Tema V

Trazado de rayos (ray tracing)

Ricardo Ramos

Colaboradores: Jorge Puente Peinador, José Ramón de Diego Rodríguez, Alicia Marlasca Sedano, Mª Magdalena Vázquez García, Mª Josefa Pando Corteguera

Ya conocemos los aspectos básicos de renderizado directo, aplicando un modelo de iluminación local (ray casting). Veamos ahora las mismas ideas, pero utilizando un modelo de iluminación semiglobal (ray tracing).

1.1 Algoritmo trazador de rayos (ray tracing)

Como vimos en el Tema 1, los modelos de iluminación semiglobales, además de considerar la contribución de las fuentes de luz, también tienen en cuenta la luz reflejada y la transmitida procedente de los objetos circundantes. El algoritmo **trazador de rayos** (o **ray tracing**), que es una generalización del algoritmo de *ray casting* visto en los temas anteriores, utiliza un modelo de iluminación semiglobal.

Así, en la versión típica del algoritmo de *ray tracing*, el color (intensidad) en un punto de intersección rayo-superficie cualquiera viene determinado por tres tipos de aportaciones lumínicas:

* Por un lado está la *contribución* o **color local**, que se debe a la iluminación directa de las fuentes, y la luz ambiental. En definitiva, se trata de aplicar el modelo de iluminación local utilizado *ray casting*.

* Por otro lado está la *contribución* o **color reflejado**, que consiste en la luz que llega al punto de intersección desde los objetos circundantes, siguiendo la trayectoria de reflexión de la luz.

* Por último, el algoritmo de ray tracing también cuenta con la aportación del **color transmitido**, que se debe la luz que llega al punto de intersección después de cruzar (traspasar) los objetos vecinos (si es que son transmisores de la luz), siguiendo la trayectoria de transmisión.

En la Figura 1 se pueden apreciar las tres contribuciones en un punto de intersección rayo-superficie dado.

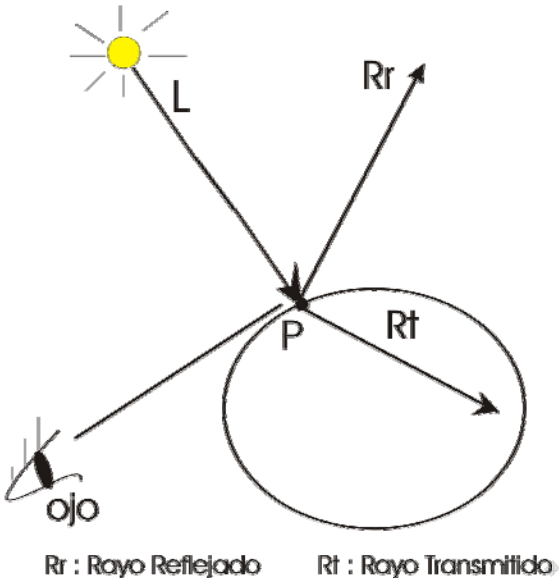

Figura 1: modelo de iluminación semiglobal utilizado con el algoritmo de ray tracing

Hasta el momento sólo hemos visto cómo calcular la intensidad en un punto de intersección rayo-superficie cualquiera. Veamos ahora cómo se pueden aplicar estas ideas, de modo que permitan calcular las intensidades en todas las superficies de los objetos del escenario visibles por el observador.

1.1.1 Naturaleza recursiva del algoritmo de ray tracing

Según se ha comentado arriba, *ray tracing* no es más que una expansión o generalización del algoritmo de ray casting. Como sabemos, en éste solamente se traza un tipo de rayos, conocidos como *rayos primarios*, que van desde el observador hasta los objetos en el escenario, a través de cada píxel del plano visual. En ray tracing, además de los rayos primarios, en cada punto de intersección rayo-superficie son trazados otros dos rayos, el **rayo reflejado** y el **rayo transmitido** (*NOTA*: el trazado de estos rayos dependerá de que el objeto intersecado sea reflectante y/o transmisor de la luz).

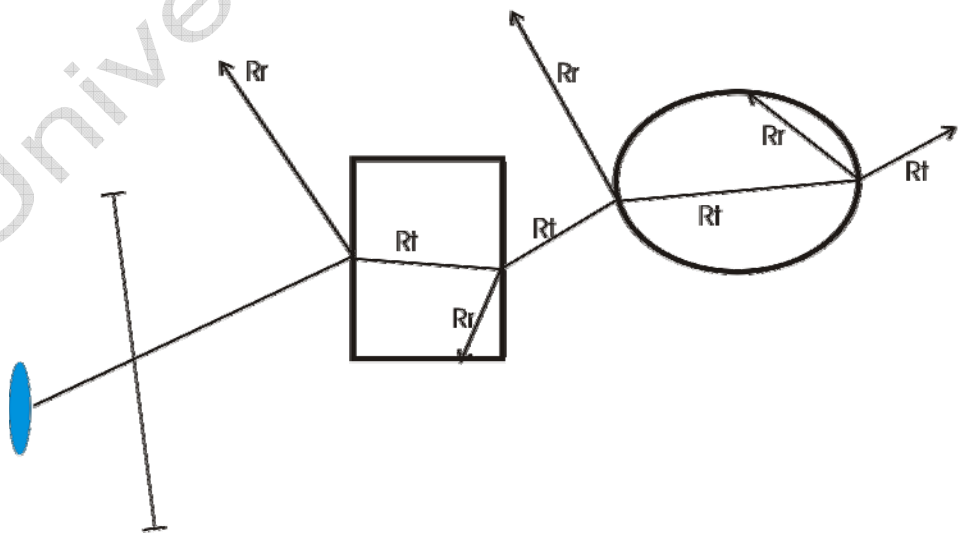

Figura 2: rayos reflejados y transmitidos en ray tracing

En la figura anterior puede apreciarse el trazado del rayo reflejado y

transmitido, en cada punto de intersección.

En la misma figura puede verse que cada rayo primario trazado lleva asociado un árbol binario (**árbol de rayos**) como el mostrado a continuación:

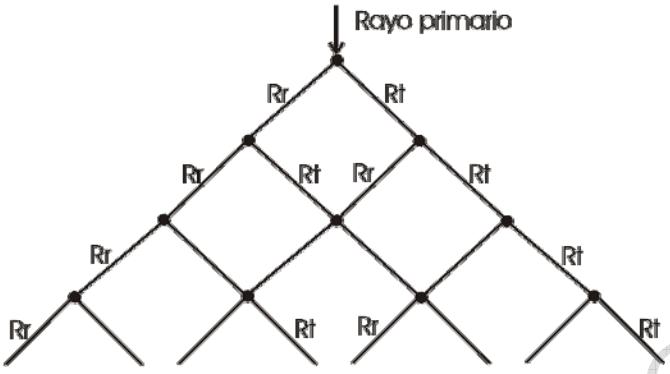

Figura 3: árbol binario asociado a cada rayo primario

A la vista del árbol anterior es fácil comprender la naturaleza recursiva del algoritmo de ray tracing. Por la descripción gráfica que acabamos de ver del algoritmo, los pasos a dar para su desarrollo son:

1) Como en *ray casting,* primero se ha de trazar el rayo primario, es decir, hay que calcular el rayo procedente del ojo que pasa a través de un píxel dado, buscando la intersección más cercana con los objetos del escenario.

2) Una vez encontrado el punto de intersección, para averiguar el **color global** (final) del rayo primario (y del píxel), se calcula primero la *contribución local* en el punto de intersección. Para ello es preciso conocer, entre otros datos, qué fuentes aportan luz, y cuales no. Esto se consigue trazando **rayos de sombra** desde el punto de intersección hacia cada una de las fuentes de luz, evaluando la contribución de cada fuente en función de sus características y de los objetos interpuestos (si los hay) en la trayectoria del correspondiente rayo de sombra (Figura 4).

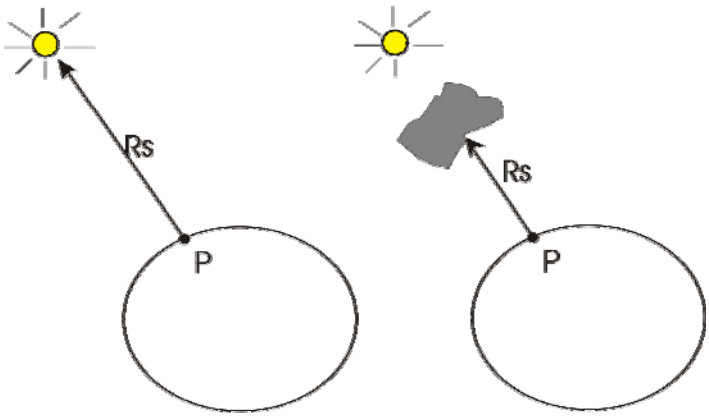

Figura 4: rayos de sombra para evaluar la *contribución local* en un píxel

3) En el caso de que la superficie presente reflexión (que es lo más frecuente) calcularemos la trayectoria del *rayo reflejado* con respecto a la normal a la superficie en el punto de intersección.

En el cálculo de esta trayectoria normalmente se supone que el objeto (superficie) es un *reflector perfecto*.

4) De la misma forma, si el objeto es transmisor de la luz, se ha de calcular la trayectoria del *rayo transmitido* hacia el interior del objeto, determinando el *ángulo de refracción* por la *ley de Snell*.

Ambos rayos pueden verse en la Figura 5.

