유니티(Unity)를 활용한

그래픽스 프로그래밍

11 Color, Lighting

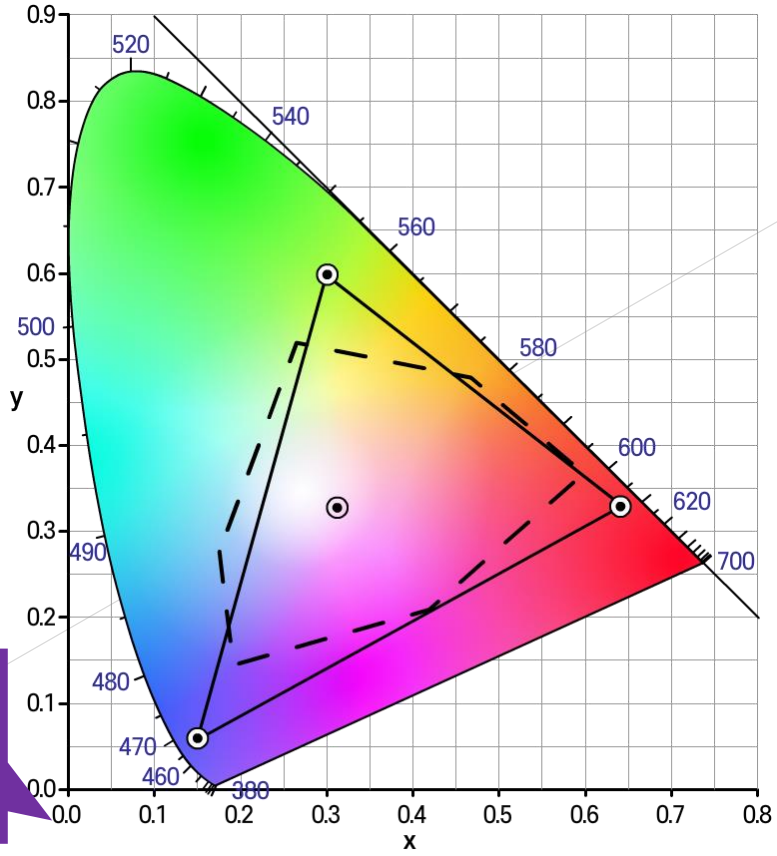
Overview

- Representing Color
  - RGBA
- Flat Shading, Gouraud Shading, Phong Shading
  - Gouraud Shading : Per Vertex Light Rendering
  - Phong Shading : Per Pixel Light Rendering
- Lighting
  - Light-Material Interaction
  - Ambient/Diffuse/Specular
  - Point/Directional/Spot Light Source

Overview

- Shadow
  - Hard shadow
  - Soft shadow
- Halo
- Lens Flare

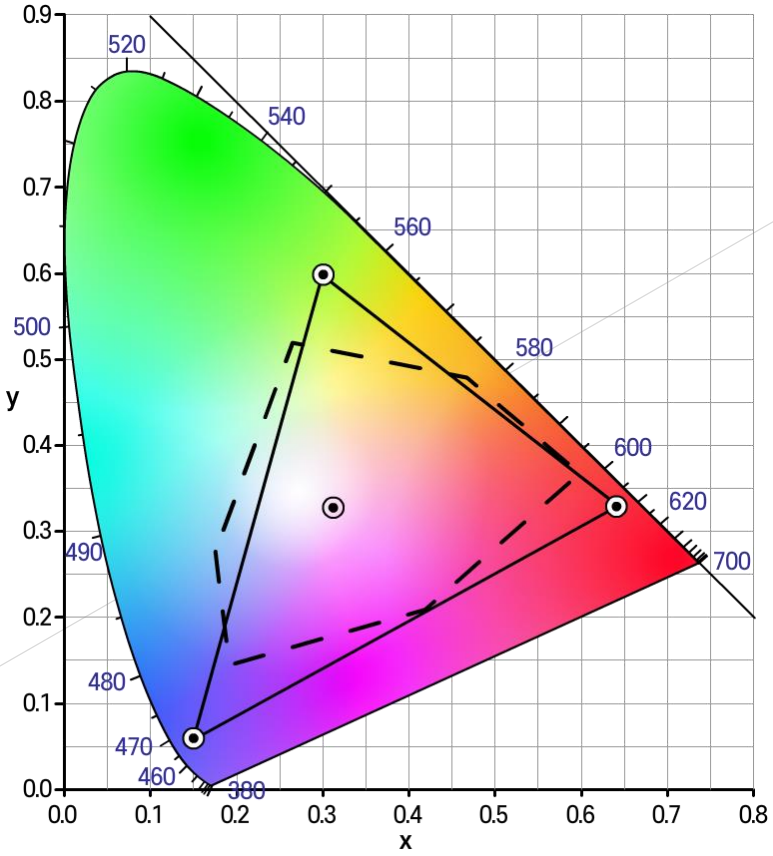
Chromatic Color



이미지 출처 :-

- Hue
  - Color visual experience
- Saturation
  - Saturation is the degree to which the purity of the color decreases due to white/grey
  - Dark colors (red, blue) are those with high saturation. Far from gray.
  - Pastel tones (pink, sky blue) are those with low saturation. Getting closer to gray.

Chromatic Color

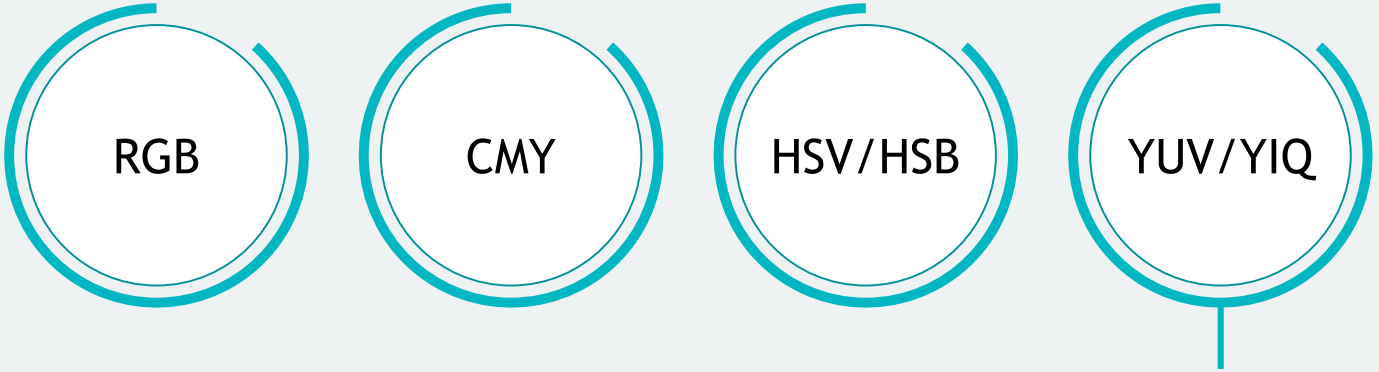


이미지 출처 :-

- Brightness
  - The physical intensity of the light stimulus determines the sensed brightness.
  - The dimension of brightness varies from black to white where there is no color and the brightness is at its maximum.

Color Model

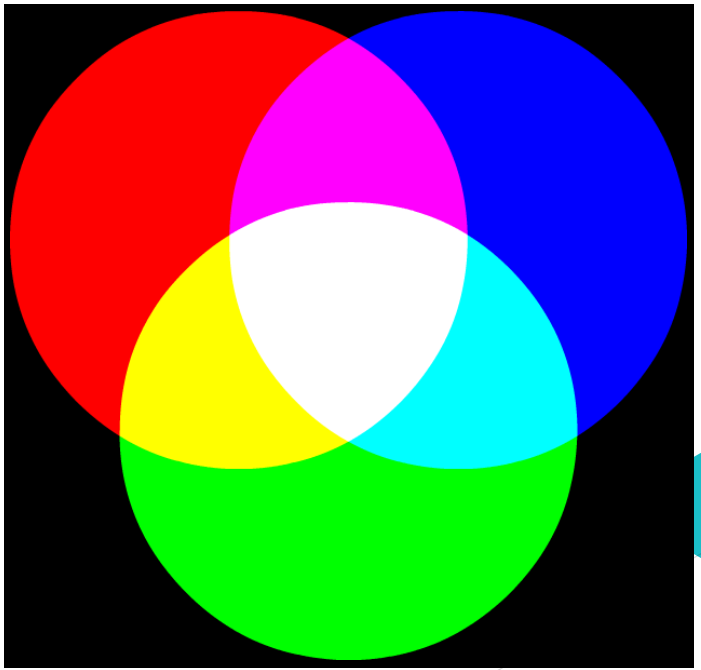
- Which model will you use to express colors?



Color Model

RGB

- Three primary colors of light, suitable for color display
- In the RGB color model, each pixel value is expressed by adding (RED, GREEN, LUE) values
  - n_R: # of bits for R channel
  - n_G: # of bits for G channel
  - n_B: # of bits for B channel
  - $n = n_R + n_G + n_B$
  - $2^n = 2^{(n_R + n_G + n_B)}$ colors can be expressed



RGB

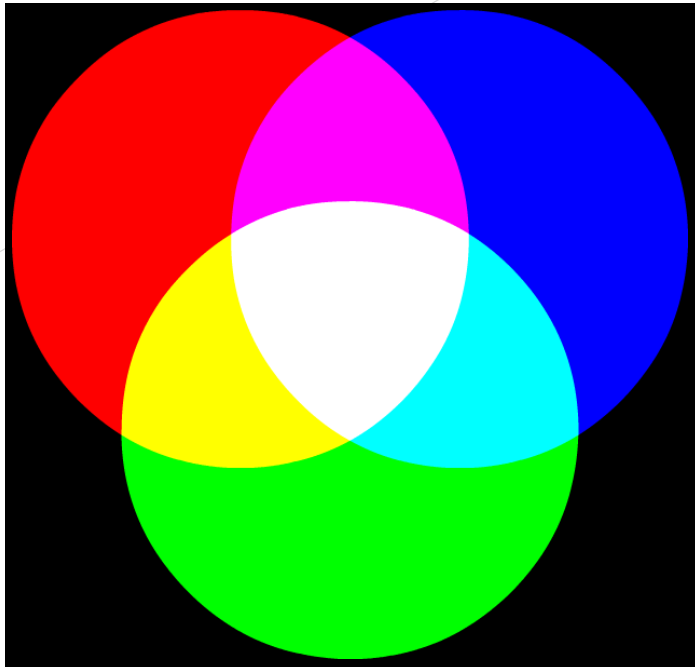
RGB

Black (0, 0, 0)

White (1, 1, 1)

Red (1, 0, 0)

Cyan(0, 1, 1)



Green (0, 1, 0)

Magenta (1, 0, 1)

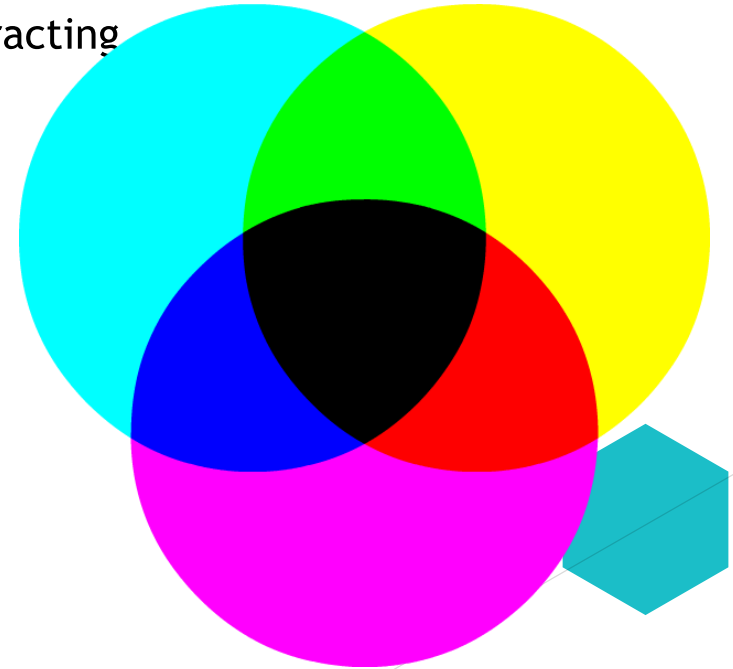
Blue (0, 0, 1)

Yellow (1, 1, 0)

RGB

CMY

- Three primary colors of color, suitable for printing
- In the CMY color model, Cyan, Magenta, Yellow which are the complementary colors of RGB. A method of subtracting RGB from white color.
  - Cyan = 1 - red (remain only green & blue)
  - Magenta = 1 - green (remain only red & blue)
  - Yellow = 1 - blue (remain only red & green)



