을 실험적으로 바꾸어 가며 가장 적절한 값을 선택하여야 한다.

정반사에 대하여 한 가지 더 고려를 해야 할 것은 많은 경우 정반사에 의해 반사되는 빛의 색깔은 물체의 고유 색깔보다는 주로 광원의 색깔에 좌우된다는 점이다. 예를 들어 검은색의 자동차 표면에 흰색 광원의 빛이 반사되는 장면을 상상해보면, 하이라이트가 하얗게 생김을 쉽게 알 수 있다. 따라서 정반사의 공식을 사용할 때 이러한 점을 고려하여 각 인자 값을 설정하여야 한다. 광원의 색깔 $I_{l\lambda}$ 값이 이미 이 공식에 있으므로 물체의 고유 색상으로 설정하는 난반사 계수 $k_{d\lambda}$와는 달리, 일반

적으로 $k_{s\lambda}$의 경우에는 0과 1사이의 적절한 값을 사용하여 광원에서 들어온 빛을 어떤 비율로 반사시킬 것인가를 조절하게 된다.

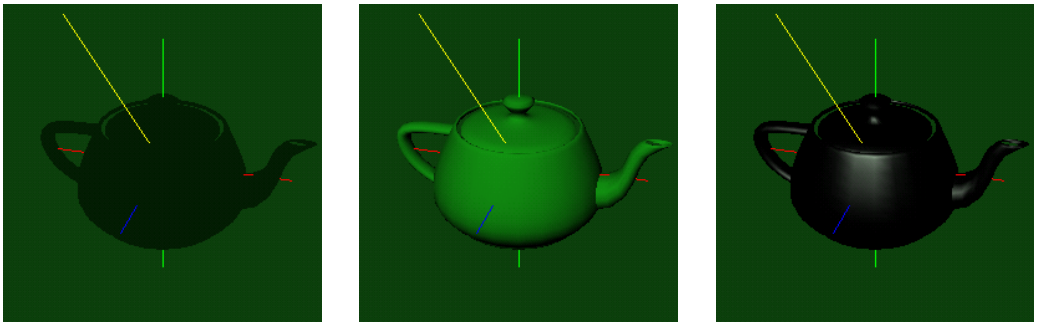
5.2 퐁의 조명 모델에 대한 변형

5.2.1 기본적인 퐁의 조명 모델

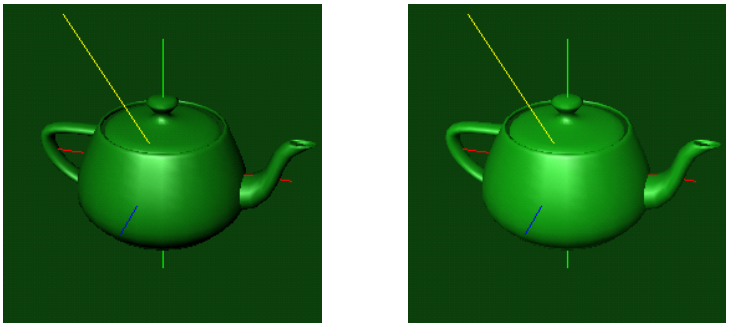
앞에서 순수하게 난반사를 하거나 또는 정반사를 하는 물체가 그 표면에서 빛을 어떻게 반사시키는지에 대하여 알아보았는데, 일반적으로 보통 물체들은 순수하게 한 가지 종류의 반사가 아니라 두 가지 모두를 적절히 혼합하여 빛의 반사 형태를 시뮬레이션하게 된다. 간접 조명의 효과를 위한 앰비언트 반사까지 포함하여 다음과 같은 기본적인 퐁의 조명 모델을 얻을 수 있다.

$$\begin{aligned} I_\lambda &= I_{a\lambda} \cdot k_{a\lambda} + I_{l\lambda} \cdot k_{d\lambda} \cdot (N \cdot L) + I_{l\lambda} \cdot k_{s\lambda} \cdot (R \cdot V)^n \\ &= I_{a\lambda} \cdot k_{a\lambda} + I_{l\lambda} \cdot \{k_{d\lambda} \cdot (N \cdot L) + k_{s\lambda} \cdot (R \cdot V)^n\} \end{aligned}$$