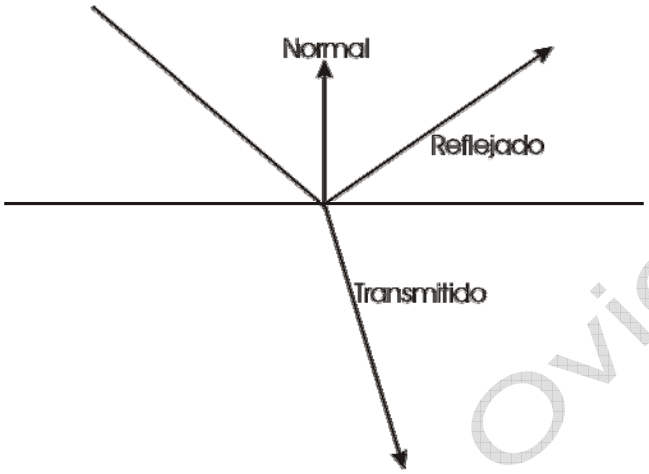

Figura 5: trazado de los rayos reflejado y transmitido

Puesto que en el algoritmo de ray tracing los rayos normalmente se trazan hacia atrás, a continuación se ha de seguir la pista del rayo reflejado (o bien la del transmitido), para encontrar los respectivos puntos de procedencia de la luz, es decir, los puntos de intersección (más cercanos) con los objetos del escenario. Localizados éstos, de nuevo se generan rayos de sombra (para calcular la contribución local), de transmisión (para hallar la contribución transmitida) y de reflexión (para la contribución reflejada), todos ellos con origen en los puntos de intersección recién encontrados. Como vemos, algorítmicamente estamos en una situación similar a la del punto de partida (aunque no igual), lo que aconseja una implantación recursiva del proceso de trazado de los rayos.

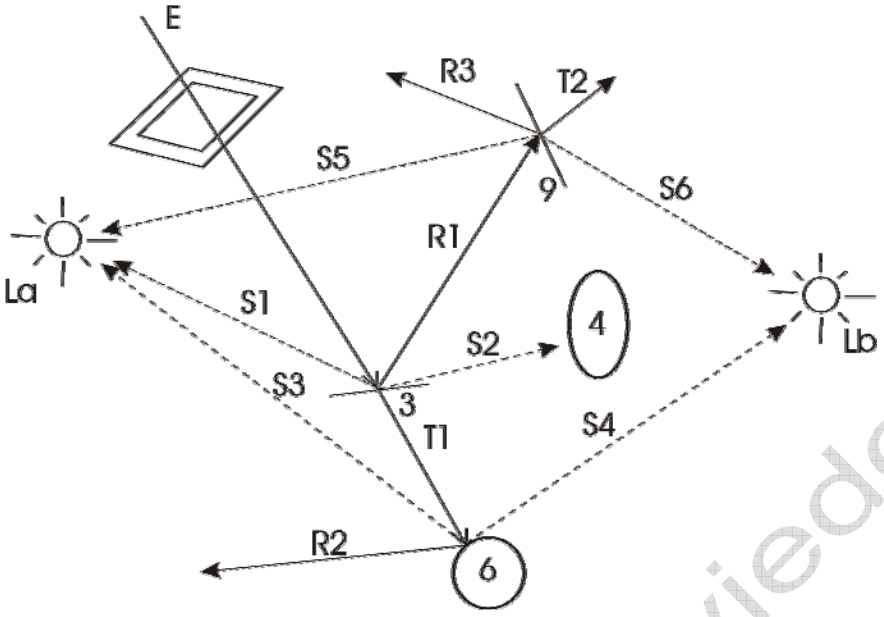

Figura 6: proceso de ray tracing

Veamos este proceso a través del ejemplo de la Figura 6. Supongamos un escenario con planos y esferas, todos con un cierto grado de transparencia y reflexión. Siendo E un rayo primario que interseca con la superficie 3, primero calcularemos la luz procedente de las fuentes generando los rayos de sombra S1 y S2; como se ve en la figura, el rayo S2 está bloqueado por el esferoide 4, que al ser opaca anula el efecto de la fuente Lb. Además, como la superficie 3 es conductora de la luz, se ha de trazar el rayo transmitido T1. Puesto que la superficie es al mismo tiempo reflectora, también ha de trazarse el rayo reflejado (R1). De modo similar se irían generando y evaluando los rayos Si, Rt y Ri, para poder determinar finalmente el color del rayo E.

La Figura 7 muestra el árbol de rayos correspondiente al ejemplo anterior. La raíz del árbol representa el rayo primario procedente del ojo, y cada uno de los nodos es una intersección con un objeto. De cada nodo salen por un lado los rayos de sombra, y por otro los rayos reflejados y transmitidos, en el caso de que los objetos sean reflexores y transmisores, respectivamente.

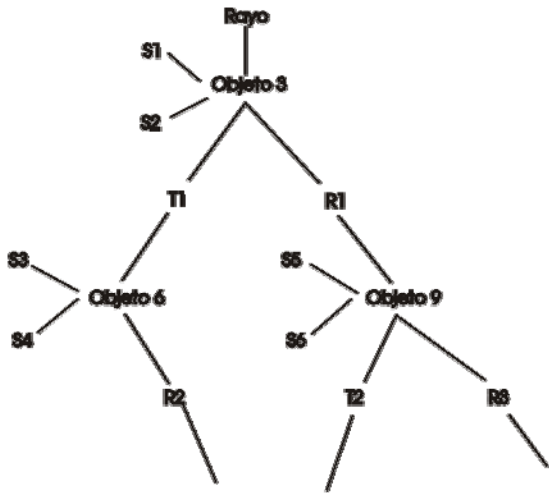

Figura 7: árbol de rayos del ejemplo

1.1.2 Contribución de los diferentes rayos al color global

A la hora de calcular el color global deberemos tener en cuenta que no todos los rayos generados influyen en la misma proporción. Por ejemplo, la luz que llega directamente de una fuente, normalmente es bastante más intensa que la que llega a través de una trayectoria de reflexión, tras efectuar varias intersecciones. Cuanto más abajo sea el nivel de un rayo (en el árbol de rayos), menor será su influencia a la hora de determinar el color final de un píxel.

En el momento de trazar, una cuestión que se plantea es cuándo se ha de dar por finalizado el proceso de generación del árbol de rayos. Por lo común, a parte de las limitaciones de memoria y de potencia de cálculo que imponga el sistema donde se realice el trazado, el avance de un rayo se dará por finalizado cuando no interseque con ningún objeto (y salga del escenario), o bien cuando su contribución al color final sea nula, situación que normalmente quedará establecida por el nivel en el árbol de los rayos trazados. Así, todo rayo cuyo nivel dentro del árbol de rayos sea inferior a cierto nivel establecido será desestimado, ya que su contribución al color global se presupone ínfima. Por ejemplo, en el caso anterior, si fijamos en 1 la profundidad límite, el árbol de rayos resultante sería el de la Figura 8; en él vemos que sólo se consideran hasta tres intersecciones para cada rayo generado.

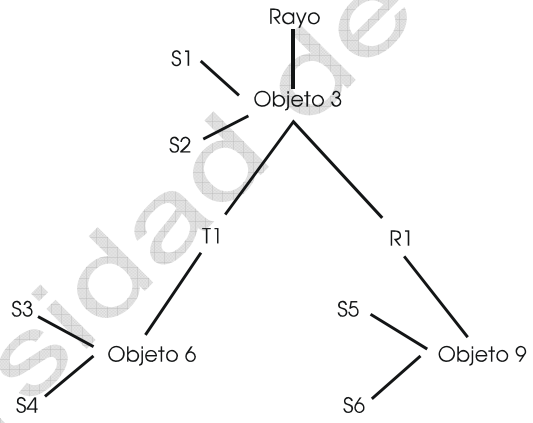

Figura 8: árbol de rayos con profundidad 1

La evaluación del árbol de rayos se realiza desde los nodos hoja hacia la raíz, acumulando las sucesivas aportaciones parciales, a modo de "afluentes de luz". Por tanto, la intensidad de un determinado nodo se calcula a partir de la intensidad de sus nodos hijos.

El pseudocódigo de una versión pedagógica (no optimizada, ni completa) del algoritmo trazador de rayos (TR) sería:

```
color TR(VECTOR inicio_rayo, VECTOR dir_rayo, int profundidad)
{
   #define PROFUNDIDAD_MAX 4
   #define COLOR_FONDO AZUL

   VECTOR pto_interseccion, dir_reflexion, dir_transmision;
   COLOR color_local, color_reflexion, color_transmision;

   //Se verifica si el rayo ha alcanzado la profundidad máx.
   if(profundidad > PROFUNDIDAD_MAX)
      return(negro);
   else
   {
```

```
        //Se busca el punto de intersección rayo-superficie
        pto_interseccion = hallar_interseccion(inicio_rayo, dir_rayo);

        //Si no hay intersección se devuelve el color de fondo
        if(pto_interseccion == 0)
           return(COLOR_FONDO)
        else
        {
           /* Se calcula la contribución local en el punto de intersec-
           ción.
           NOTA: Los argumentos requeridos no se especifican */
           color_local = contribución_local();

           // ***Se calcula la contribución reflejada***

           /* Primero se averigua la dirección en el espacio del rayo
           reflejado. No se especifican los argumentos */
           dir_reflexion = direccion_reflexion();

           /* A continuación se llama recursivamente al proceso TR, para
           calcular la contribución del rayo reflejado */
           color_reflexion = TR(pto_interseccion, dir_reflexion, profun-
           didad + 1);
           // ***Se calcula la contribución transmitida***

           /* Primero se averigua la dirección en el espacio del rayo
           transmitido. No se especifican los argumentos */
           dir_transmision = direccion_transmision();

           /* A continuación se llama recursivamente al proceso TR, para
           calcular la contribución del rayo transmitido */
           color_transmision = TR(pto_interseccion, dir_transmision,
           profundidad + 1);

           /* Finalmente se calcula el color global, resultante de la
           suma de las tres contribuciones anteriores. A cada contribu-
           ción se le asocia un peso, para así poder controlar el resul-
           tado final */
           return(combinar_colores(color_local, peso_cl, co-
           lor_reflexion, peso_cr, color_transmision, peso_ct));
        }//Final, si hay intersección
     }//Final, si no se sobrepasa la profundidad máxima.
}//Final del proceso TR.
```