CMY

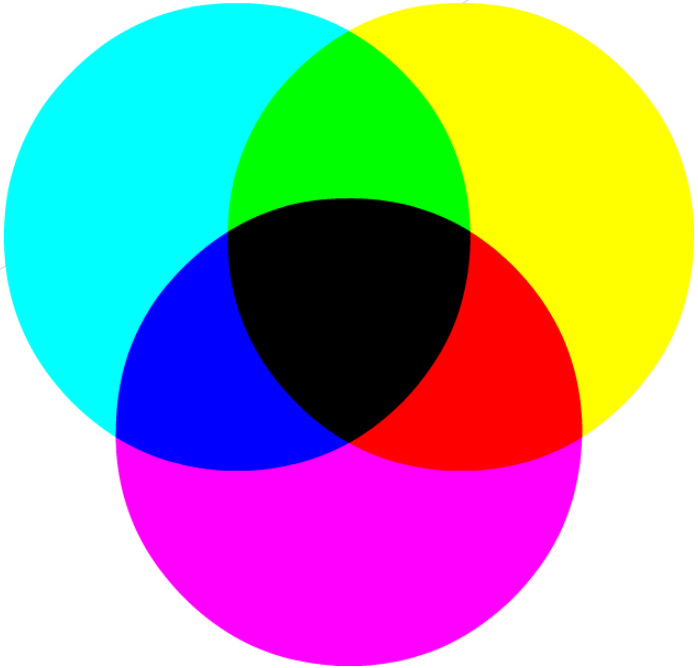
CMY

Black (1, 1, 1)

White (0, 0, 0)

Red (0, 1, 1)

Cyan(1, 0, 0)



Green (1, 0, 1)

Magenta (0, 1, 0)

Blue (1, 1, 0)

Yellow (0, 0, 1)

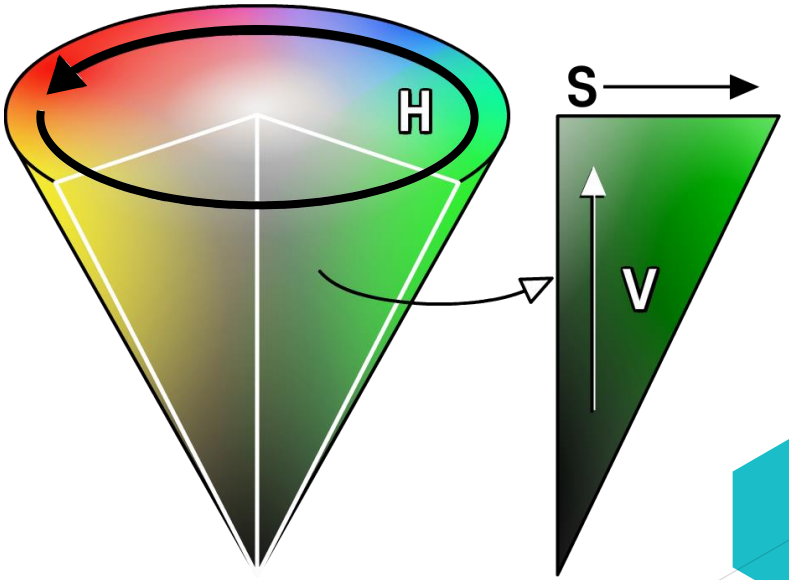
CMY

CMY

- RGB/CMY conversion
  - CMY = (1, 1, 1) - RGB
  - RGB = (1, 1, 1) - CMY
- CMYK is mainly used in printing by adding (Black, K)

HSV/HSB

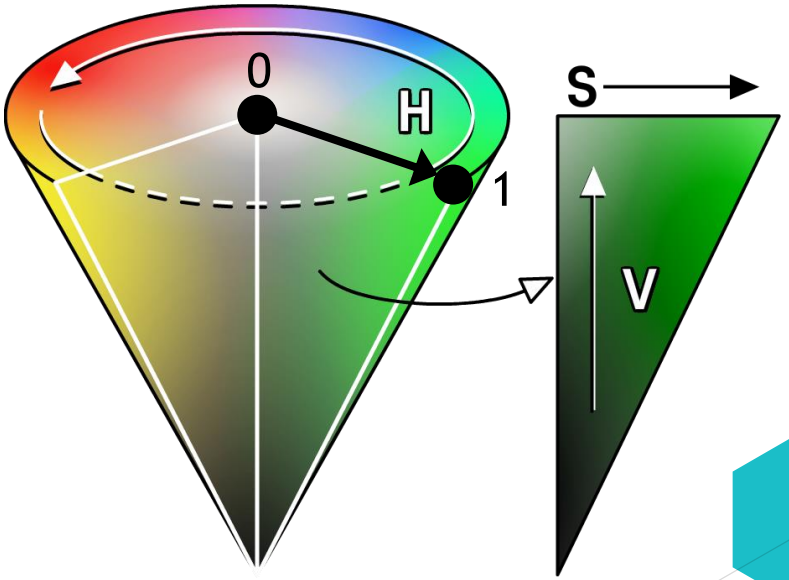
- The HSV color model is Hue, Saturation, Value/Brightness
- Color ranges from 0 to 360 degrees
  - 0 : red
  - 120 : green
  - 240 : blue



HSV/HSB

HSV/HSB

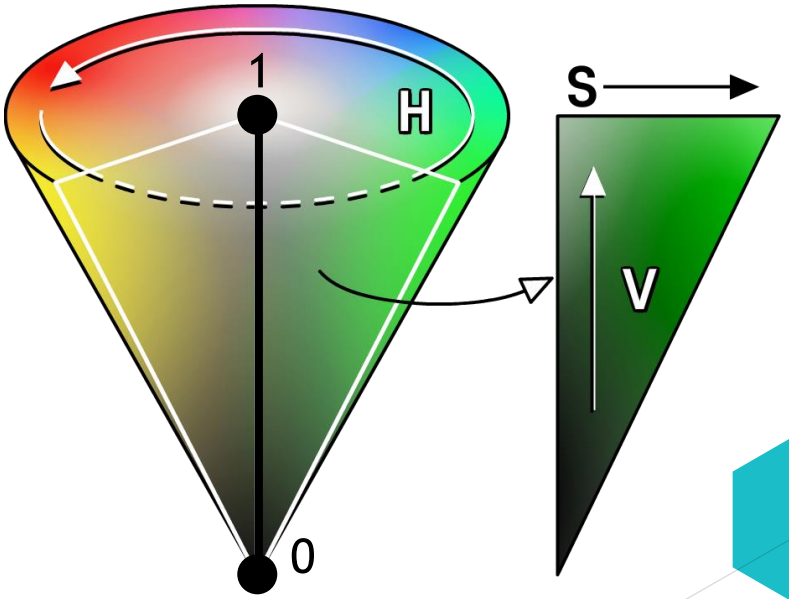
- The HSV color model is Hue, Saturation, Value/Brightness
- Saturation is a radius in the range of 0 to 1
  - 0 : Contrast of brightness (achromatic color)
  - 1 : Color is 100% saturated at the top edge of conical base



HSV/HSB

HSV/HSB

- The HSV color model is Hue, Saturation, Value/Brightness
- Brightness is the position on the z-axis
  - 0 : Black
  - 1 : White



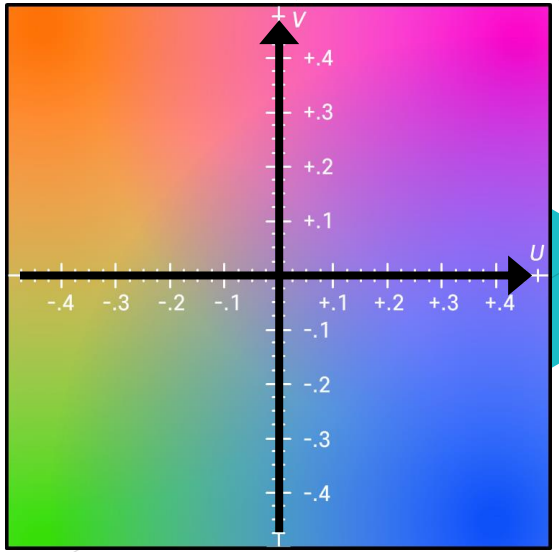
HSV/HSB

YUV/YIQ

- The YUV color model is a method of converting an RGB into a luminance (Y) representing the contrast of an image and two chrominance (U, V) on TV
  - Y : Brightness expressed as a number from 0 to 1
  - U : The degree of blue is expressed as a number from -0.5 to 0.5
  - V : The degree of red is expressed as a number from -0.5 to 0.5

V(빨간색 정도)

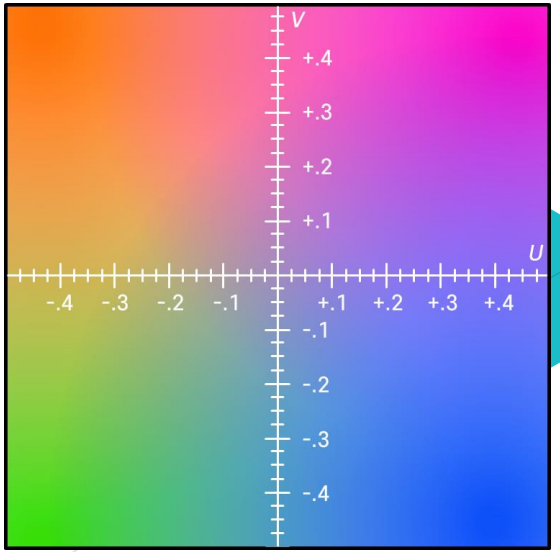
U(파란색 정도)



U-V Color Plane

YUV/YIQ

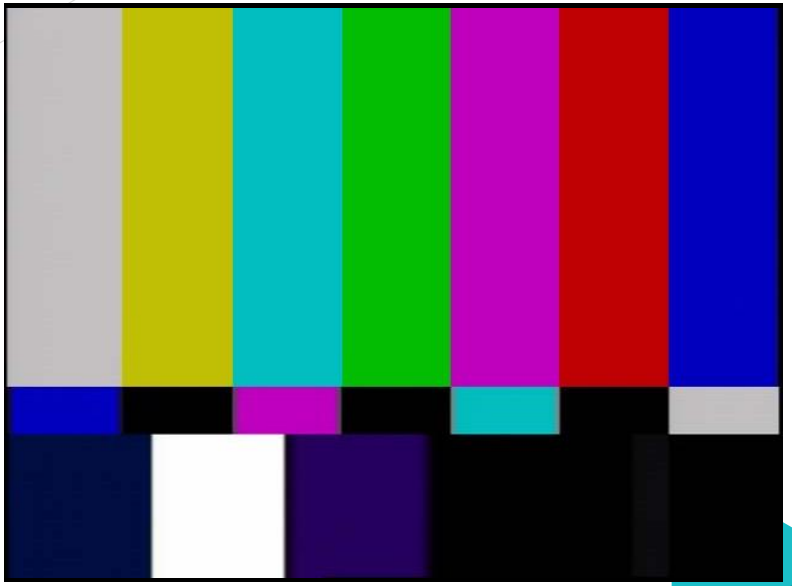
- RGB → YUV
  - Y = 0.299 R + 0.587 G + 0.114 B
  - U = -0.147 R - 0.289 G + 0.436 B
  - V = 0.615 R - 0.515 G - 0.100 B
- YUV → RGB
  - R = Y + 1.140 V
  - G = Y - (0.396 U + 0.581 V)
  - B = Y + 2.029 U



U-V Color Plane

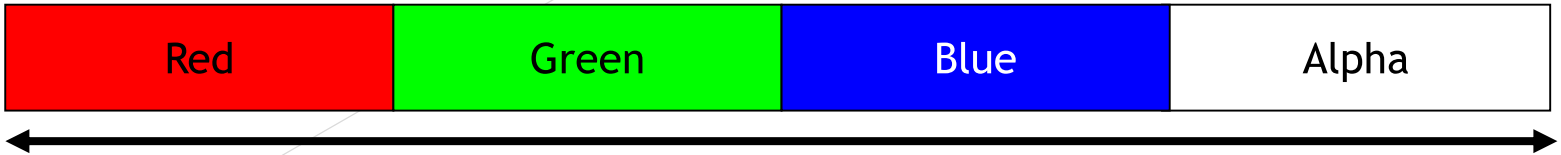
Luminance

- Color brightness
- Convert from RGB color to brightness
  - NTSC standard brightness
    $= 0.299 \times R + 0.587 \times G + 0.114 \times B$
    - According to NTSC standard
  - Average brightness
    $= 0.33 \times R + 0.33 \times G + 0.33 \times B$
    - Converting from RGB to HSV



Representing Colors

- Color structure in Unity
  - Red, Green, Blue, Alpha
    - Alpha component defines transparency
  - Each color component is a floating point value with a range from 0 to 1



Representing Colors

- Color structure in Unity
  - Static variables


black (0, 0, 0, 1)


clear (0, 0, 0, 0)


gray (0.5, 0.5, 0.5, 1)


magenta (1, 0, 1, 1)


white (1, 1, 1, 1)


blue (0, 0, 1, 1)


cyan (0, 1, 1, 1)


green (0, 1, 0, 1)


red (1, 0, 0, 1)

yellow (1, 0.92, 0.016, 1)

Representing Colors

- Color structure in Unity
  - Variables
    - r, g, b, a, gamma, grayscale, linear, maxColorComponent, this[int]
  - Static methods
    - HSVToRGB, Lerp, LerpUnclamped, RGBToHSV

Shading

- Shading is the process of determining the colors of the pixels in a primitive.