그림 3.13은 평행 광원을 사용하여 에머랄드로 만든 것처럼 보이도록 물주전자를 렌더링한 장면을 보여주고 있다. 여기서 비스듬한 직선은 빛이 들어오는 방향을 나타내는데, 앞에서도 설명한 바와 같이 앰비언트 반사는 빛의 방향에 상관없이 물체 전체에 대하여 아주 약하게, 그리고 고르게 반사함을 알 수 있다(그림 3.13(a)). 난반사의 경우에는 빛이 들어오는 방향과 물체의 법선 벡터 방향간의 각도가 작은 지점일수록 밝게 반사됨을 알 수 있으며, 특정 지역에 하이라이트가 생기는 것이 아니라 물체 표면의 색깔이 전체적으로 부드럽게 반사되어 보인다(그림 3.13(b)). 특히 난반사를 통하여 물체의 전반적인 색깔을 표현하고 있다. 한편 정반사의 경우 물주전자의 표면에 하이라이트를 만들어 마치 금속성의 물질로 만든 것과 같으 효과를



(a) 앰비언트 반사 (b) 난반사 (c) 정반사



(d) 난반사+정반사 (e) 세 가지 반사의 합

그림 3.13: 세 가지 종류의 반사

내려 하고 있다(그림 3.13(c)). 여기서 하이라이트는 물체의 표면을 바라보는 방향과 정반사 방향이 일치하는 지점을 주위로 생김을 알 수 있다. 그림 3.13(d)는 난반사와 정반사를 합쳐 렌더링을 한 것인데, 빛이 내려 쪼이는 부분과 그렇지 않은 부분의 명암 차이가 커서 약간 거친 느낌을 준다. 여기서 앰비언트 반사 효과까지 합하면 훨씬 자연스러운 모습을 보이게 된다(그림 3.13(e)).

일반적으로 이렇게 세 가지 종류의 반사를 더하여 물체의 쉐이딩이 된 색깔을 표현하게 되는데, 각 반사에 대하여 어떻게 가중치를 주는가에 따라 보이는 모습이 다르게 될 것이다. 예를 들어 물체 표면에서 난반사보다 정반사를 더 많이 하게 하여 반짝거리는 효과를 내고 싶으면 $k_{s\lambda}$의 값을 $k_{d\lambda}$보다 상대적으로 크게 하면 된다.

5.2.2 해프웨이 벡터

퐁의 조명 모델 공식을 계산하기 위해서는 정반사에 관련된 부분에 대하여 정반사 방향 R을 구해야 한다. 이 벡터는 $R = 2(N \cdot L)N - L$과 같이 됨을 쉽게 유도할 수 있는데, 이를 위해서는 5번의 덧셈/뺄셈 연산과 7번의 곱셈 연산을 수행하여야 한다. 이는 별로 안 되는 계산량으로 생각할 수 있으나, 복잡한 다면체 모델을 렌더링할 때에는 부담이 될 수도 있다. 앞에서도 언급한 바와 같이 OpenGL 렌더링 파이프라인에서는 눈 좌표계 공간에서 각 꼭지점에 대하여 퐁의 조명 모델을 적용하여 쉐이딩이 된 색깔을 계산한다고 하였다. 약간만 복잡한 장면을 렌더링하더라도 처리해야 할 꼭지점의 개수가 크게 증가하곤 한다. 만약 10만개의 꼭지점에 대하여 R 벡터를 계산한다면 120만 번의 부동 소수점 연산을 수행해야 하는데, 이는 상황에 따라 실시간 계산을 하는데 있어 부담이 될 수 있는 계산량이다.