Veamos otra versión del algoritmo de ray tracing, un poco más elaborada, presentada por *T. Whitted* en el año 1980.

```
procedure TR_recursivo ;
{Genera el color de cada uno de los pixels de la pantalla.}
begin
   seleccionar el centro de proyección y el plano de la imagen;
   for cada línea horizontal de la imagen do
     for cada pixel de la línea do
       begin
        determinar el rayo desde el centro de proyección hacia el pixel
        pixel := TR_trazado (rayo, 1);
       end;
end;

procedure TR_trazado (rayo: TR_rayo; prof_actual: integer): RT_color;
{Intersecciona el rayo con los objetos de la escena y obtiene la tonalidad de la intersección más cercana.}
{rayo indica la dirección del rayo y prof_actual representa la profundidad actual dentro del árbol de rayos.}
begin
   determinar la intersección mas cercana del rayo con un objeto;
```

```
    if interseca con un objeto then
      begin
        calcular la dirección del vector normal en el punto de intersección
        TR_trazado := TR_tonalidad (objeto interseccionado, rayo, punto de intersección, normal, prof_actual);
      end
    else
    {Si no intersecta con ningún objeto le damos la luz de fondo de la escena.}
      TR_trazado := COLOR_DE_FONDO;
end;

procedure TR_tonalidad (
                    objeto : TR_objeto;    {Objeto intersecado}
                    rayo : TR_rayo;        {Rayo incidente}
                    punto:TR_punto;    {Punto de intersección a evaluar}
                    normal : TR_normal;    {Normal a ese punto}
                    prof_actual : integer; {Profundidad en el árbol de rayos}
                   ) : TR_color;
{Obtiene la tonalidad en el punto de intersección trazando los rayos de sombra, reflexión y refracción.}
var
   color : TR_color;                    {Color del rayo}
   RayoR,RayoT, RayoS : TR_rayo;            {Rayos reflejado, transmitido y de sombra}
   ColorR,ColorT :TR_color;                {Color del rayo reflejado y transmitido}
begin
    {color se inicializa con el color debido a la luz ambiental}
    color := COLOR_AMBIENTAL;
    if prof_actual<ProfundidadLimite then
      begin
        for cada fuente de luz do
          begin
            RayoS := rayo desde el punto hacia la fuente de luz;
            {Se comprueba que la luz no provenga del lado opuesto del objeto}
             if la dirección de la luz es positiva con respecto a la normal then
              calcular cuanta luz es bloqueada por las superficies opacas y transparentes, y
              utilizar los términos de luz difusa y especular antes de actualizar color;
          end;
        if el objeto es reflectante then
          begin
            RayoR:=dirección del rayo reflejado en el punto de intersección;
            ColorR := TR_trazado (RayoR, prof_actual+1);
            escalar ColorR mediante el coeficiente especular de reflexión y añadirlo a color;
          end;

        if el objeto es transparente then
          begin
            RayoT := dirección del rayo transmitido en el punto de intersección;
            if no hay reflexión total then
              begin
                RayoT := TR_trazado (RayoT, prof_actual+1);
                escalar ColorT mediante el coeficiente de refracción y añadirlo a color.

              end
          end
      end
    {Si se alcanzó la profundidad límite la contribución de la luz reflejada y refractada es nula.}
    TR_tonalidad := color; {Devuelve el color del rayo.}
end;
```

La función *TR_recursivo* es la encargada de calcular el color de cada píxel de la pantalla. Para ello se apoya en el proceso *TR_trazado* que calcula el color de un determinado rayo, hallando primero el punto de intersección rayo-superficie más cercano; si el rayo no interseca con ningún objeto, simplemente devuelve el color de fondo de la imagen. La función *TR_intensidad* recibe información referente al punto de intersección, y calcula qué luz llega a ese punto (*contribución local*). Comienza calculando la aportación de la luz ambiental, y de cada fuente de luz activa, teniendo en cuenta las características ópticas de los objetos intersecados por los rayos de sombra correspondientes; así, un objeto opaco anulará total o parcialmente el efecto de la fuente, mientras que un objeto translúcido atenuará en una determinada medida dicho efecto.

En el caso de no haber alcanzado la profundidad límite se obtendría también la contribución debida a la luz transmitida y reflejada (si es que el objeto es translúcido y reflexivo, respectivamente), mediante una llamada recursiva a *TR_trazado* con cada rayo, especificando que se desciende un nivel más en el árbol de rayos. Otra información que podría incluirse es el índice de refracción del medio por el que se está propagando el rayo, permitiendo así calcular más fácilmente los ángulos de refracción.

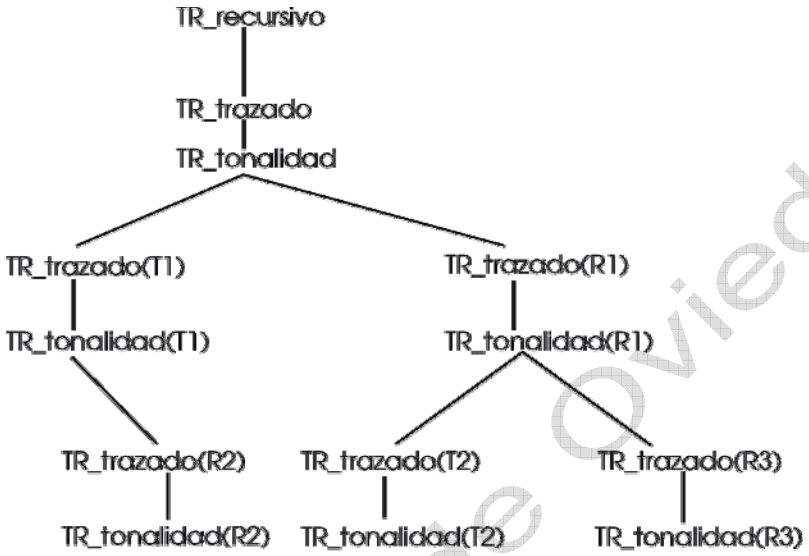

Figura 9: árbol de llamadas

Dada la naturaleza recursiva del proceso, se iría generando un *árbol de llamadas* al proceso TR, equiparable al *árbol de rayos*. En el ejemplo de arriba, el árbol de llamadas que se generaría puede verse en la Figura 9.

Hemos visto cómo la recursividad permite implementar de manera sencilla el algoritmo TR, generando imágenes con efectos de sombras, transparencias y/o reflejos especulares. Veamos ahora con más detalle los principales subprocesos en los que se basa el algoritmo de ray tracing.

1.2 Modelo de iluminación semiglobal

Como acabamos de ver, a la hora de calcular la intensidad de un píxel con *ray tracing* se han de tener en cuenta tres tipos de contribuciones: la *local*, la *reflejada* y la *transmitida*. Veamos los detalles sobre el cálculo de las intensidades de estas tres contribuciones al color final de los píxeles.

1.2.1 Cálculo de la contribución local

El cálculo de la aportación local en ray tracing no difiere mucho de su homólogo en ray casting. En la contribución local normalmente se utiliza el modelo de intensidad de Phong, ampliado con la componente (local) transmitida. No se han de confundir las componentes especular y transmitida del modelo de intensidad de Phong (contribución local), con las contribuciones reflejada y transmitida del modelo de iluminación semiglobal ya que, entre otras razones, el cálculo de los respectivos vectores de dirección varía ligeramente, como pronto veremos.

Recordando el modelo de Phong, el cálculo del vector director de la

componente especular venía dado por el vector unitario $\mathbf{R} = \mathbf{L} + 2\mathbf{N}\cos\varphi$, o bien, $\mathbf{R} = 2\mathbf{N}(\mathbf{N}\cdot\mathbf{L}) - \mathbf{L}$

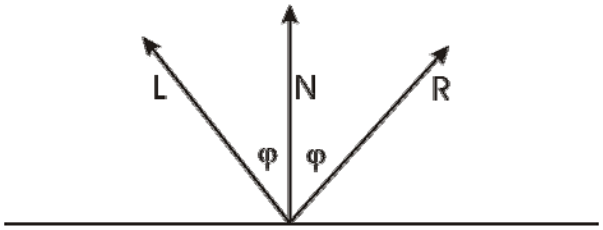

Figura 10: reflexión especular (local)

siendo

L: vector unitario que representa la dirección del rayo incidente.

N: vector unitario que representa la normal a la superficie.

Los vectores **L**, **N** y **R** son coplanarios, es decir, uno puede expresarse como combinación lineal de los otros dos (Figura 10).

Por otro lado, un rayo que llega a una superficie parcial o totalmente transmisora es refractado (cambia su dirección), debido a la variación de la velocidad de la luz en diferentes medios. Los ángulos de incidencia y refracción están relacionados por la ley de *Snell*, la cual afirma que

$$\frac{sen\phi_1}{sen\phi_2} = \frac{\eta_2}{\eta_1}$$

donde η_1 y η_2 son los índices de refracción de los medios lumínicos 1 y 2 (Figura 11).

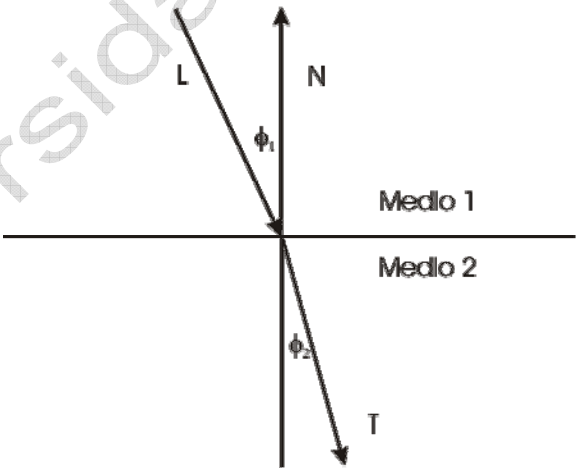

Figura 11: transmisión de la luz (local)

El cálculo del vector unitario de transmisión (**T**) viene dado por la fórmula

$$\mathbf{T} = \frac{1}{\eta}\mathbf{L} - \left(\cos\phi_2 - \frac{1}{\eta}\cos\phi_1\right)\cdot\mathbf{N}$$

donde η es el índice de refracción del medio 2, con respecto al medio 1, o sea,

$\eta = \frac{\eta_2}{\eta_1}$, y $\cos\phi_2 = \sqrt{1 - \frac{1}{\eta^2}\left(1 - \cos^2\phi_1\right)}$.