Flat shading

Flat shading fills the polygon face with the first vertex color

Gouraud shading

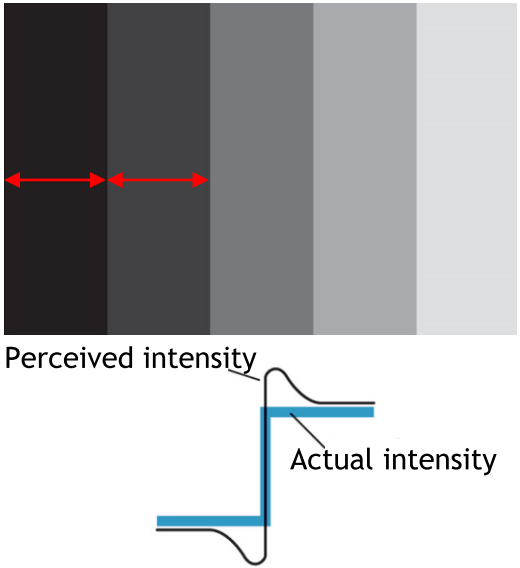
Gouraud shading (smooth shading) interpolates the polygon face with the colors at each vertex

Phone shading

Phone shading interpolates the polygon normal

Flat Shading

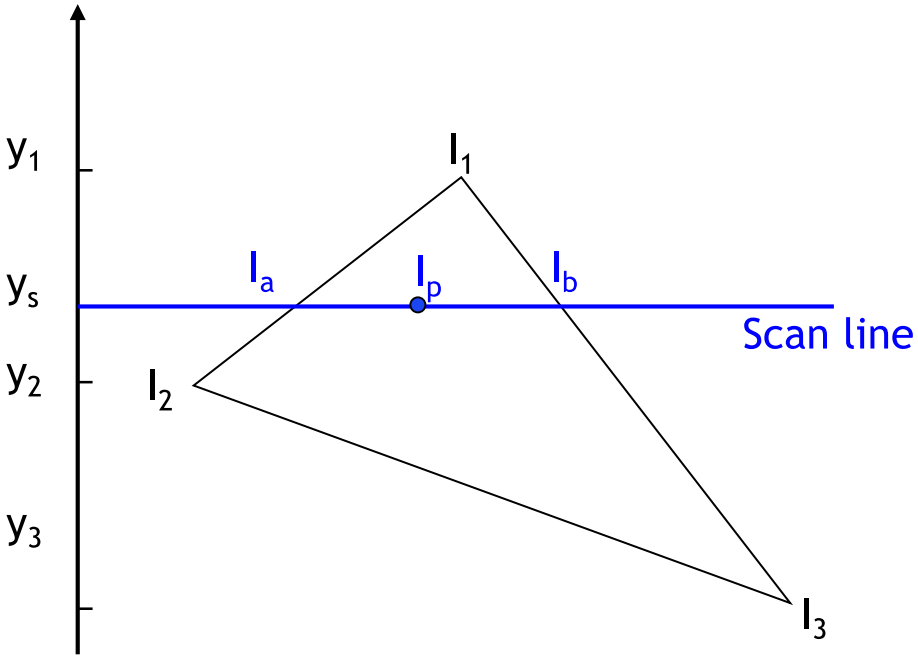
- The color of the entire given polygon painted with the same color.
- Also called constant shading, facet shading
- Mach band effect in flat shading - an optical phenomenon from edge enhancement due to lateral inhibition of the retina



이미지 출처 :https://en.Wikipedia.org/wiki/Utah_teapot

Gouraud Shading

- The color of a pixel is linearly interpolated using the colors of all vertices in the primitives



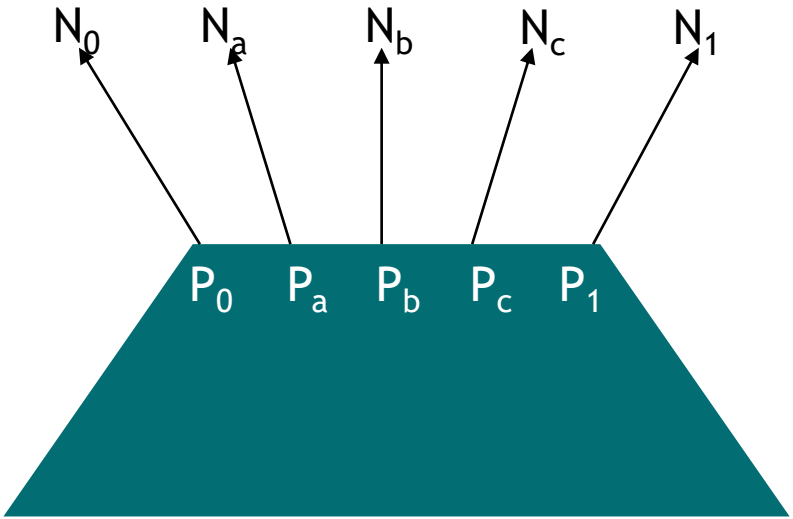
$$I_a = I_1 - (I_1 - I_2)\frac{y_1 - y_s}{y_1 - y_2}$$

$$I_b = I_1 - (I_1 - I_3)\frac{y_1 - y_s}{y_1 - y_3}$$

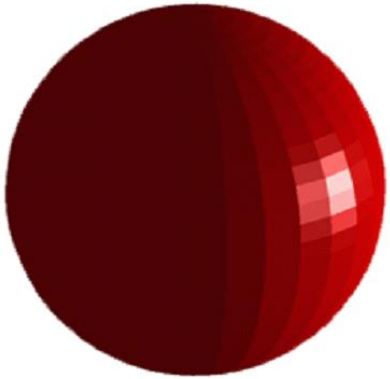
$$I_p = I_b - (I_b - I_a)\frac{x_b - x_p}{x_b - x_a}$$

Phong Shading

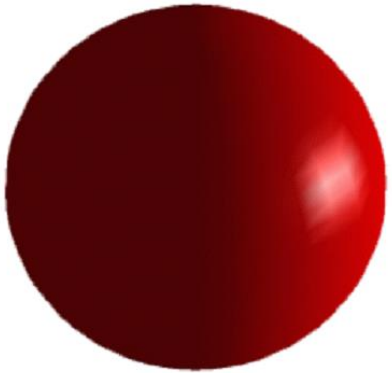
- Normal-vector interpolated shading
- Phong shading computes the normal vector for each pixel on the polygon



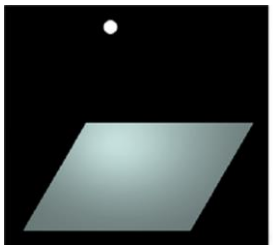
Flat, Gouraud, and Phong Shading

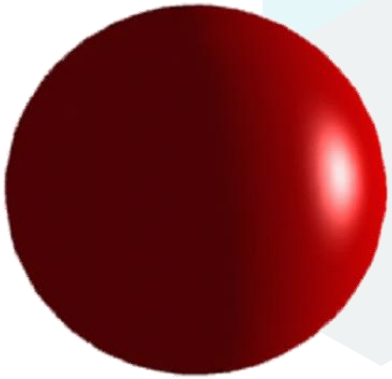


Flat Shading

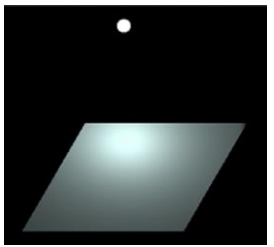


Gouraud Shading



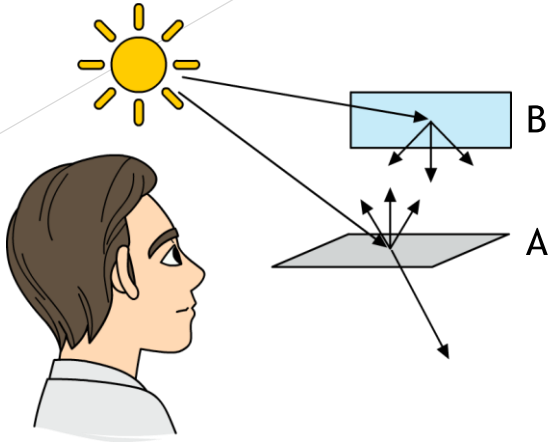


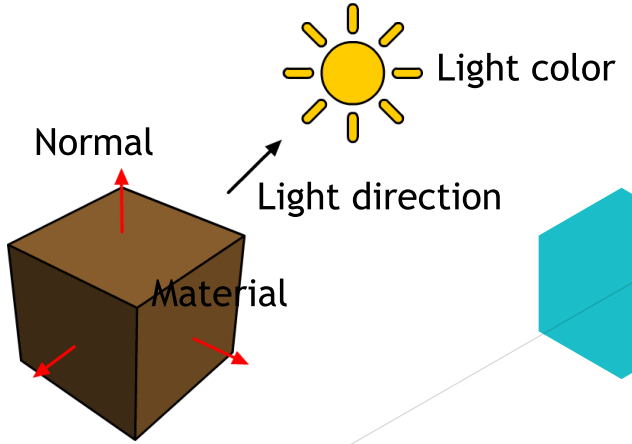
Phong Shading



Lighting

- Light starts at the lighting source
- The light strike on the surface, it
  - Absorption
  - Reflection
  - Transmission or Refraction
- Shading is determined by light source color, material properties, viewer location, surface orientation.





Lighting

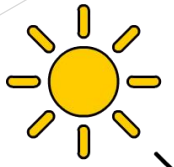
- Light Components

Lights

Materials

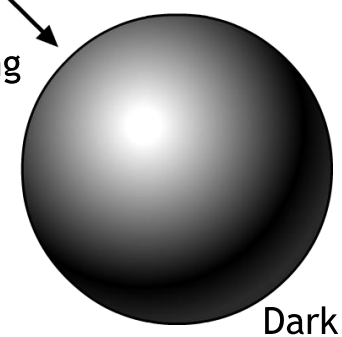
Vertex normals

Viewer



Light source

Bright shading



Dark shading

Light and Material Interaction

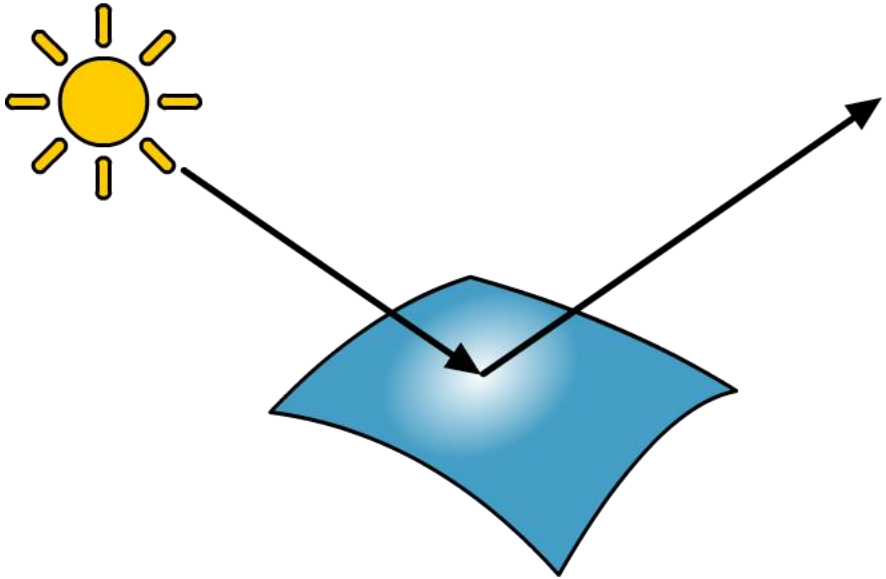
- When using lighting, we no longer specify vertex colors ourselves; rather, we specify materials and lights, and then, apply a lighting equation, which computes the vertex colors for us based on light/material interaction.

- Materials can be thought of as the properties that determine how light interacts with an object.

- We model lights by an additive mixture of red, green, and blue light (RGB); we can simulate many light colors.

