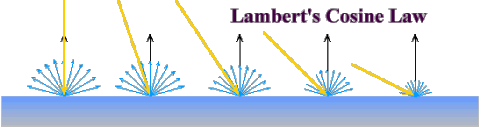
# Phong Lighting

The following formula describes Phong lighting

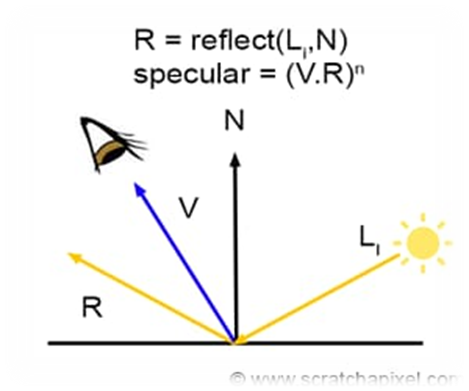


This formula is broken up into three main parts, the first part outside the summation is to do with ambient light which is the same intensity on every surface to be drawn. The other two parts within the summation are for the diffuse and specular intensity for a given pixel at a given position. The diffuse reflection is governed by Lambert’s cosine law which is described below and the specular intensity is governed by Snell’s law which follows Lambert’s law.



Lambert’s Cosine law relates to diffuse or matte type reflections, i.e. chalk like surfaces as opposed to shiny surfaces. Here the light usually denoted by L is compared to the normal of the surface usually denoted by N. Clearly The closer L and N are to being perfectly aligned the brighter the intensity of the reflection should be. Mathematically this can be described as the intensity will be proportional to the cosine of the angle between these two vectors. As the L of the light vector and N normal to the surface are unit vectors we can replace the cosine of the angle between them with the scalar dot product of the two vectors in question. The intensity at a given pixel or point on a surface must then be totalled for every light and that is how the law is implemented in the above expression.

# Snells Law



Snells Law, Is the principle behind specular reflection. Specular reflection is effectively the placement of specular highlights are the “shiny dots” which are the reflections of the light sources on smooth surfaces. Here the light ray comes in and is reflected from the surface. This is calculated by the reflected ray having the same angle relative to the normal as the incoming light ray And all being on the same plane. Viewer will see the intensity at the point in question from its perspective which is described by the vector V, and the closer V is to R will determine the intensity of light at that point IE if V were to coincide exactly with R then it would be at its most brightest and as V deviates from R the intensity will diminish. In other words the intensity is related to the cosine of the angle between V and R and as we have seen before this can be evaluated by getting the scalar dot product of V and R again V and R are unit vectors. The intensity Our sharpness of the specular highlight depends on the shyness of the surface very shiny surfaces would have a pinpoint reflection and dollar surfaces would have a less pinpoint or more spread out specular highlight. As all values of The scalar product with R Are between zero and one raising these values to higher powers of N will concentrate the values

# Interaction between rasterization, the frame buffer, texturing, shading and hidden surface removal

Rasterization is the process of turning a pixel on for a given Primitive which identifies a location within The frame buffer which is a section of memory which has an allocation for each pixel of the proposed screen. Once the pixel is identified the appropriate colour from the texture map may be calculated or found. The final shade or colour of the pixel the lighting algorithm must be used to determine the level or intensity of light at that given pixel. Once the appropriate colour has been identified the hidden surface removal or the Z buffer algorithm is queried to determine whether the actual location of the frame buffer is to be assigned the appropriate data for the colour. Obviously this is so the Z buffer must also be updated with the appropriate depth of the surface that that pixel is on.

# Hidden Surface removal With the Z buffer algorithm

Hidden surface removal or hsr is an algorithm that determines which pixels should be drawn when Objects in a scene overlap each other from the perspective of the camera. In practise the graphics pipeline will draw all objects in the scene and clearly is possible that one object would obscure another. If not dealt with it is possible that an object which is behind or obscured by another object will overwrite the pixels of the object in front of it.

The Z Buffer algorithm is the algorithm of choice for hsr. To implement this algorithm we set the frame buffer to the background colour and the Z buffer to the maximum distance of the viewing frustrum.

When rasterizing a given primitive Pixel positions are determined Texturing and shading determines the colour for that pixel position. The depth of the defining vertices of the primitive would be known as rasterization and therefore the depth of each pixel can be calculated from these. Once the colour and the depth of the pixel to be drawn is known we query the depth buffer To determine if any other pixel which has a depth less than depth of the current pixel Has already been drawn if so we do nothing and leave both the frame buffer and the depth buffer alone, We assign the colour of the pixel position in the frame buffer and assign the depth of the pixel position into the depth buffer.

# Other Hsr techniques

The only other main technique used here would have been the painting algorithm where objects are ordered from far to near and drawn in that order similar to how a painter paints. This technique is inaccurate and inefficient as the sorting is a bottleneck and there will always be situations where objects been drawn overlap each other in the Z sense causing anomalies.

The Z buffer algorithm is pixel perfect and immediate in terms of speed, has no sorting or such is required.

The only downsides of the z buffer algorithm is the allocation of memory required for these z buffer and possibly multiple versions of the z buffer As outlined below.

# Anti aliasing

Rasterization determines if a pixel should be turned on or not, But rasterization is imperfect as for example, a line is infinitely thin and the appropriate pixels are an approximation of that line.

Anti aliasing instead of determining whether a pixel should be turned on or not assigns a colour to that pixel that represents the proportion of that pixel which would be in the foreground compared to the proportion of this in the background. So A line representing the edge of say a black surface would if drawn properly, split a pixel into the black part of the surface and the other part which would be the background to that surface. Anti aliasing assigns an appropriately weighted colour to the pixel.