La incorporación de la componente transmitida en el modelo de Phong es la mayor novedad, respecto lo visto al estudiar ray casting. Dentro de la ecuación de Phong (contribución local), el cálculo de las componentes transmitida y especular es similar.

En efecto, en el modelo de Phong la intensidad debida a la componente transmitida viene dada por la fórmula

$$I_t = K_t(\mathbf{N}\cdot\mathbf{H}')^n$$

donde K_t es el **coeficiente de transmisión** (o de *refracción*), y **N** la normal a la superficie.

Al igual que **H** en la componente especular, **H'** es el *vector bisector* del ángulo formado por los vectores **L** y **V**, siendo este último el *vector unitario dirigido hacia el observador* (Figura 12). El cálculo de **H'** viene dado por la expresión

$$\mathbf{H}' = \frac{-\mathbf{L} - \eta\mathbf{V}}{\eta - 1}, \text{ o bien, } \mathbf{H}' = \frac{\mathbf{V} - \eta\mathbf{L}}{\eta - 1}$$

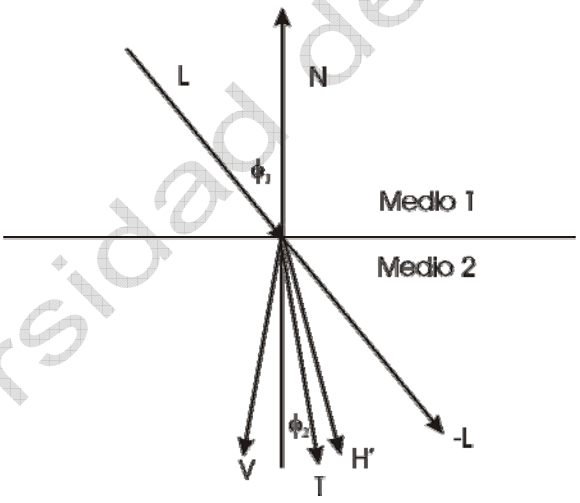

Figura 12: H' es el vector bisector del ángulo formado por los vectores V y -L

Incorporando la componente transmitida en la ecuación de Phong, se tiene que la *contribución local* en un punto dado de la superficie viene dada por

$$I_{local} = I_a K_a + I_i \left[K_d(\mathbf{NL}) + K_e(\mathbf{NH})^{n_e} + K_t(\mathbf{NH}')^{n_t}\right]$$

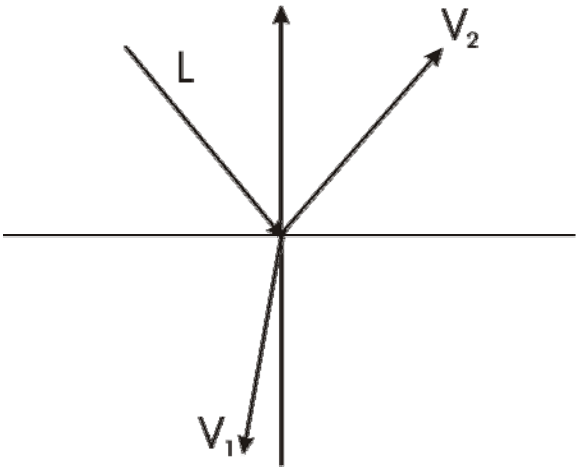

Figura 13: el observador no puede estar en dos lugares diferentes

Ver que para una determinada fuente de luz, *sólo uno de los dos últimos términos* de la expresión anterior puede estar activo, ya que el observador o está en V_1, o bien está en V_2 (Figura 13). Ver también que aparece el exponente n_t en la componente transmitida (la *t* es para diferenciarlo del "*n*" especular (n_e)). Esto significa que, dependiendo del valor que tome n_t, *la superficie será más o menos transmisora*, de igual modo que n_e, según vimos, establece el grado de reflexividad.

1.2.2 Cálculo del vector director de los rayos reflejado y transmitido

El vector unitario del rayo reflejado (el utilizado para calcular la *contribución reflejada* en el modelo de iluminación semiglobal) viene dado por la expresión

$$\mathbf{R} = \mathbf{L} - 2\mathbf{N}(\mathbf{L} \cdot \mathbf{N})$$

siendo **L** el vector director normalizado del rayo incidente. Notar que esta fórmula varía ligeramente con respecto a la utilizada al calcular la contribución local (**R** = 2**N**(**N** ·**L**) – **L)** debido a que cambia el signo del vector incidente. En cambio, las ecuaciones utilizadas para calcular la dirección del rayo transmitido son las mismas que las utilizadas en el modelo local.

Ver que en el cálculo de las direcciones de ambos rayos, los objetos se consideran reflexores y transmisores perfectos, a diferencia de lo que ocurre en el cálculo de la contribución local, donde las mismas superficies son más o menos difusoras y/o transmisoras, dependiendo de los valores de n_e y n_t. Por este motivo suele decirse que el modelo de iluminación semiglobal utilizado en ray tracing es *híbrido.*

1.2.3 Combinación del color: intensidad global

Según quedó indicado en el algoritmo RT, la intensidad (color) final de un píxel se calcula sumando la *intensidad local* (I_{local}), la *transmitida* ($I_{transmitida}$) y la *reflejada* ($I_{reflejada}$). Para poder ejercer un mayor control sobre el modelo de iluminación semiglobal, cada una de las contribuciones anteriores tiene un valor de ponderación (peso) asociado, de modo que

$$I_{final} = I_{local}W_l + I_{reflejada}W_r + I_{transmitida}W_t$$

1.2.4 Limitaciones del modelo de iluminación semiglobal

Cuando se trabaja con reflexiones difusas y especulares en un modelo de iluminación, de los cuatro efectos lumínicos posibles (en la Figura 14, a: difuso → difuso, b: especular → difuso, c: difuso → especular, d: especular → especular), el modelo semiglobal empleado con ray tracing sólo utiliza el caso "d", y no completamente, pues la luz es reflejada sólo en la dirección de **R**, por lo que no es difuminada dentro del ámbito establecido.

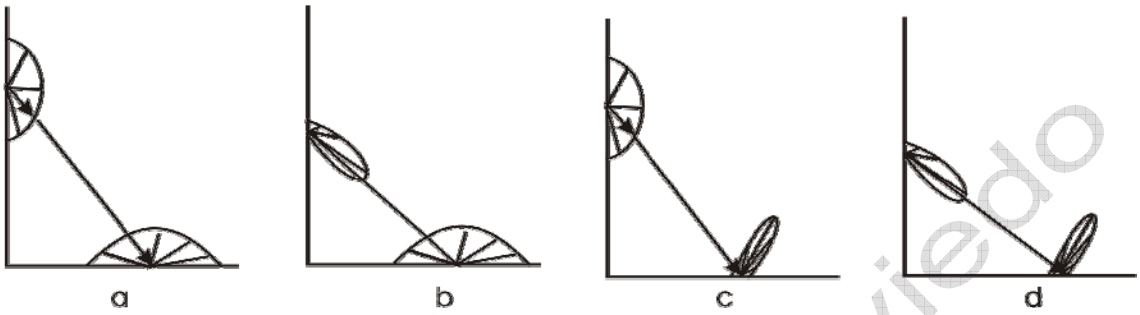

Figura 14: efectos lumínicos posibles

Por otro lado, la situación "b" también se realiza parcialmente, ya que la difusión de la luz directa que llega de las fuentes, queda modelada al calcular la I_{local}.

1.2.5 Sombreado en ray tracing

No hay muchas novedades en cuanto al sombreado, con respecto a lo visto al estudiar el algoritmo de *ray casting*. Las sombras se incorporan de modo sencillo (aunque normalmente caro, computacionalmente hablando) en el modelo de iluminación semiglobal utilizado en ray tracing.

Según vimos, en cada punto de intersección rayo-superficie se calcula I_{local} suponiendo que $\mathbf{L}_i$ (el vector hacia la fuente i) no es interrumpido por ningún objeto en su camino. Sin embargo, si $\mathbf{L}_i$ estuviese bloqueado por algún obstáculo, el punto de intersección se encontraría en sombra con respecto a la fuente de luz i.

Para averiguar si $\mathbf{L}_i$ está obstruido o no, al igual que en ray casting se ha de trazar un *rayo de sombra* desde el punto de intersección hacia la fuente i. Si un objeto opaco se interpone en su camino, la I_{local} normalmente se reduce a la componente ambiente; si el objeto fuese semitransparente, se calcularía una atenuación de la I_{local}.

Para finalizar, ver que si el número de fuentes es elevado, el tiempo de cálculo requerido para el sombrado crece exponencialmente, pudiendo llegar a superar al tiempo requerido para trazar el *árbol de rayos*. Esto es debido a que, si hay n fuentes de luz, *en cada punto de intersección se han de trazar n rayos de sombra*, uno por cada fuente. Existen métodos para acelerar el sombreado en RT, que se verán en el próximo tema.