Light-Material Interactions

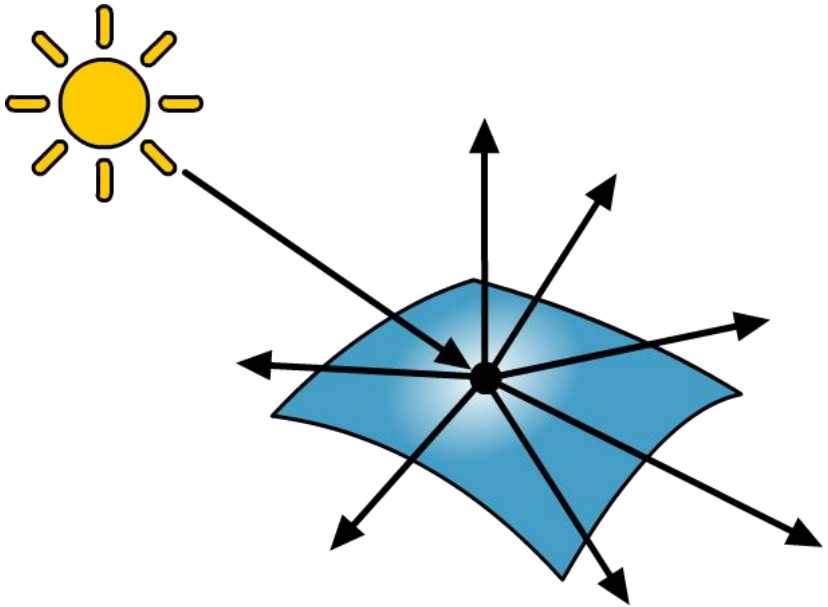
- Specular surface
  - The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflected the light.



smooth surface

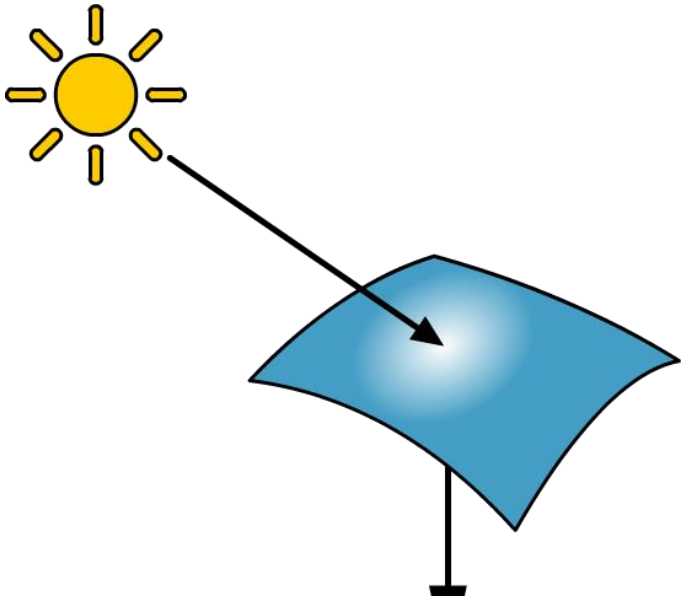
Light-Material Interactions

- Diffuse surface
  - A very rough surface scatters light in all directions.



Light-Material Interactions

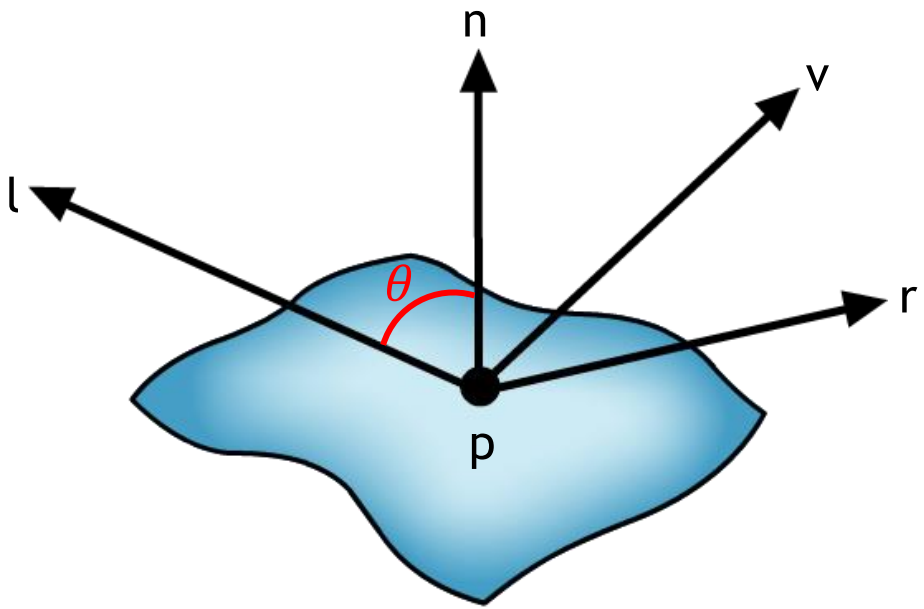
- Translucent surface
  - In a translucent surface, some light penetrates the surface and exist to other locations on the object (refraction)



Angle of Incidence

- The angle between the light source vector and the normal vector

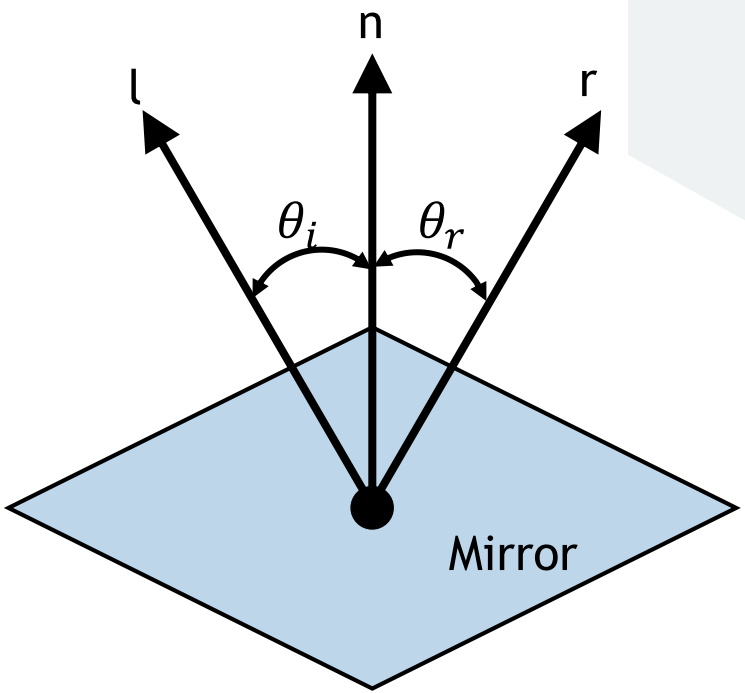
$$N \cdot L = \|N\|\|L\| cos\theta = (1)(1)\, cos\theta = cos\theta$$



Angle of Reflection

- The angle of incident and the angle of reflection are the same.

$$\theta_i = \theta_r$$



Projecting L onto N $=(\mathrm{L}\cdot\mathrm{N})\mathrm{N}$ when $\|\mathrm{N}\|=1$

Angle of Reflection

- The angle of incident and the angle of reflection are the same.

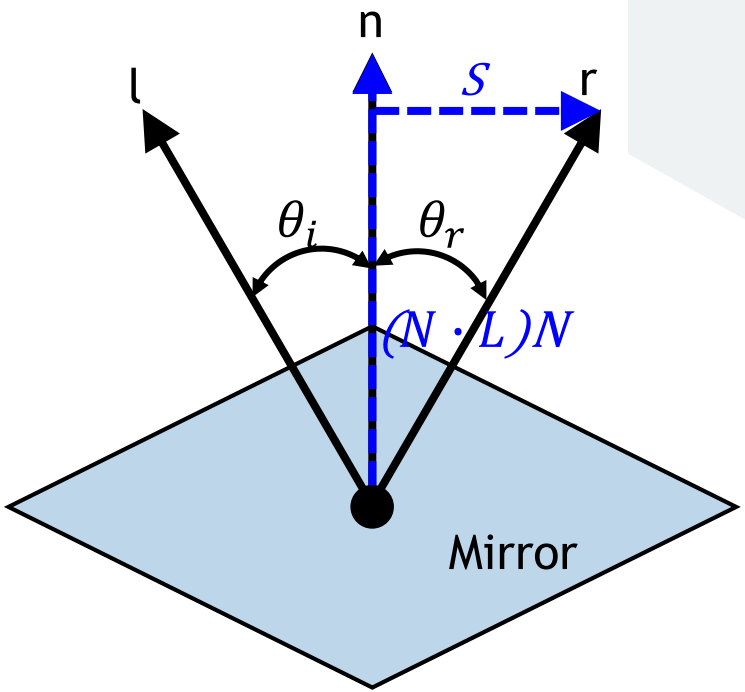
$$\theta_i = \theta_r$$

$$R = (N \cdot L)N + S$$

$$S = (N \cdot L)N - L$$

$$L = (N \cdot L)N - S$$

$$\Rightarrow R = 2(N \cdot L)N - L$$



Projecting L onto N $= (L \cdot N)N$ when $\|N\| = 1$

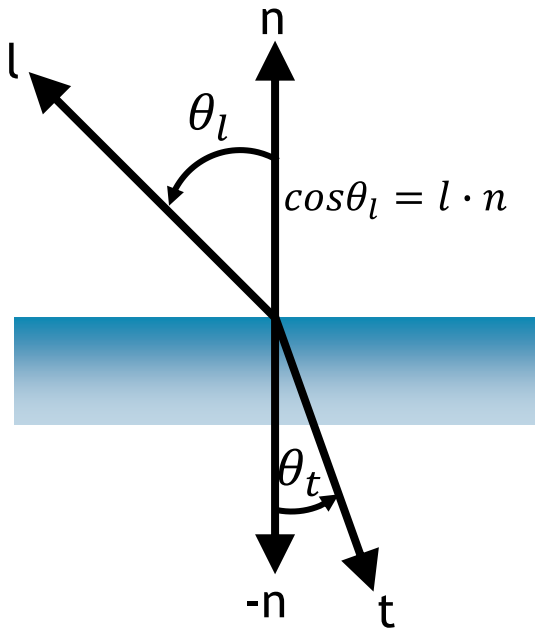
Indices of Refraction

- Refraction
  - η_l, η_t = the indices of refraction of two materials
- Snell's law

$$\frac{sin\theta_l}{sin\theta_t} = \frac{\eta_t}{\eta_l} = \eta$$

$$cos\theta_t = \left(1 - \frac{1}{\eta^2}(1 - cos^2\theta_l\right)^{\frac{1}{2}}$$

$$T = -\frac{1}{\eta}L - \left(cos\theta_t - \frac{1}{\eta}cos\theta_l\right)N$$



Perfect light transmission

Lighting Component

- Light
  - Ambient light
    - Ambient light comes from no particular direction. It reflects equally in all directions.
  - Diffuse light
    - The basic lighting effects is diffuse lighting. The intensity of diffuse lighting depends on the orientation of the surface relative to the light source. Diffuse light is reflected equally in all directions.
  - Specular light
    - Specular light is the light that is directly reflected off the surface to the camera. Its intensity depends on the orientation of the surface relative to camera, as well as to the light source.

Lighting Component

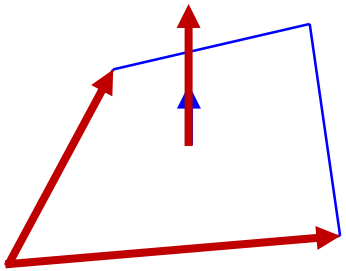
Materials

- Material properties define how a surface reflects light.
  Basically, they represent the surface color.
- When lighting is active, the material is used instead of color.
- Material
  - Surface diffuse/ambient/specular reflections
  - Emissive material (to make object appeared to be self-luminous)
  - Sharpness of specular reflection (higher value, smaller highlights)

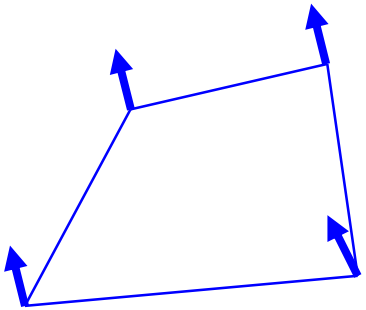
Materials

Vertex Normal

- Normal
  - Lighting computation uses vertex normals
- Vertex structure
  - Use normals instead of colors