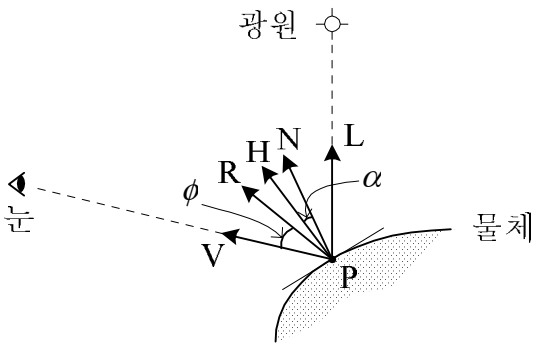


그림 3.14: 해프웨이 벡터

퐁의 조명 모델을 구현할 때 계산량을 줄이기 위한 시도로서 $H = \frac{L+V}{|L+V|}$와 같이 정의되는 해프웨이 벡터(halfway vector)라고 하는 벡터를 사용하여 정반사 공식의 $(R \cdot V)^n$을 $(N \cdot H)^n$으로 대치하여 계산을 한다. 그림 3.14에 도시된 바와 같이 해프웨이 벡터는 광원의 방향 L과 시선의 방향 V의 중간 방향으로서, 물체 표면의 법선 벡터 방향이 이 방향과 일치할 때 가장 강하게 정반사를 하는 방향이 된다.

과연 해프웨이 벡터를 사용하면 어떠한 이점이 있을까? 광원과 관찰자 시점이 렌더링을 하려는 물체로부터 충분히 멀리 떨어져 있다면, 평행 광원과 평행 투영을 해도 무방하고, 이러한 경우 L과 V는 상수 벡터가 된다. 이 때 H 벡터도 상수 벡터

가 되어 L과 V가 고정이 되었을 때, 한 번만 계산을 해주면 된다. R 벡터의 경우 일반적으로 꼭지점 마다 법선 벡터의 방향이 바뀌므로, 각 꼭지점 마다 다시 계산을 해주어야 하기 때문에, H가 상수 벡터인 상황에서는 상당한 양의 계산을 줄일 수가 있다.

한편 퐁의 조명 모델 공식을 계산할 때, $(R \cdot V)^n$을 $(N \cdot H)^n$으로 대치할 경우 원래 계산하려하던 값과 다른 값을 얻게 된다. 두 벡터 N과 H사이의 각도를 α라 하면 L, V, R, N이 모두 한 평면에 있을 때 $\alpha = \frac{\phi}{2}$임을 보일 수가 있고, 만약 같은 평면상에 있지 않을 경우에는 $\alpha > \frac{\phi}{2}$와 같은 관계를 가진다. 해프웨이 벡터를 사용하면 원래 사용하던 $\cos\phi$ 대신에 $\cos\alpha$ 값에 대하여 n 제곱을 하기 때문에 그 값이 변하게 된다. 하지만 이것은 심각한 문제는 아니다. 어짜피 이 공식은 실험적으로 구한 것으로서 만약 사용하는 각도가 커져 원래의 코사인 값보다 큰 값을 얻게 된다면, 정반사 지수를 약간 크게 하여 이를 실험적으로 조절하면 된다.

다시 한번 강조하지만 OpenGL은 실시간 렌더링을 위한 그래픽스 시스템이기 때문에 제한된 시간 안에 필요한 계산을 마치기 위해서 원래의 정확한 렌더링 모델을 상당히 단순화시켜 사용을 한다. 퐁의 조명 모델도 그러한 목적으로 단순화된 실험적인 모델중의 하나이고, 해프웨이 벡터도 그러한 관점에서의 시도라 할 수 있다. 해프웨이 벡터는 일반적으로 널리 쓰이는데 이 때의 조명 모델은 다음과 같다.

$$I_\lambda = I_{a\lambda} \cdot k_{a\lambda} + I_{l\lambda} \cdot \{k_{d\lambda} \cdot (N \cdot L) + k_{s\lambda} \cdot (N \cdot H)^n\}$$