1.3 Ejemplo numérico de trazado de rayos

Supongamos que en un escenario hay un plano, una esfera, una fuente puntual, un observador, luz ambiente y luz de fondo. Los parámetros geométri-

cos y ópticos de estos elementos son:

PLANO: **Parámetros geométricos**: [A, B, C, D] = [1, 0, 0, –4] **Parámetros ópticos**: No transmisor $Ka_{(R, G, B)}$ = (0.2, 0.3, 0.1) $Kd_{(R, G, B)}$ = (0.7, 0.5, 0.4) $Ke_{(R, G, B)}$ = (0.8, 0.6, 0.9) n = 255	ESFERA: **Parámetros geométricos**: C_0 (X, Y, Z) = (17, 9, 8) Radio = 3 **Parámetros ópticos**: $Ka_{(R, G, B)}$ = (0.3, 0.2, 0.4) $Kd_{(R, G, B)}$ = (0.5, 0.7, 0.9) $Ke_{(R, G, B)}$ = (0.7, 0.4, 0.2) n = 2	FUENTE: **Parámetros geométricos**: I_f(X, Y, Z) = (9, 12, 16) **Color de la luz**: (R,G,B) = (220, 200, 255)
LUZ AMBIENTE: **Color de la luz**: (R,G,B) = (180, 160, 190)	LUZ DE FONDO: **Color de la luz**: (R,G,B) = (200, 200, 255)	OBSERVADOR: **Posición**: P_0(X, Y, Z) = (16, 2, 7) **Rayo director**: Rd_0(X, Y, Z) = (–12, 4, 1)

Aplicando ray tracing, con un nivel de profundidad máximo de 1 (o sea, el árbol de trazado tendrá 2 niveles, el raíz (nivel 0) y el nivel 1), se ha de calcular la intensidad de **luz roja** que llega al observador (P_0), según el modelo de intensidad de Phong.

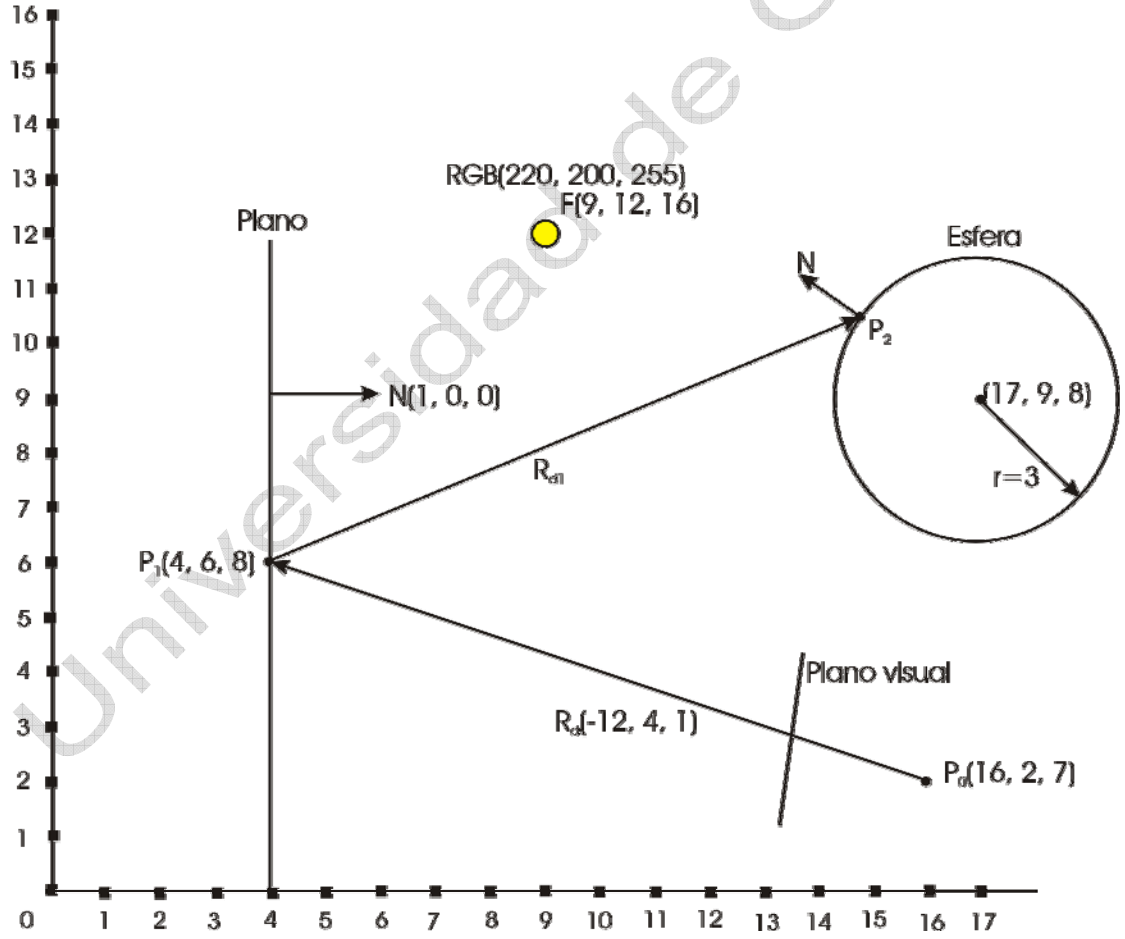

Figura 15: condiciones iniciales del ejemplo numérico

Condiciones para simplificar el cálculo manual:

* El rayo primario sólo interseca con el plano (Figura 15), luego no es necesario probar si el rayo primario interseca con la esfera (el algoritmo de

TR debería hacerlo).

* Tampoco vamos a trazar rayos de sombra (el algoritmo también debería trazarlos).

NOTAS:

* En la intersección rayo-esfera, utilizaremos la versión geométrica.

* En el cálculo de la intensidad de la componente local utilizaremos el vector **H**. Para facilitar las cosas, no compensaremos el resultado modificando "*n*".

* Cada vez que calculemos un nuevo vector de dirección, éste será normalizado.

* Recordemos que, siendo **L** el vector unitario de un rayo incidente **en el árbol de rayos,** para encontrar el vector director del *rayo reflejado* se ha de utilizar la expresión $\mathbf{R} = \mathbf{L} - 2\mathbf{N}(\mathbf{L}\cdot\mathbf{N})$. Además, en el cálculo de la intensidad local el vector director del rayo incidente será igual a **–L**.

* Efectuaremos los cálculos con una precisión de 3 cifras, redondeado a mayor si la cuarta cifra es ≥ 5 (p.e., 0.2317254 → 0.232).

Desarrollo:

1) Se comienza normalizando el rayo primario, con origen en P_0 e intersección en P_1.

$$|R_d| = \sqrt{12^2 + 4^2 + 1} = 12{,}689$$

R_d = (–0,946, 0,315, 0, 079)

2) Se calcula el punto de intersección con el plano

$$t = \frac{-(\mathbf{N_p}\cdot\mathbf{R_0} + D)}{\mathbf{N_p}\cdot\mathbf{R_d}} = \frac{\mathbf{V_o}}{\mathbf{V_d}}$$

$\mathbf{N_p}$ = (1, 0, 0), luego, $\mathbf{V_d}$ = (1, 0, 0) $\cdot\mathbf{R_d}$ = –0,946 ·1 + 0,315 ·0 + 0, 079 ·0. Por tanto $\mathbf{V_d}$ = –0,946.

Por otro lado, $\mathbf{V_o}$ = –(1 ·16 + 0 ·2 + 0 ·7 – 4) = –(16 – 4) = –12.

En definitiva, $\boxed{t = \frac{-12}{-0{,}946} = 12{,}685}$

3) Se calculan las coordenadas de intersección rayo-plano

$$\mathbf{P_1}(x, y, z) = \mathbf{P_0} + \mathbf{R_d}\cdot t$$

x = 16 + –0,946 ·12,685 = 4

y = 2 + 0,315 ·12,685 = 6

z = 7 + 0,079 ·12,685 = 8. Luego $\mathbf{P_1}(x, y, z)$ = (4, 6, 8).

4) Se calcula $\mathbf{R_{d1}}$ (vector director del rayo reflejado en el plano). No calcularemos el rayo transmitido ya que el plano es opaco.

$$\mathbf{R_{d1}} = \mathbf{R_d} - 2\mathbf{N}(\mathbf{R_d} \cdot \mathbf{N})$$

Haremos las operaciones simbólicamente, pues en este caso resultan más sencillas, al ser $\mathbf{N} = \mathbf{N_p} = (1, 0, 0)$.

Siendo $\mathbf{R_d} = (a, b, c)$, queda que $\mathbf{R_d} \cdot \mathbf{N} = (a, b, c)(1, 0, 0) = (a \cdot 1 + b \cdot 0 + c \cdot 0) = (a, 0, 0)$. De igual modo $2\mathbf{N} = 2(1, 0, 0) = (2, 0, 0)$. Por lo tanto

$2\mathbf{N}(\mathbf{R_d} \cdot \mathbf{N}) = (2, 0, 0)(a, 0, 0) = (2a, 0, 0)$. Finalmente,

$\mathbf{R_{d1}} = \mathbf{R_d} - 2\mathbf{N}(\mathbf{R_d} \cdot \mathbf{N}) = (a, b, c) - (2a, 0, 0) = (-a, b, c)$. Como $a = -0{,}946$, $b = 0, 315$, y $c = 0, 079$, se tiene que

$$\boxed{\mathbf{R_{d1}} = (0{,}946, 0{,}315, 0, 079)}$$

5) Se averigua si el rayo interseca con la esfera, aplicando el método geométrico

Sabiendo que $\mathbf{P_1} = (4, 6, 8)$, y que $\mathbf{R_{d1}} = (0{,}946, 0{,}315, 0, 079)$

a) Se calcula la distancia entre el origen del rayo y el centro de la esfera

$\mathbf{OC} = \mathbf{S_c} - \mathbf{P_1} = (17, 9, 8) - (4, 6, 8) = (13, 3, 0)$

$L_{2oc} = (13^2 + 3^2 + 0) = 178$

b) Se calcula la distancia mínima entre el rayo y el centro de la esfera.