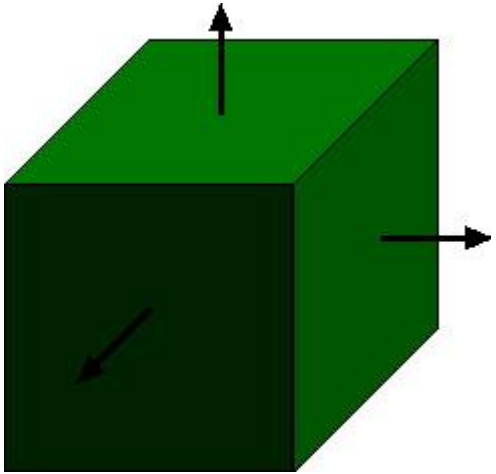
Face normal

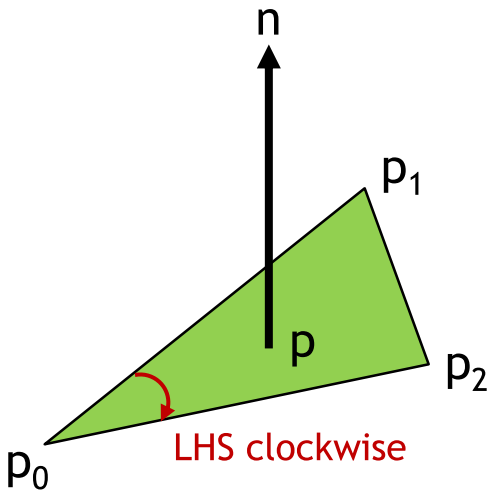


Vertex normal

Face Normal

- Compute the normal vector of a triangle consisting of vertex p_0, p_1, p_2





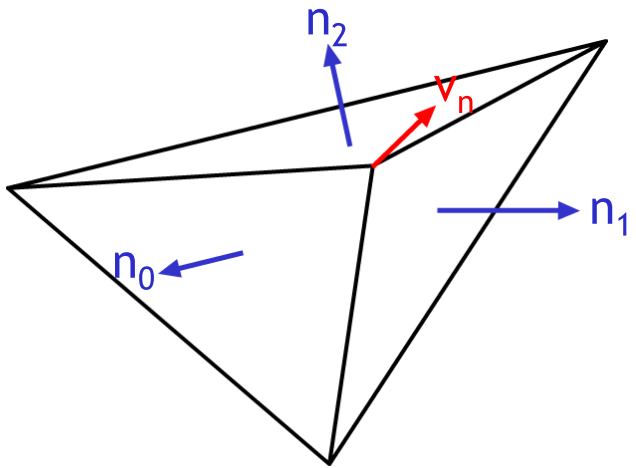
$$n \cdot (p - p_0) = 0$$

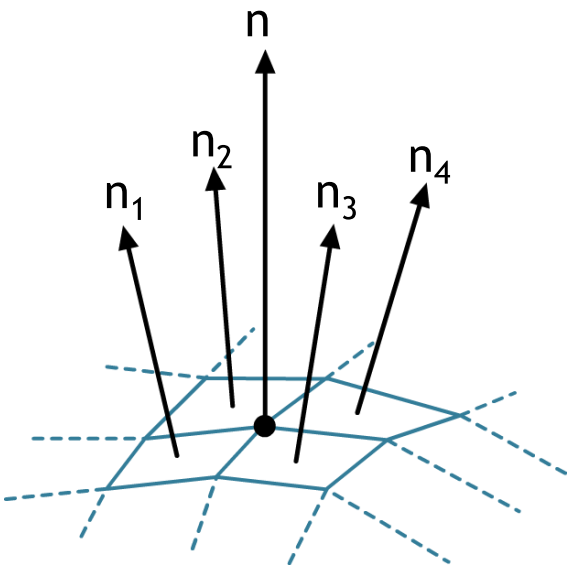
$$n = (p_1 - p_0) \times (p_2 - p_0)$$

normalize $n \leftarrow n/|n|$

Vertex Normal

- Vertex normal calculation using adjacent faces





$$n = \frac{n_1 + n_2 + n_3 + n_4}{|n_1 + n_2 + n_3 + n_4|}$$

Normal to Sphere

- Implicit function of Sphere

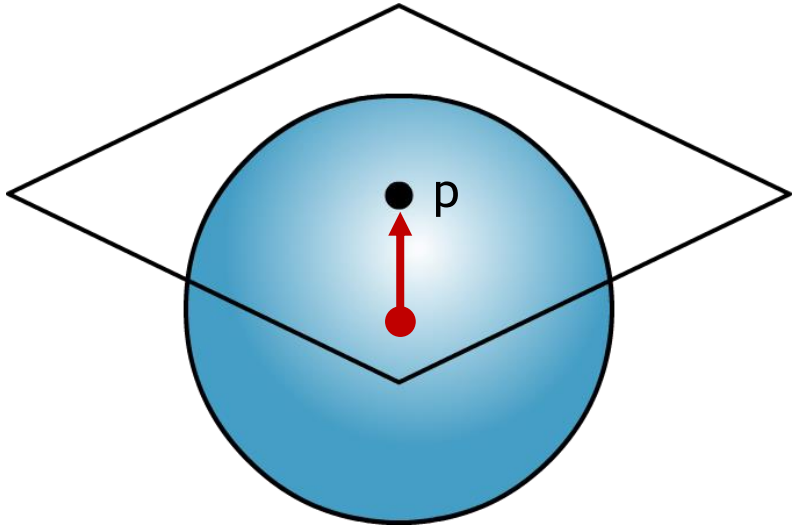
$$f(x, y, z) = 0$$

- Unit Sphere

$$f(p) = p \cdot p - 1 = 0$$

- Sphere Normal

$$n = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]^T = p$$



Ambient Lighting

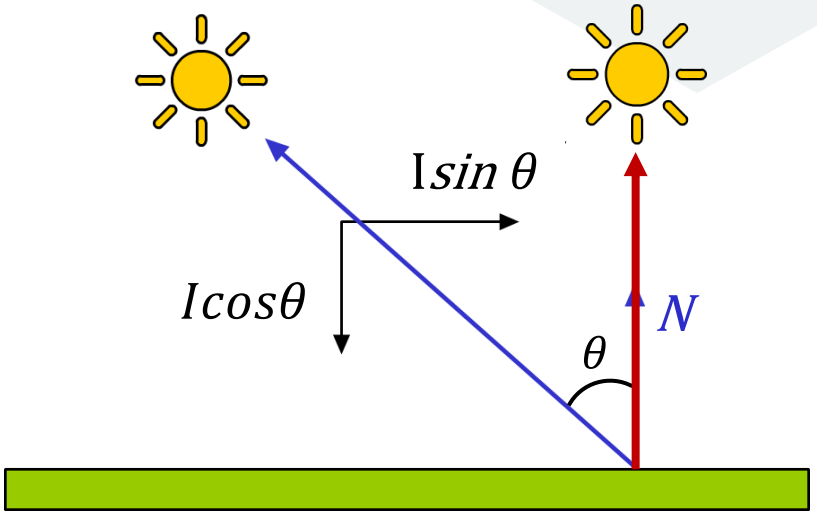
- Ambient light, also known as diffuse environmental light, is light that is present all around the Scene and doesn't come from any specific source object. It can be an important contributor to the overall look and brightness of a scene.

$$(L_a \otimes M_a)$$

- L_a : ambient light color
- M_a : ambient material color

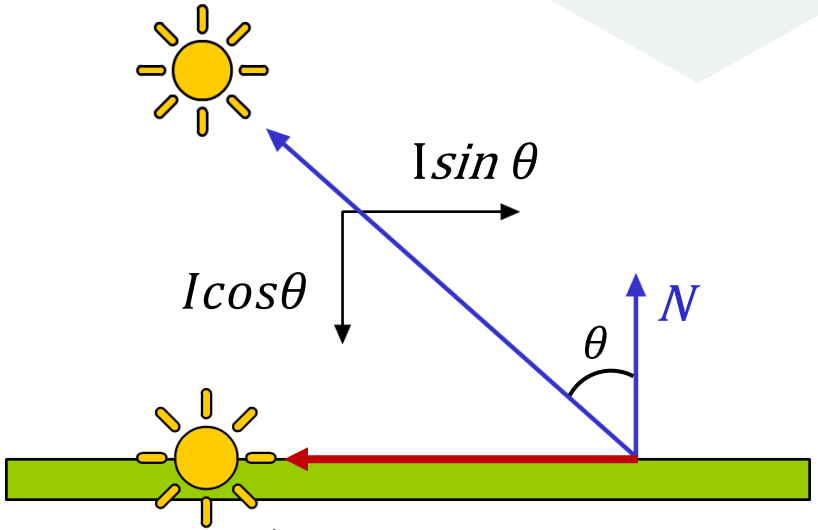
Diffuse Lighting

- Lambert's Cosine Law
  - θ between the normal and light vector
  - Maximum intensity when the normal and light vector are perfectly aligned (θ = 0)



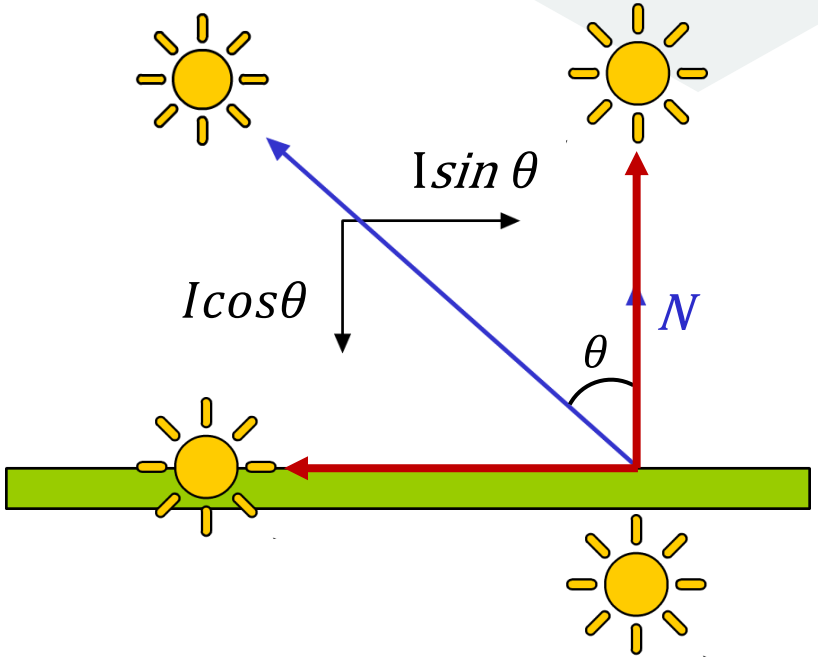
Diffuse Lighting

- Lambert's Cosine Law
  - θ between the normal and light vector
  - Maximum intensity when the normal and light vector are perfectly aligned (θ = 0)



Diffuse Lighting

- Lambert's Cosine Law
  - θ between the normal and light vector
  - Maximum intensity when the normal and light vector are perfectly aligned (θ = 0)



Diffuse Lighting

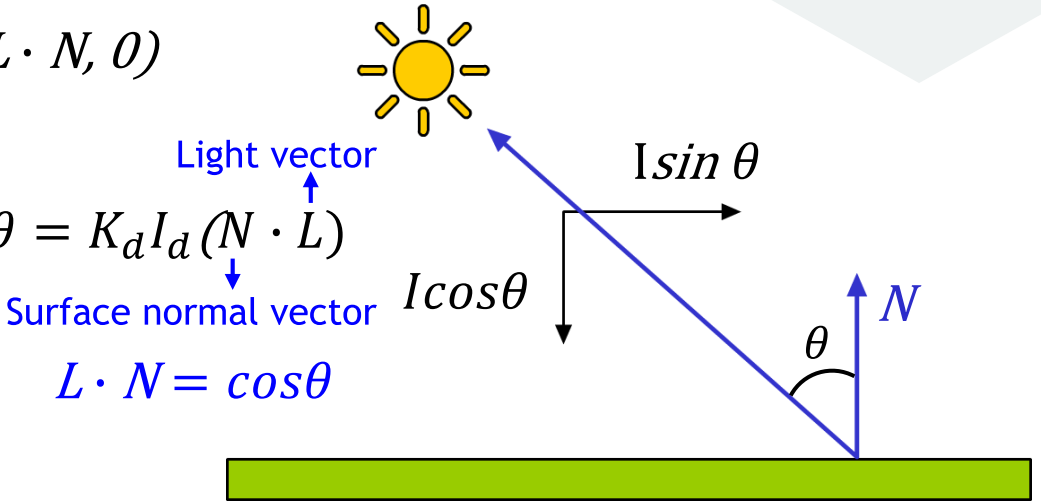
- Lambert's Cosine Law
  - θ between the normal and light vector
  - Maximum intensity when the normal and light vector are perfectly aligned (θ = 0)

$$f(\theta) = max(cos\theta, 0) = max(L \cdot N, 0)$$

$$Diffuse\ Reflection \propto cos\,\theta$$

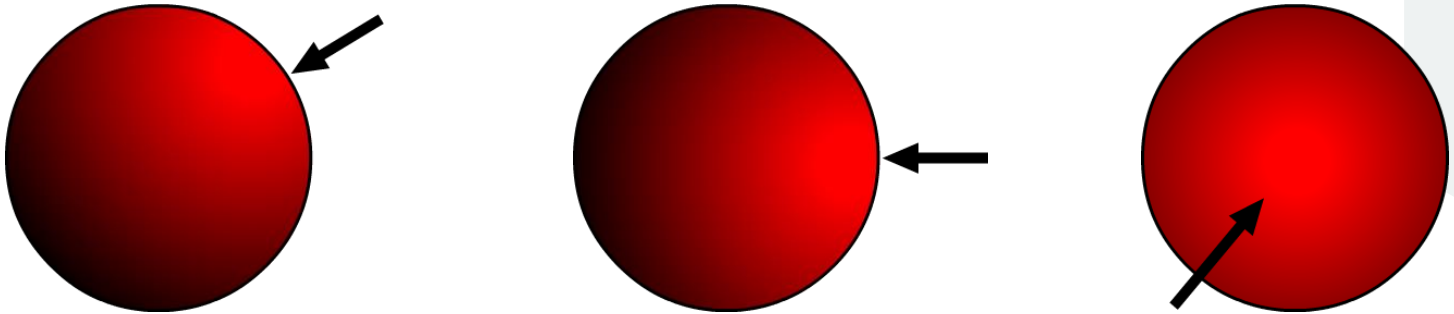
$$Diffuse\ Reflection \propto K_d I_d cos\,\theta = K_d I_d (N \cdot L)$$