5.2.3 광원의 거리에 따른 빛의 감쇠 효과

어두운 방에서 촛불을 들고 이리저리 움직이면서 물체 표면에 빛이 비추는 광경을 관찰해보면 촛불이 표면에 가까워질수록 밝게 보이고 멀어질수록 점점 어두워짐을

알 수 있다. 점 광원에서 사방으로 고루게 빛이, 즉 복사 에너지가 퍼져 나가갈 때 물체 표면에서 단위 면적 당 받는 에너지의 양은 광원과 물체 표면간의 거리의 제곱에 반비례하게 된다. 이러한 거리에 빛의 감쇠 효과(light attenuation effect)를 퐁의 조명 모델에 집어넣을 수 있으면, 좀 더 사실적인 조명 효과를 낼 수 있을 것이다. 광원에서 현재 쉐이딩을 하려하는 지점까지의 거리를 d라 할 때, 점 광원의 밝기 $I_{l\lambda}$에 $f_{att}(d) = \frac{1}{d^2}$을 곱하면 이러한 효과를 낼 수 있다. 어떻게 보면 자연스러운 공식이라 할 수 있으나, 이 함수의 그래프를 생각해보면 d값이 작을 때, 즉 광원이 물체에 가까이 있을 때는 광원의 움직임에 따라 밝기가 너무나 급격히 변하고, d값이 클 때, 즉 광원이 물체에서 멀리 떨어져 있을 때는 광원의 움직임의 효과가 거의 나타나지 않음을 쉽게 알 수 있다. 실제로 이 함수를 사용하여 렌더링을 할 때 d 값을 적절히 정규화하지 않으면 종종 너무 어둡거나, 너무 밝거나 또는 감쇠 효과가 전혀 나타나지 않는 현상이 발생한다.

이러한 문제를 해결하기 위해 많은 그래픽스 시스템에서는 $f_{att}(d) = \frac{1}{k_0+k_1 \cdot d+k_2 \cdot d^2}$과 같은 형태의 함수를 사용을 한다. 여기서 세 개의 상수 k_0, k_1, k_2는 함수 $f_{att}(d)$가 적절한 범위의 값을 갖도록 실험적으로 선택을 하여 사용을 하면 된다. 이 때에도 이 값이 너무 작아져 전체 이미지가 어두워지는 것을 막기 위하여 $f_{att}(d) = min(\frac{1}{k_0+k_1 d+k_2 d^2}, 1)$과 같이 함수 값에 제한을 두기도 한다. 여하한 경우 광원의 거리에 따른 빛의 감쇠 효과를 고려하는 퐁의 조명 모델은 다음과 같다.

$$I_\lambda = I_{a\lambda} \cdot k_{a\lambda} + f_{att}(d) \cdot I_{l\lambda} \cdot \{k_{d\lambda} \cdot (N \cdot L) + k_{s\lambda} \cdot (N \cdot H)^n\}$$

5.2.4 다중 광원

광원이 한 개가 아니라 여러 개가 있을 때, 즉 다중 광원(multiple light sources)을 사용할 때에는 그러한 상황을 처리하기 위하여 퐁의 조명 모델에서 각 광원의 직접 조명 효과는 모두 더하고 전체 광원에 기인하는 간접 조명 효과는 앰비언트 광원 $I_{a\lambda}$를 통하여 조절을 한다. 광원이 0번 광원에서 $m-1$ 광원까지 m개가 있을 때의 퐁의 조명 모델은 다음과 같다.

$$I_\lambda = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \cdot L_i) + k_{s\lambda} \cdot (N \cdot H_i)^n\} \quad (3.1)$$

여기서 첨자 i가 붙은 값들은 모두 광원의 색깔이나 위치, 방향에 영향을 받는 값들이다. 이 정도가 스폿 광원과 양면 조명 효과를 제외한 일반적으로 많이 쓰이는 퐁의 조명 모델의 한 형태인데, 스폿 광원과 양면 조명에 관련된 사항은 다음절에서 자세히 다루도록 하겠다.