$\mathbf{T_A} = \mathbf{OC} \cdot \mathbf{R_{d1}} = (13, 3, 0)(0, 946, 0, 315, 0, 079) = 13 \cdot 0, 946 + 3 \cdot 0, 315 + 0 \cdot 0, 079 = 13, 243$

$$\boxed{\mathbf{T_A} = 13, 243}$$

c) Se verifica si el origen del rayo se encuentra fuera de la esfera

¿$L_{2oc} > r^2$? $178 > 9$, luego el origen del rayo se encuentra fuera de la esfera. Además, como $\mathbf{T_A} > 0$, el origen está delante de la esfera, es decir, aún puede intersecar con ella.

d) Se calcula $T_{2Ai} = r^2 - L_{2oc} + (\mathbf{T_A})^2$

$T_{2Ai} = 3^2 - 178 + (13, 243)^2 = 6, 377$

e) Como $T_{2Ai} > 0$, el rayo interseca con la esfera. Se calcula $t_i = \mathbf{T_A} - \sqrt{T_{2Ai}}$

$t_i = 13{,}243 - \sqrt{6{,}377} = 10{,}718$

f) Se calculan las coordenadas del punto de intersección ($\mathbf{P_2}(x_s, y_s, z_s) = \mathbf{P_1} + \mathbf{R_{d1}} \cdot t_i$).

$x_s = 4 + 0{,}946 \cdot 10{,}718 = 14{,}139$

$y_s = 6 + 0{,}315 \cdot 10{,}718 = 9{,}376$

$z_s = 8 + 0{,}079 \cdot 10{,}718 = 8{,}847$, luego

$$\boxed{\mathbf{P_2} = (14{,}139,\ 9{,}376,\ 8{,}847)}$$

g) Se calcula la normal en $\mathbf{P_2}$ $\left(\mathbf{N}_2 = \left[\frac{x_s - x_c}{r}, \frac{y_s - y_c}{r}, \frac{z_s - z_c}{r}\right]\right)$

$$\mathbf{N}_2 = \left[\frac{14{,}139 - 17}{3}, \frac{9{,}376 - 9}{3}, \frac{8{,}847 - 8}{3}\right] = [-0{,}954,\ 0{,}125,\ 0{,}282].$$

En definitiva,

$$\boxed{\mathbf{N_2} = [-0{,}954,\ 0{,}125,\ 0{,}282]}$$

6) Comienza el cálculo de las intensidades locales (contribución local)

Primero en $\mathbf{P_2}$ (intersección con la esfera)

* Vector incidente $\mathbf{L_2} = \mathbf{F} - \mathbf{P_2} = (9,\ 12,\ 16) - (14{,}139,\ 9{,}376,\ 8{,}847)$. Por tanto, $\mathbf{L_2} = (-5{,}139,\ 2{,}624,\ 7{,}153)$.

* Se normaliza $\mathbf{L_2}$

$|L_2| = \sqrt{5{,}139^2 + 2{,}624^2 + 7{,}153^2} = 9{,}190$, de donde se deduce que el vector normalizado será:

$$\boxed{\mathbf{L_2} = (-0{,}559,\ 0{,}286,\ 0{,}778)}$$

***** A continuación calculamos el vector **H** (**NOTA**: En el nivel 1 el observador se encuentra en $\mathbf{P_1}$. Ver la Figura 15)

$\mathbf{V_1} = -\mathbf{R_{d1}} = -(0{,}946,\ 0{,}315,\ 0,\ 079) = (-0{,}946,\ -0{,}315,\ -0,\ 079)$.

En este nivel, al vector **H** lo llamaremos $\mathbf{H_2}$, para distinguirlo de los vectores **H** en otros niveles del árbol de rayos. $\mathbf{H_2}$ viene dado por

$$\mathbf{H}_2 = \frac{\mathbf{L}_2 + \mathbf{V}_1}{|\mathbf{L}_2 + \mathbf{V}_1|}$$

$\mathbf{L_2} + \mathbf{V_1} = (-0{,}559,\ 0{,}286,\ 0{,}778) + (-0{,}946,\ -0{,}315,\ -0,\ 079) = (-1{,}505,\ -0{,}029,\ 0{,}699)$. $|\ \mathbf{L_2} + \mathbf{V_1}\ | = 1{,}66$. Por lo tanto

$$\boxed{\mathbf{H_2} = (-0{,}907,\ -0{,}017,\ 0{,}421)}$$

7) Se calcula la intensidad local (roja) en $\mathbf{P_2}$

$I_{local} = (IK)_a + I_i[K_d(\mathbf{N_2L_2}) + K_e(\mathbf{N_2H_2})^n$

$\mathbf{N_2L_2} = (-0{,}954,\ 0{,}125,\ 0{,}282)\ (-0{,}559,\ 0{,}286,\ 0{,}778) = 0{,}533 + 0{,}036 + 0{,}219 = 0{,}788$. Por otro lado,

$\mathbf{N_2H_2} = (-0{,}954) \cdot (-0{,}907) + 0{,}125 \cdot (-0{,}017) + 0{,}282 \cdot 0{,}421 = 0{,}865 - 0{,}002 + 0{,}119 = 0{,}982$. Queda por tanto que

$I_{local\text{-}2(rojo)} = 180 \cdot 0{,}3 + 220 \cdot 0{,}5 \cdot 0{,}788 + 220 \cdot 0{,}7 \cdot (0{,}982)^2 = 289{,}19$

Nota: Como el árbol de rayos solamente tiene 2 niveles, no puede haber contribuciones reflejada y transmitida en el punto de intersección de

la esfera ($\mathbf{P_2}$). Por tanto, la I_{local} en $\mathbf{P_2}$ pasa a ser la intensidad (roja) reflejada en $\mathbf{P_1}$.

8) Se calcula la intensidad local (roja) en $\mathbf{P_1}$

$\mathbf{V_0} = -\mathbf{R_d} \Rightarrow \mathbf{V_0} = (0{,}946, -0{,}315, -0{,}079)$.

El vector incidente en $\mathbf{P_1}$ será

$\mathbf{L_1} = \mathbf{F} - \mathbf{P_1} = (9, 12, 16) - (4, 6, 8) = (5, 6, 8)$. Normalizado queda:

$|\mathbf{L_1}| = \sqrt{5^2 + 6^2 + 8^2} = 11{,}180$, de donde se deduce que

$$\boxed{\mathbf{L_1} = (0{,}447,\ 0,\ 537,\ 0{,}716)}$$

El vector $\mathbf{H}$ en este nivel ($\mathbf{H_1}$) viene dado por

$\mathbf{L_1} + \mathbf{V_o} = (1{,}393,\ 0{,}222,\ 0{,}637)$

$|\mathbf{L_1} + \mathbf{V_o}| = 1{,}548$. Por tanto,

$$\boxed{\mathbf{H_1} = (0{,}9,\ 0{,}143,\ 0{,}412)}$$

$\mathbf{N_1L_1} = (1, 0, 0)\ (0{,}447, 0, 537, 0{,}716) = 0{,}447$

$\mathbf{N_1H_1} = (1, 0, 0)\ (0{,}9, 0{,}143, 0{,}412) = 0{,}9$. Se tiene entonces que,

$I_{local\text{-}1(rojo)} = 180 \cdot 0{,}2 + 220 \cdot 0{,}7 \cdot 0{,}447 + 220 \cdot 0{,}8 \cdot (0{,}9)^{255} = 104{,}838$.

En conclusión, teniendo presente que $I_{reflejada\text{-}1} = I_{local\text{-}2(rojo)}$, la intensidad global (correspondiente al rojo) que llega a $\mathbf{P_1}$ será

$I_{globa\text{-}1l(\text{rojo})} = I_{local\text{-}1(rojo)} + I_{reflejada\text{-}1} = 104{,}838 + 289{,}19 = 394{,}028$

Finalmente, como la intensidad que llega $\mathbf{P_0}$ es la que se refleja en $\mathbf{P_1}$, se tiene que

$$\boxed{I_{global(\text{rojo})} = 394{,}028}$$

1.4 Aliasing y antialiasing

Al traspasar a un espacio discreto (*discretizar*) un modelo definido en un espacio continuo n-dimensional, inevitablemente *se pierde información* sobre el modelo, debido a la diferencia de puntos utilizados en la definición de los modelos en ambos espacios. Cuanta más información visual del modelo se pueda transferir hacia el espacio discreto, mayor será la calidad de la imagen obtenida.

Por otro lado, cuando se discretiza el movimiento continuo de los objetos, también se produce pérdida de información, ya que no es posible registrar todas las "instantáneas" (frames) que constituyen el movimiento continuo.

A la pérdida de información espacial se conoce como **aliasing espacial**, y a las discontinuidades del movimiento (pérdida de información) que pueden aparecer en las animaciones de los objetos, se asocian (o se deben) al **aliasing temporal**.

1.4.1 Aliasing espacial

Cuando se traza un rayo primario a través de un píxel hacia el escenario, se obtiene una *muestra* (en inglés *sample*) del modelo, si es que el rayo interseca con éste. Entonces, si se pinta el píxel con el color de la muestra, estaremos asignando las características de la muestra a un trozo de superficie del modelo, cuya área será igual a la proyección en paralelo del píxel sobre la superficie del modelo (Figura 16).

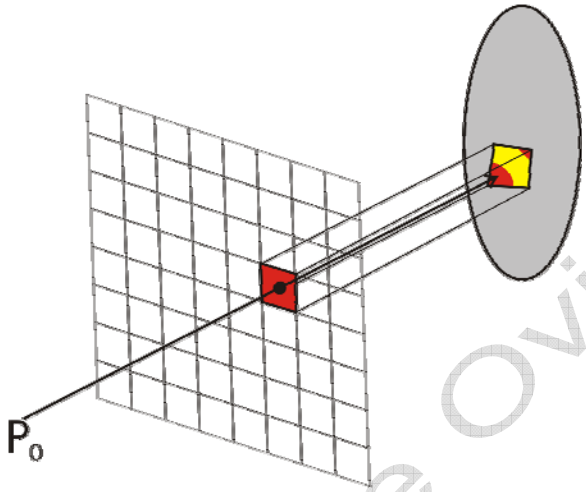

Figura 16: muestra del modelo

Sin embargo, como ocurre en el ejemplo, la superficie del modelo correspondiente a la proyección del píxel *puede tener otros puntos con características diferentes al de la muestra*, por lo que se estaría produciendo una pérdida de información (aliasing) en la imagen sintetizada.

A) Efectos más comunes del aliasing

A continuación veremos, a modo de resumen, algunas de las principales consecuencias sobre la calidad de las imágenes, que se producen a causa de la pérdida de información.