- I_d : diffuse light intensity
- K_d : diffuse light coefficient



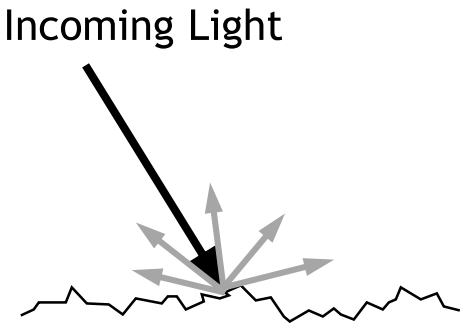
Diffuse Lighting

- Lambert's Cosine Law



Diffuse Lighting

- Diffuse lighting calculation (viewpoint independent)

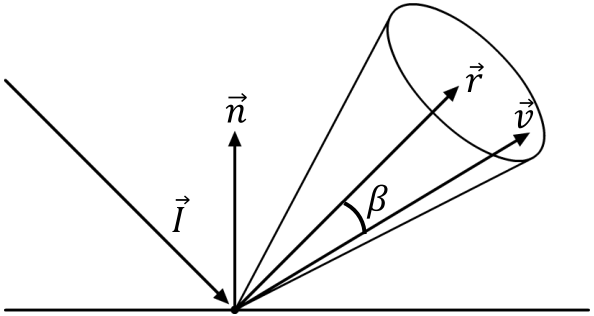


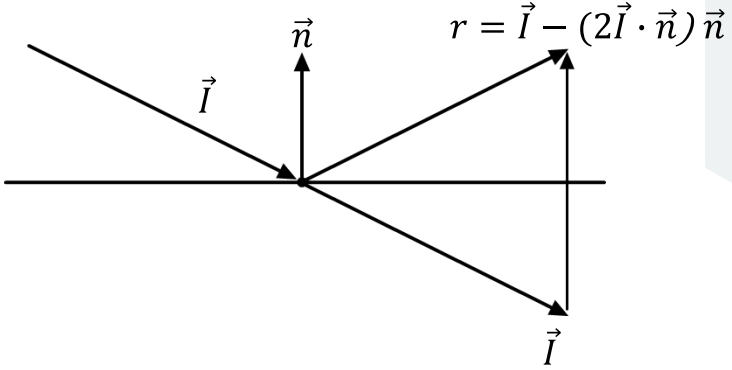
$$max(L \cdot N, 0) \cdot (L_a \otimes M_a)$$

- L_d : diffuse light color
- M_d : diffuse material color
- L : light vector
- N : vertex normal vector

Specular Lighting

- Specular lighting calculation (viewpoint dependent)



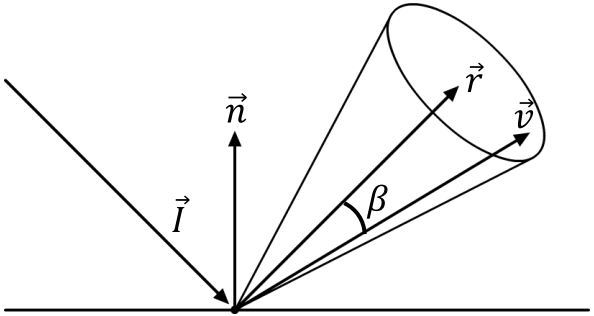


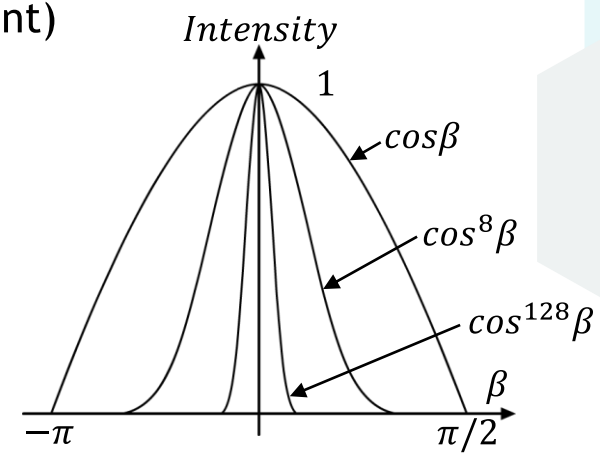
$$max(V \cdot R, 0)^p \cdot (L_s \otimes M_s)$$

- L_s : specular light color
- M_s : specular material color
- V : view vector
- R : light reflection vector, $R = L - (2L \cdot N)N$

Specular Lighting

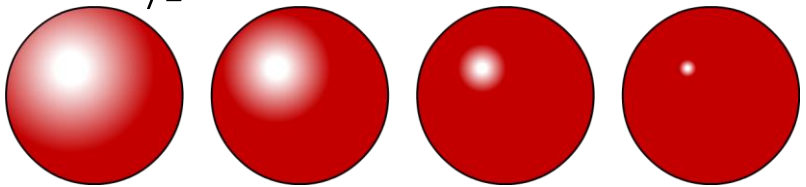
- Specular lighting calculation (viewpoint dependent)





$$max(V \cdot R, 0)^p \cdot (L_s \otimes M_s)$$

- L_s : specular light color
- M_s : specular material color
- V : view vector
- R : light reflection vector, $R = L - (2L \cdot N)N$



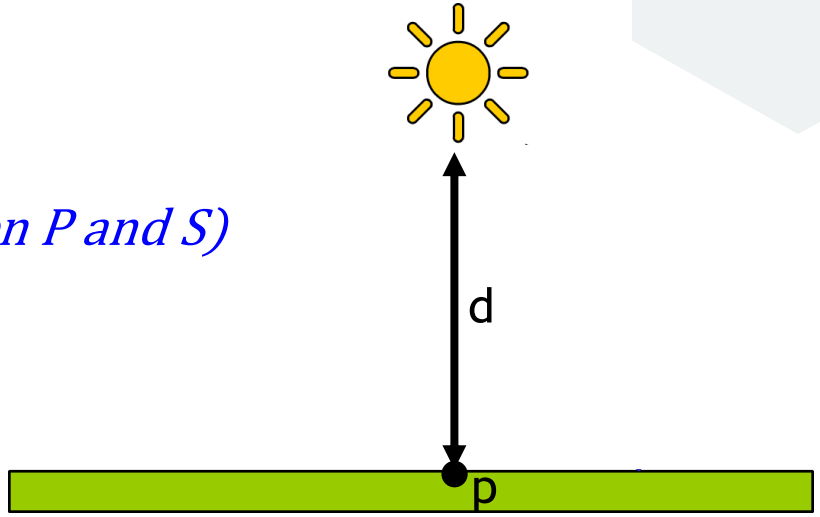
Light Attenuation

- Light intensity weakness as a function of distance based on the inverse square law.

$$I(d)=\frac{I_0}{d^2}$$

$$I(d)=\frac{I_0}{a_0+a_1 d+a_2 d^2}$$

$d = \|\mathrm{S} - \mathrm{P}\|$ *(i.e., the distance between P and S)*



Lighting in Unity

- Direct and Indirect lighting
  - Direct light is light that is emitted, hits a surface once, and is then reflected directly into a camera.
  - Indirect light is all other light that is ultimately reflected into a camera, including light that hits surfaces several times, and sky light.
- Real-time and Baked lighting
  - Real-time lighting is when Unity calculates lighting at runtime.
  - Baked lighting is when Unity performs lighting calculations in advance and saves the results as lighting data, which is then applied at runtime.

Lighting in Unity

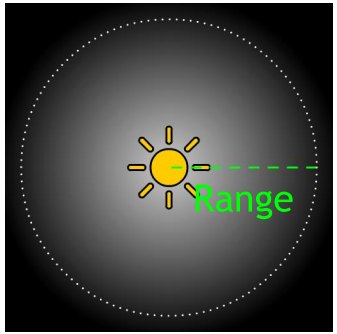
Lighting in Unity

- Global Illumination
  - Global illumination is a group of techniques that model both direct and indirect lighting to provide realistic lighting results.

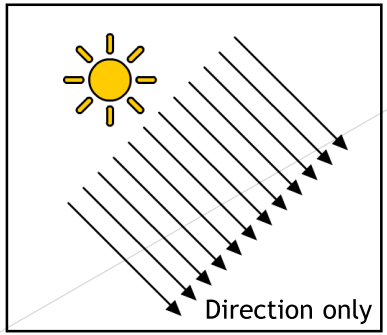
Lighting in Unity

Light Sources

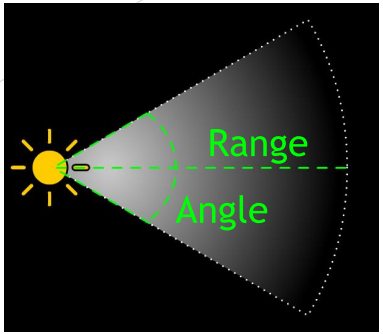
Light sources



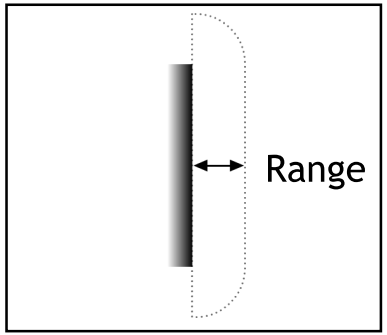
Point light



Directional light



Spot light

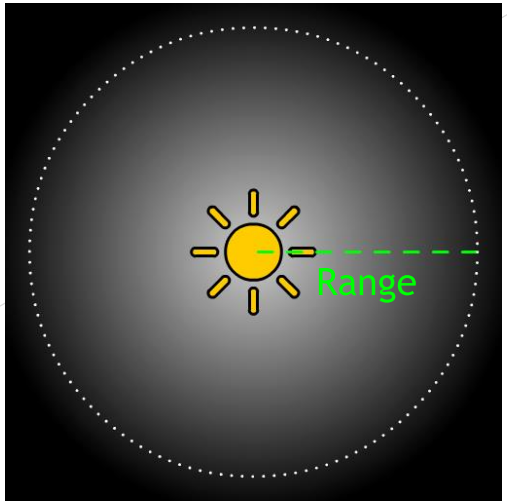


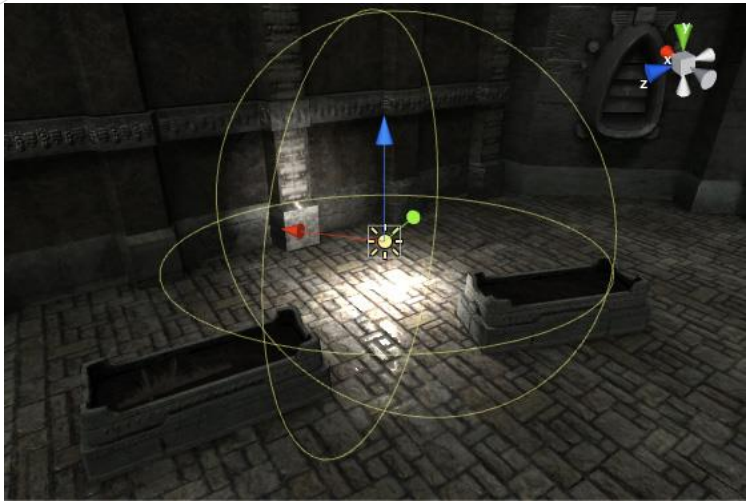
Area light

이미지 출처 :https://docs.unity3d.com/2019.4/Documentation/Manual/Lighting.html

Point Light

- A point light is located at a point in space and sends light out in all directions equally.
- The intensity diminishes with distance from the light, reaching zero at a specified range. Light intensity is inversely proportional to the square of the distance from the source.

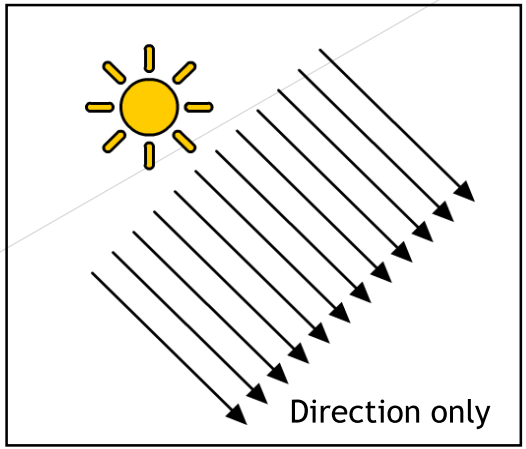


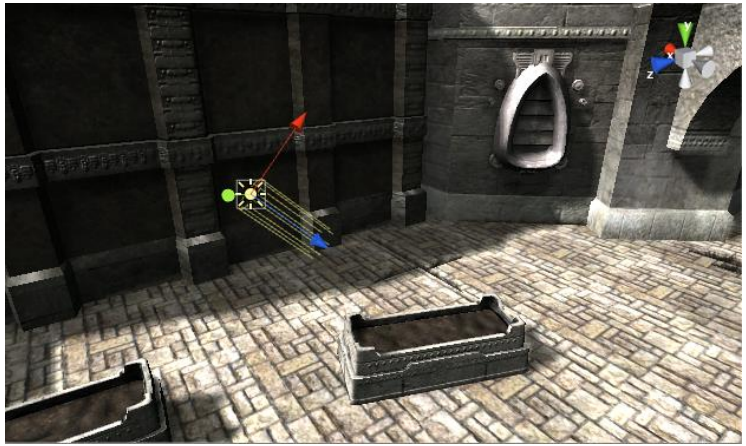


이미지 출처 :https://docs.unity3d.com/2019.4/Documentation/Manual/Lighting.html

Directional Light

- Directional lights can be thought of as distant light sources which exist infinitely far away. A directional light does not have any identifiable source position and so the light object can be placed anywhere in the scene.
- All objects in the scene are illuminated as if the light is always from the same direction. The distance of the light from the target object is not defined and so the light does not diminish.

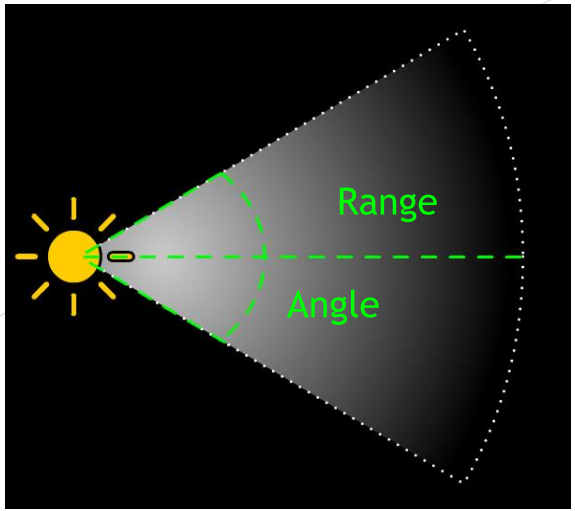


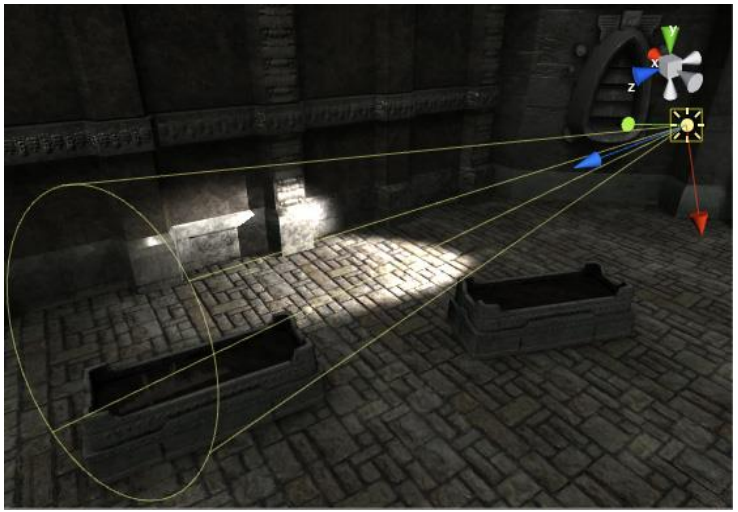


이미지 출처 :https://docs.unity3d.com/2019.4/Documentation/Manual/Lighting.html

Spot Light

- Spot lights are generally used for artificial light sources such as flashlights, car headlights and searchlights.
- With the direction controlled from a script or animation, a moving Spot Light will illuminate just a small area of the scene and create dramatic lighting effects.

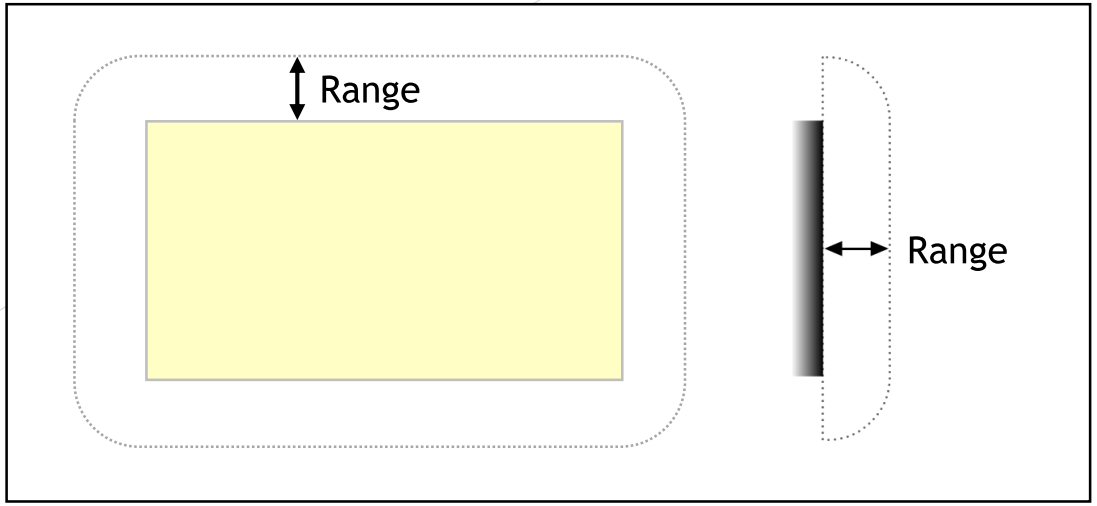


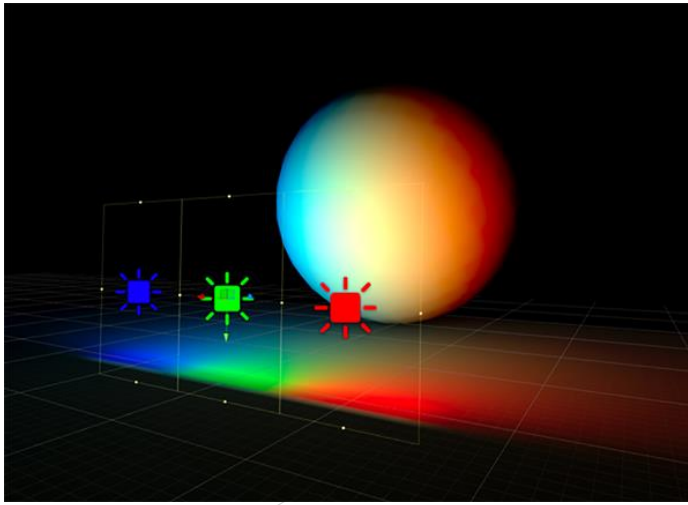


이미지 출처 :https://docs.unity3d.com/2019.4/Documentation/Manual/Lighting.html

Area Light

- An area light emits light from one side of rectangle (or disc). The emitted light spreads uniformly in all directions across that shape's surface area.
- The Range property determines the size of that shape. The intensity of the illumination provided by an area light diminishes at a rate determined by the inverse square of the distance from the light source.





이미지 출처 :https://docs.unity3d.com/2019.4/Documentation/Manual/Lighting.html

Light Modes

- Unity performs the lighting calculations for Real-time Lights at runtime, once per frame. You can change the properties of Real-time Lights at runtime to create effects such as flickering light bulbs, or a torch being carried through a dark room.

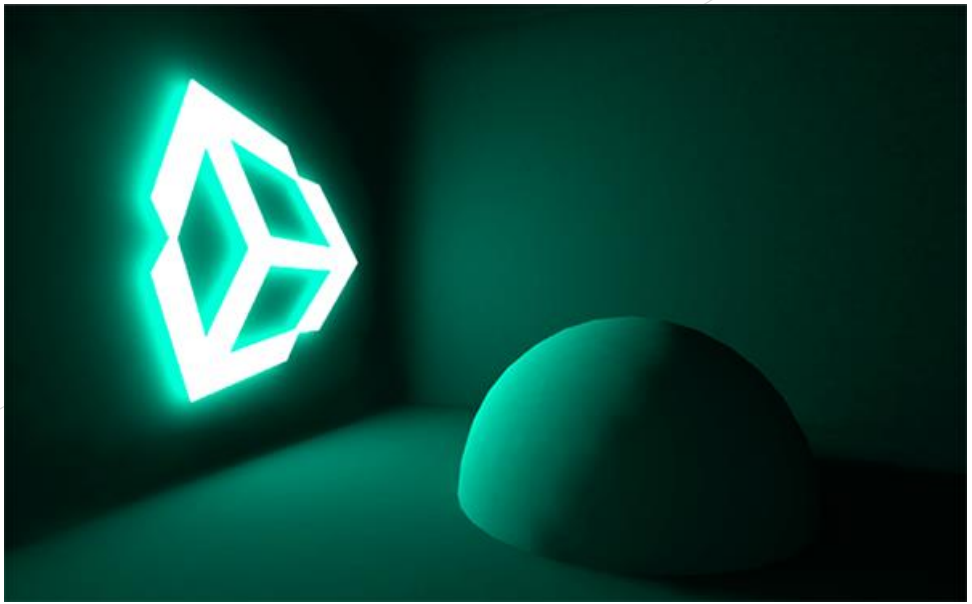
- Unity performs the calculations for Baked Lights in the Unity Editor, and saves the results to disk as lighting data. This process is called baking. At runtime, Unity loads the baked lighting data, and uses it to light the Scene. Because the complex calculations are performed in advance, Baked Lights reduce shading cost at runtime, and reduce the rendering cost of shadows.

- Mixed Lights combine elements of both real-time and baked lighting. You can use Mixed Lights to combine dynamic shadows with baked lighting from the same light source, or when you want a light to contribute direct real-time lighting and baked indirect lighting.

Light Modes

Emissive Materials

- Materials with an ‘Emission’ above zero will still appear to glow brightly on-screen even if they are not contributing to scene lighting.
- Emissive materials only directly affect static geometry in your scene in Unity.



이미지 출처 :https://docs.unity3d.com/Manual/lighting-emissive-materials.html

Emissive Materials

- Like area lights, emissive materials emit light across their surface area. They contribute to bounced light in your scene and associated properties such as color and intensity can be changed during gameplay. Whilst area lights are not supported by real-time Global Illumination, similar soft lighting effects in real-time are still possible using emissive materials.

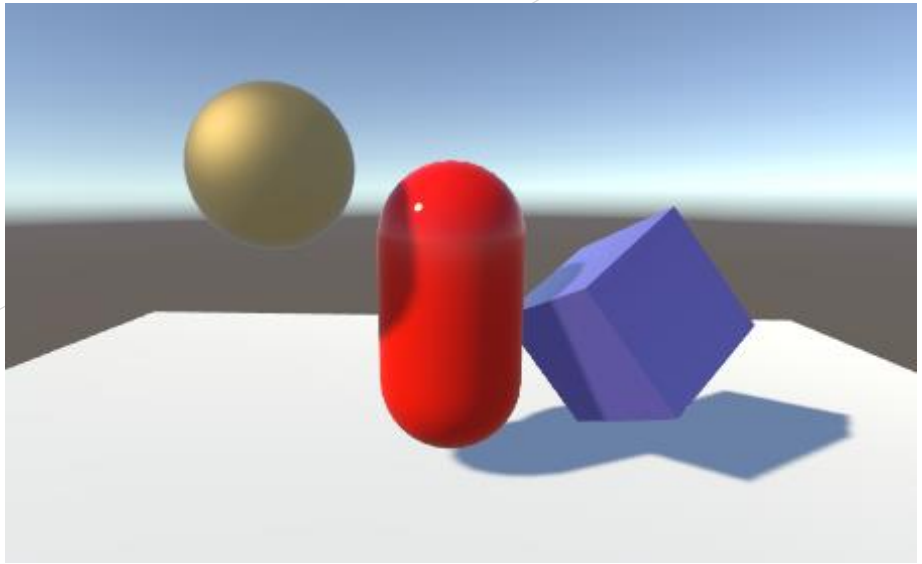
- ‘Emission’ is a property of the Standard Shader which allows static objects in our scene to emit light. By default the value of ‘Emission’ is set to zero. This means no light will be emitted by objects assigned materials using the Standard Shader.

- There is no range value for emissive materials but light emitted will again falloff at a quadratic rate. Emission will only be received by objects marked as ‘Static’ or ‘Lightmap Static’ from the Inspector. Similarly, emissive materials applied to non-static, or dynamic geometry such as characters will not contribute to scene lighting.