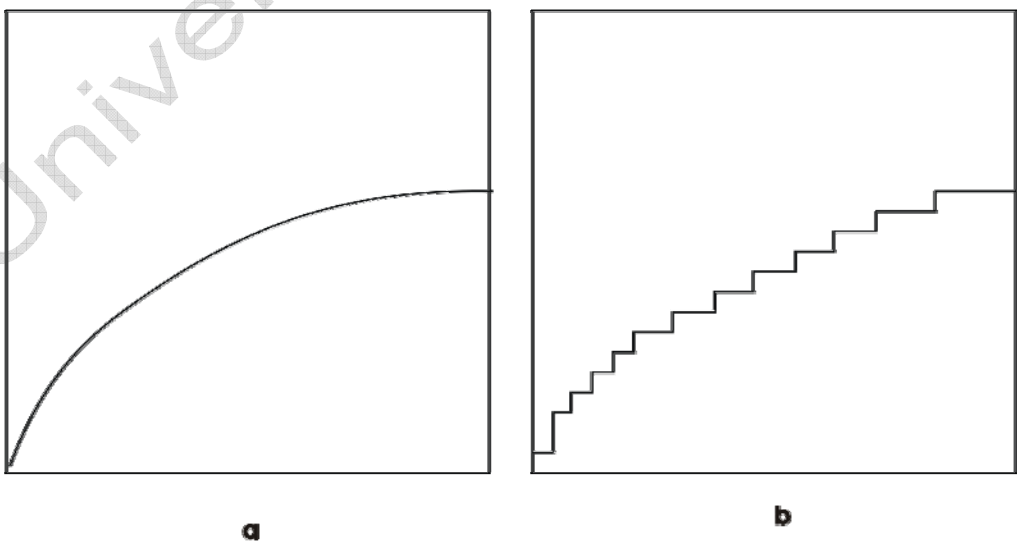

Figura 17: aliasing sobre curvas y perfiles

Esta figura muestra la manifestación más usual del aliasing, que con-

siste en que las líneas curvas (y los perfiles en general) se transforman en curvas dentadas debido a la geometría rectangular de los píxeles. Este efecto es menor cuanto mayor sea la resolución del dispositivo gráfico. También se aprecia menos cuando la iluminación del fondo y de la imagen es la misma, pero el color es diferente, dado que entonces el ojo es más sensible a la diferencia de luminosidad, que a la diferencia de color.

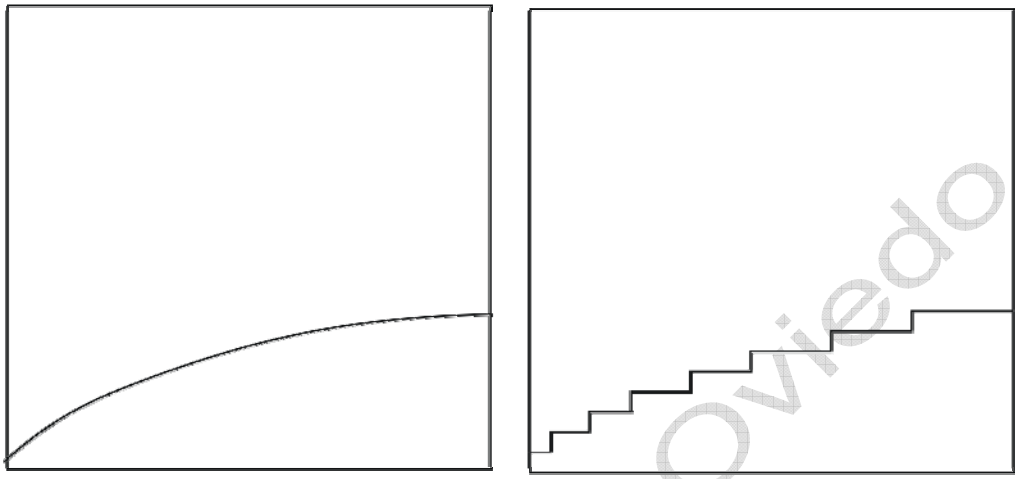

Figura 18: el aliasing de las curvas también depende de su orientación

El dentado depende también del grado de orientación de la curva. Como se ve en esta figura, el dentado es mucho mayor que el de la Figura 17, al tratarse de una curva más plana.

Otro de los efectos del aliasing consiste en la no apreciación de objetos cuya superficie sea menor que un píxel; también puede darse el caso de la existencia de objetos mayores que un píxel, pero estrechos, de modo que su aparición en la imagen dependerá de su ubicación con respecto a los puntos de muestreo.

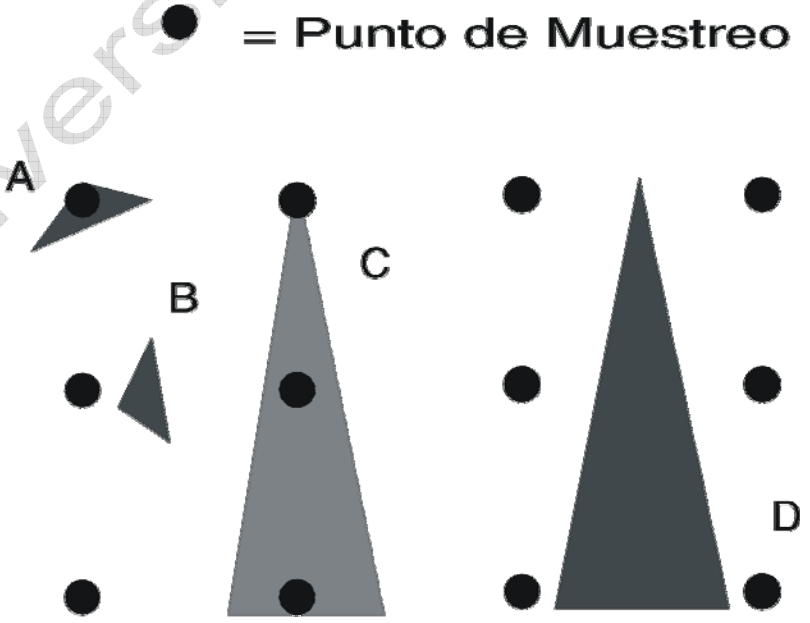

Figura 19: información no accesible al muestreo

En la figura anterior se puede ver que dos objetos iguales A y B co-

rrerán distinta suerte durante el trazado, puesto que el objeto A tendrá representación en la imagen, ya que coincide con uno de los puntos de muestreo, mientras que el objeto B no aparecerá en la imagen, al no ser muestreado.

Algo similar sucede con los objetos C y D pues, aunque su tamaño sea mayor que un píxel, ninguna de las muestras alcanza al objeto D, por lo que no será detectado, al contrario del C, que sí figurará en la imagen trazada, auque con una calidad dudosa, como ahora veremos.

Otro efecto común del aliasing es el de la "rotura" de los objetos, que depende de su orientación respecto a la matriz de muestreo (plano visual), es decir, dependiendo del número de puntos de muestreo que inciden sobre un objeto, un factor que varía según sea la posición del objeto, con respecto a la matriz de píxeles.

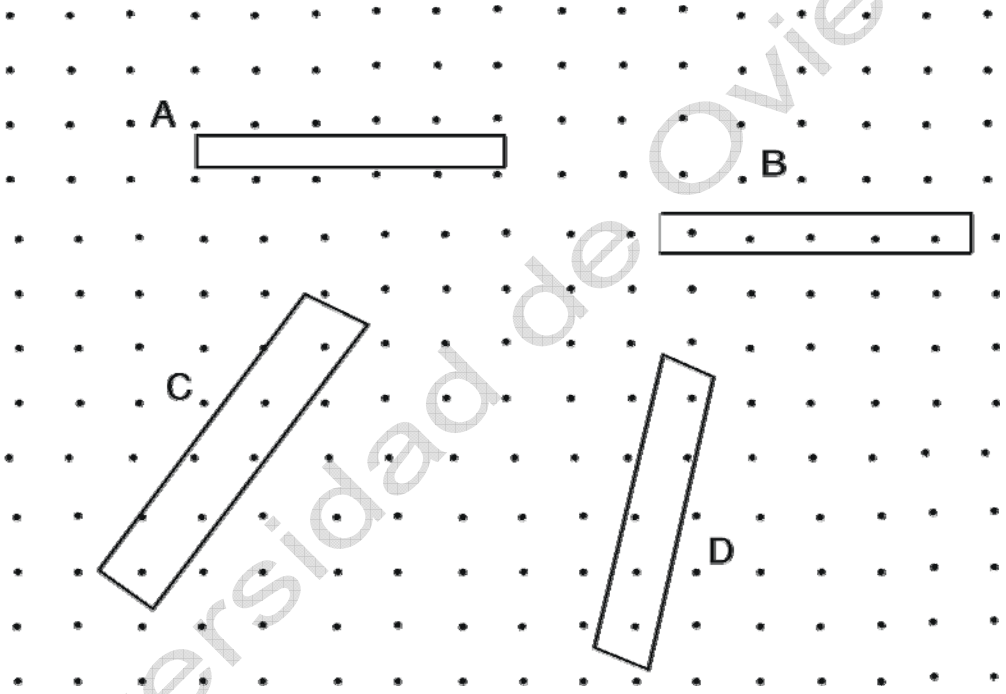

Figura 20: rotura de los objetos

En esta figura, A y B son de nuevo un ejemplo de objetos cuya aparición en la imagen dependerá de su posición en el plano visual.

Por otra parte, el objeto C queda suficientemente representado pues coincide con suficientes puntos de muestreo, mientras que el objeto D quedaría "roto" a causa de que los puntos de muestreo que coinciden con él están en dos columnas diferentes de la matriz de muestreo, por lo que quedará representado por píxeles que están en diferentes columnas, con lo cual el objeto parecerá roto o fragmentado. Como vemos este problema está muy relacionado con los anteriores.