Emissive Materials

Shadow

- Shadows add a degree of depth and realism to a Scene because they bring out the scale and position of objects that might otherwise look flat.
- In Unity, Lights can cast shadows from a GameObject onto other parts of itself, or onto nearby GameObjects.



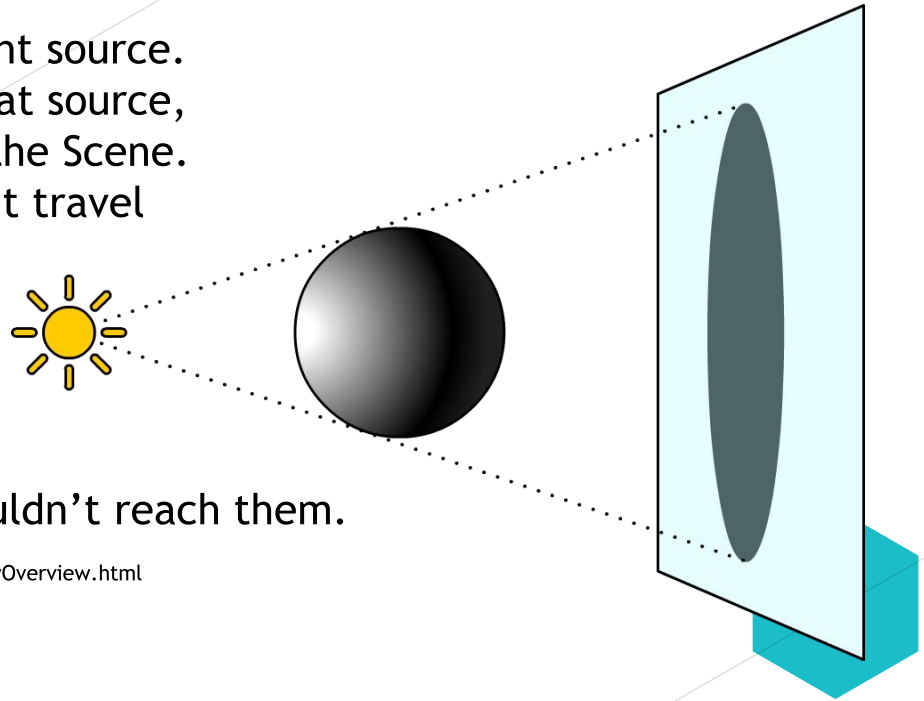
Scene with objects casting shadows

이미지 출처 :https://docs.unity3d.com/2019.4/Documentation/Manual/Lighting.html

Shadow

How do Shadows work?

- Consider a simple Scene with a single light source. Light rays travel in straight lines from that source, and may eventually hit GameObjects in the Scene. Once a ray has hit a GameObject, it can't travel any further to illuminate anything else (that is, it "bounces" off the first GameObject and doesn't pass through). The shadows cast by the GameObject are simply the areas that are not illuminated because the light couldn't reach them.

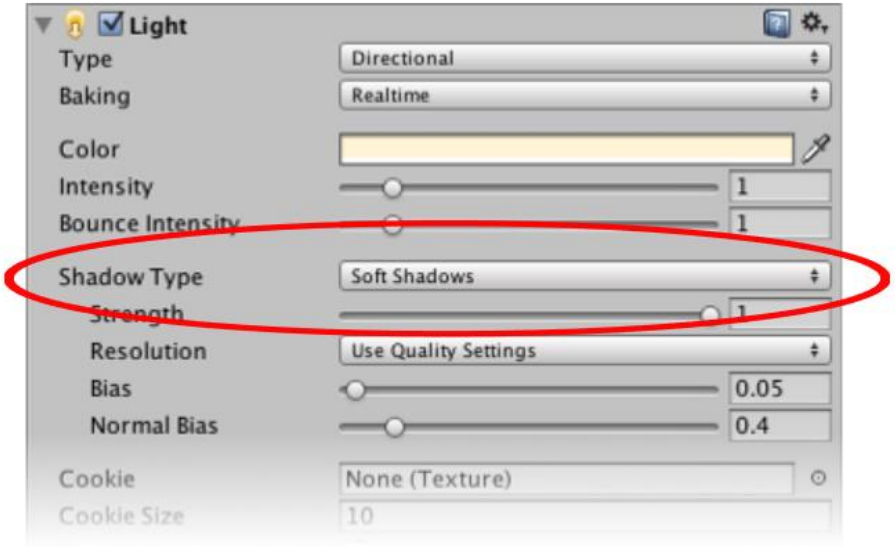


이미지 출처 :https://docs.unity3d.com/2018.4/Documentation/Manual/ShadowOverview.html

Shadow

Enabling Shadows

- Use the Shadow Type property in the Inspector to enable and define shadows for an individual light.
  - The Hard Shadows setting produces shadows with a sharp edge. Hard shadows are not particularly realistic compared to Soft Shadows but they involve less processing, and are acceptable for many purposes. Soft shadows also tend to reduce the “blocky” aliasing effect from the shadow map.

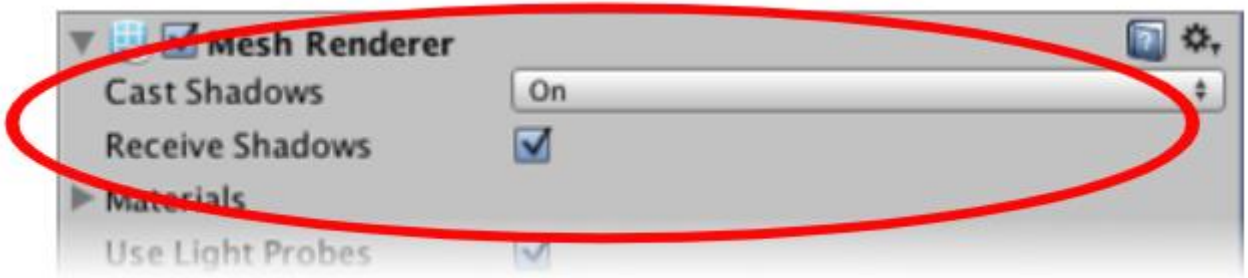


이미지 출처 :Unity

Shadow

Enabling Shadows

- Each Mesh Renderer in the Scene also has a Cast Shadows and a Receive Shadows property, which must be enabled as appropriate.



이미지 출처 :Unity

- Enable Cast Shadows by selecting On from the drop-down menu to enable or disable shadow casting for the mesh.
- Alternatively, select Two Sided to allow shadows to be cast by either side of the surface (so backface culling is ignored for shadow casting purposes), or Shadows Only to allow shadows to be cast by an invisible GameObject.

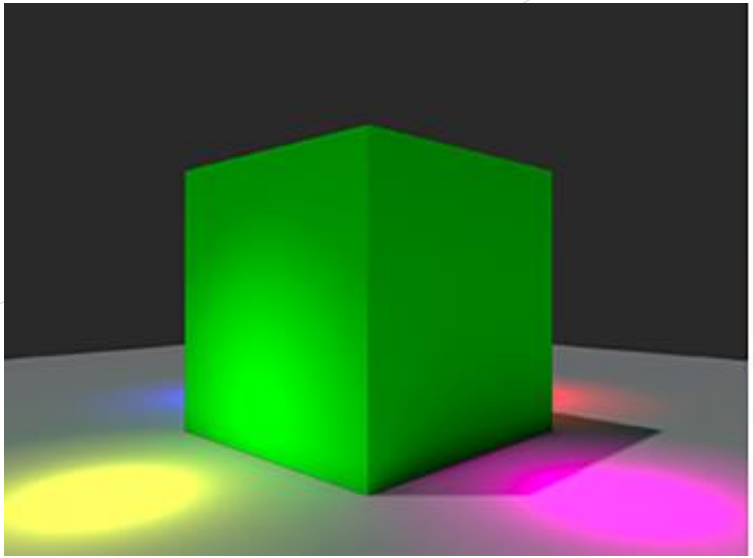
Global Illumination (GI)

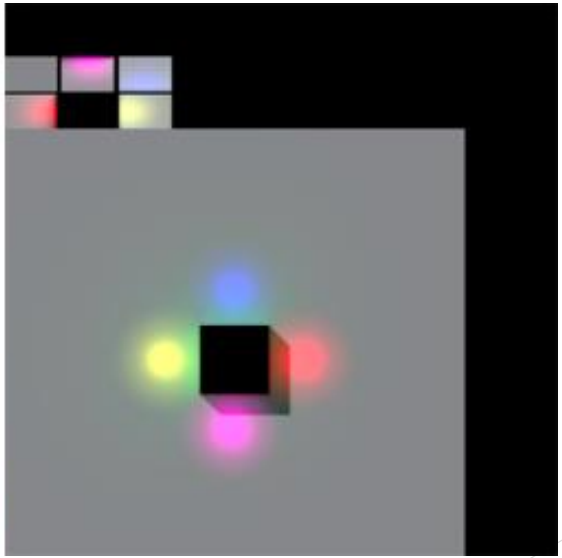
- Unity uses middleware called Enlighten for Realtime GI.
- By default, Real-time Lights contribute only direct lighting to a Scene.
  If you enable Real-time Global Illumination (Real-time GI) in your Scene,
  Real-time Lights also contribute indirect lighting to a Scene.

이미지 출처 :https://docs.unity3d.com/2018.4/Documentation/Manual/GIIntro.html

Lightmapping

- Lightmapping is the process of pre-calculating the brightness of surfaces in a Scene, and storing the result in a Texture called a lightmap.
- Lightmaps can include both direct and indirect light. This lighting texture can be used together with surface information like color (albedo) and relief (normals) by the Shader associated with an object's material.

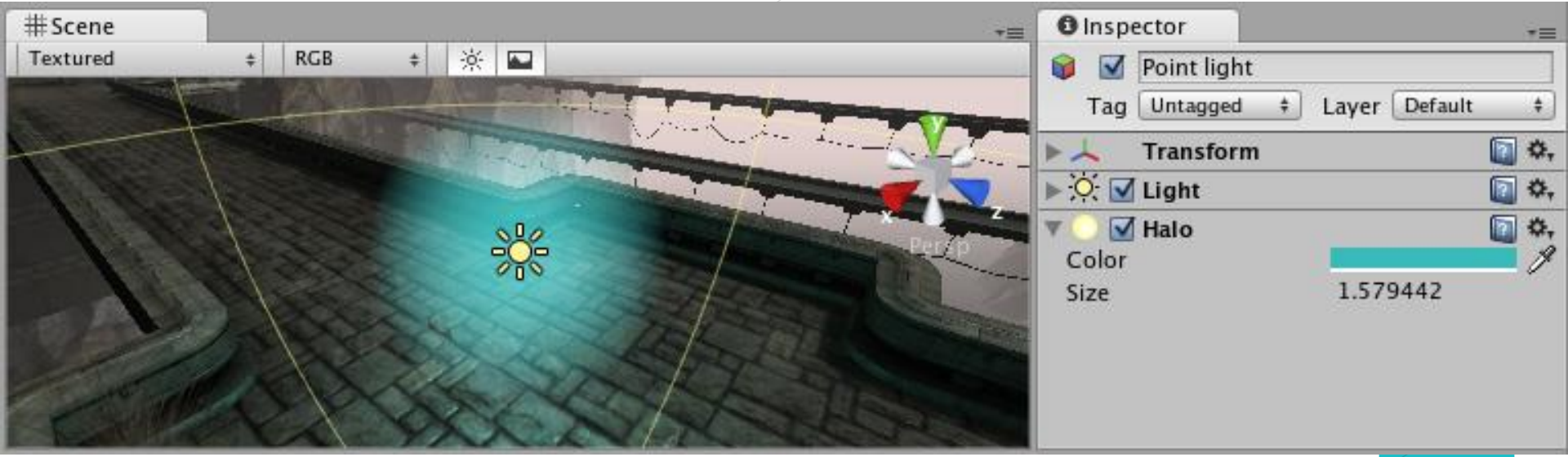




이미지 출처 :https://docs.unity3d.com/Manual/Lightmappers.html

Halo

- Halos are glowing areas around light sources. Use them to give the impression of small dust particles in the air, and add atmosphere to your scene.



이미지 출처 :https://docs.unity3d.com/Manual/class-Halo.html

Lens Flare

- A Lens Flare component displays a lens flare that is configured by a Flare asset.
- You can display a Flare asset with a Light component. If you do this, Unity automatically tracks the position and direction of the Light and uses those values to configure the appearance of the lens flare.

이미지 출처 :https://docs.unity3d.com/Manual/visual-effects-lens-flares.html

Reference

- https://docs.unity3d.com/540/Documentation/ScriptReference/Color.html
- https://www.youtube.com/watch?v=u5DNkxkXBeI Lights - Unity Official Tutorials
- https://docs.unity3d.com/Manual/LightingOverview.html
- https://www.theunitygamedev.com/2020/09/08/unity-2020-lighting-for-beginners/
- https://docs.unity3d.com/ScriptReference/Color.html