1.4.2 Antialiasing espacial

Aunque el aliasing no se puede evitar completamente, sin embargo es posible reducir sus efectos visuales hasta lograr calidad fotográfica (*realismo*) si

1) Se considera a la superficie de los píxeles como trozos de espa-

cio continuo, y no como puntos discretos.

2) Si en vez de trazar una sola muestra por píxel, se toman varias, es decir, se trazan varios rayos por píxel.

En ray tracing, las técnicas principales que disminuyen el aliasing son:

* **Sobremuestreo** (*supersampling*)

Consiste en lanzar múltiples rayos por cada píxel, distribuidos de forma regular, de forma que la composición de colores obtenidos a partir de esos rayos dará el color de dicho píxel.

El mayor problema que presenta esta técnica es que el número de cálculos necesarios para obtener una imagen se multiplicaría por n, siendo n el total de rayos trazados por cada píxel, aunque el proceso puede optimizarse haciendo que los píxeles contiguos compartan una misma muestra. Por ejemplo, en la Figura 21 podemos ver cómo trazando sólo 25 rayos es posible muestrear una matriz de 16 píxeles, trazando 4 rayos por cada píxel, por lo que en teoría deberían ser 64 los rayos trazados.

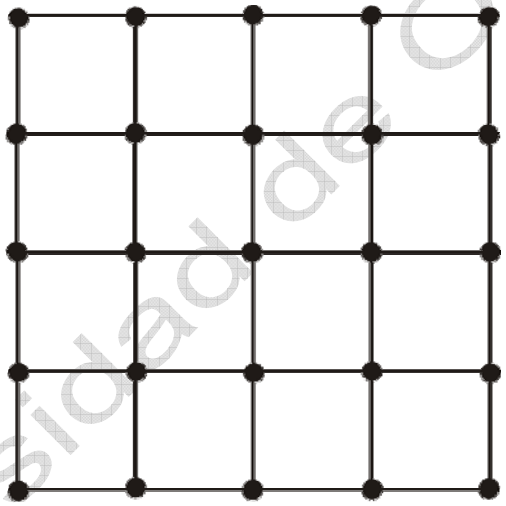

Figura 21: sobremuestreo optimizado

* **Sobremuestreo adaptativo** (*adaptive supersampling*)

Como en el caso anterior, esta técnica traza múltiples rayos por cada píxel, aunque en esta ocasión quedarán distribuidos de modo irregular, basándose en el principio de *trazar más rayos donde más se necesiten*.

En efecto, no siempre es necesario lanzar un número constante de rayos para conseguir paliar los efectos del aliasing; los objetos del escenario suelen tener superficies homogéneas donde no es necesario trazar rayos extra, mientras que en otros lugares se hace preciso tomar un mayor número de muestras.

En el *sobremuestreo adaptativo* primero traza un número establecido de rayos. Seguidamente se analiza en qué zonas es necesario trazar más rayos (aquellas, p. ej., donde el color de las muestras difiera). Una vez determinadas las zonas heterogéneas, se procede a lanzar más rayos allí donde sean necesarios.

En realidad el *sobremuestreo* y el *sobremuestreo adaptativo*, por lo común se utilizan conjuntamente, por lo que se trata de un planteamiento híbrido. Así, en el ejemplo de la Figura 21, si al efectuar el *sobremuestreo* de de 4 rayos en un píxel dado resulta que el color de cada uno de los rayos es diferente, entonces se ha de subdividir la superficie del píxel en 4 trozos, y a continuación se ha de volver a realizar el sobremuestreo en cada subdivisión. También habría que efectuar esta operación en el caso de que los sondeos (muestras) fuesen incapaces de detectar objetos de menor tamaño que la superficie del píxel (o del subpíxel). En definitiva, el proceso de subdivisión debería continuarse hasta que no se produjese ninguna de las situaciones anteriores.

*** Sobremuestreo estocástico**

Los métodos anteriores se basaban en dividir el píxel tratado en varias partes, y en cada una de ellas trazar un rayo. Sin embargo, en muchas ocasiones hay zonas tan finas que no son captadas por las sucesivas subdivisiones del píxel, ya que siempre (y en todos los píxeles) se utiliza la misma trama de división. Con el *sobremuestreo estocástico* también se lanzan múltiples rayos, aunque en esta ocasión son trazados aleatoriamente de forma que se cubra todo el píxel, pero no siempre con la misma trama.

Con esta técnica, además de minimizar los problemas de los planteamientos anteriores, se pueden conseguir efectos ópticos, como la profundidad de campo y penumbras. Sin embargo se plantea el problema del *ruido*. Así, puede ocurrir que un píxel no tome su color correcto debido a la naturaleza aleatoria del proceso. De todos modos, por ser aleatorio, el ruido pasa más desapercibido al ojo humano que los problemas que plantean con otras técnicas, por lo que el sobremuestreo estocástico es una buena solución contra el aliasing, obteniéndose buenos resultados

*** Sobremuestreo estadístico**

En el *sobremuestreo estadístico* se intenta determinar el número de rayos necesarios, a partir de estimaciones obtenidas de muestras trazadas previamente. Por ejemplo, después de trazar cuatro rayos distribuidos regularmente en un píxel, los colores de las cuatro muestras han de pasar unos tests estadísticos que calculan en qué medida la estimación del color real del píxel es aceptable. Si la estimación es buena, se deja ese color; de lo contrario, se deberán lanzar más rayos, y volver a ejecutar las pruebas.

Los parámetros de referencia que utilizan los tests estadísticos para saber si una estimación es buena o no, pueden ser modificados convenientemente, según interese una imagen de alta calidad o una imagen de menor calidad, pero obtenida con mayor rapidez.

1.5 Trazado de rayos distribuido

Hemos visto que el modelo de iluminación utilizado en ray tracing dista mucho de ser perfecto, entre otras razones porque los rayos que aportan las contribuciones reflejada y transmitida, se comportan como si los objetos fuesen transmisores y reflectores perfectos; esto hace que los rayos trazados no

sean diseminados, de manera que mantienen inalterada su densidad, intersección tras intersección, lo que produce una firma o efecto "super-real" del TR convencional.

El TR distribuido (o TR estocástico), desarrollado y presentado por *R. L. Cook*, *T. Porter* y *L. Carpenter* en 1984, utiliza técnicas estocásticas para la generación de imágenes más realistas, que incluyen toda una serie de efectos que no aparecen en el TR convencional. Esto se consigue introduciendo cierta aleatoriedad en el modelo de iluminación. Concretamente, la aleatoriedad se añade:

* a la hora de distribuir los rayos sobre la superficie de los píxeles
* cuando se realizar el trazado de los rayos de reflexión y transmisión
* al calcular las trayectorias de la luz en las lentes de las cámaras
* al realizar la animación de los objetos e imágenes.

Veamos las ideas generales sobre la distribución aleatoria de los rayos de luz en los pasos anteriores.

La presencia de métodos estocásticos en las fases anteriores genera efectos "borrosos" que permiten aliviar el problema de la pérdida de información (aliasing), así como producir toda una serie de efectos realistas en las imágenes sintetizadas.

Por lo común, para el muestro del píxel se utiliza el sobremuestreo estocástico estudiado anteriormente. Sin embargo, un muestreo completamente aleatorio puede dar lugar a una agrupación de los rayos trazados en una región pequeña de la superficie del píxel. Para evitar este problema se suele dividir el píxel, aplicando a continuación el sobremuestreo estocástico en cada uno de los subpíxeles. Normalmente se utiliza una función aleatoria que desvía las coordenadas del punto de trazado del rayo del centro del píxel. Esta técnica se conoce como vibración (jittering).

Por otro lado, las trayectorias de reflexión en los puntos de intersección son distribuidas aleatoriamente, alrededor de la trayectoria teórica de reflexión especular (**R**). Utilizando, por ejemplo, la componente especular del modelo de Phong ($\cos^n\phi$), se puede encontrar el espectro de trayectorias posibles. Este espectro se divide en zonas angulares (p. ej., 16), y partiendo del centro de la zona angular (vector **R**), se introduce una vibración aleatoria en la trayectoria del rayo reflejado.

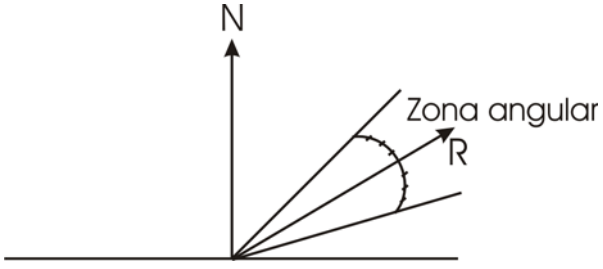

Figura 22: zonas angulares del espectro de trayectorias de reflexión

Al distribuir los rayos reflejados de acuerdo con la función de distribución especular se logran efectos de superficies pulidas (*blurry reflection*).

Si el rayo es transmitido, se procede de igual modo. Distribuyendo los rayos transmitidos, se logran efectos de superficies translúcidas (*blurry transparency*), o sea, aquellas que permiten pasar la luz, pero impiden ver definido lo que hay tras ellas.

Por otro lado, si lo que se distribuyen son los rayos de sombra a través del ángulo sólido de cada fuente de luz, lo que se consigue son *efectos de penumbra*. Además se pueden dar distintos pesos a cada rayo de sombra, considerando así a cada fuente de luz, no como un único punto, sino como una superficie, de forma que cada punto de la superficie posee una intensidad de luz distinta.

Por otra parte, si en el trazado se modelizan cámaras con lentes, introduciendo aleatoriedad en el trazado de los rayos a través de las lentes, se puede obtener el efecto de *profundidad de campo*. La profundidad de campo permite enfocar cierta zona del escenario. Así todos los objetos enfocados aparecerán perfectamente definidos, mientras que los desenfocados se mostrarán borrosos.

Por último, si los objetos tienen movimiento, es posible lograr el efecto de *imagen borrosa* (*motion blur*), si antes de finalizar el trazado de una imagen se realiza un ligero movimiento de los objetos (vibración), y se continúa trazando la imagen hasta finalizar.