



# miniRT

Mi primer RayTracer con la minilibX

*Resumen: Este proyecto es una introducción al increíble mundo del Raytracing. Cuando lo acabe, podrá generar imágenes sencillas y no volverá a tenerle miedo a la implementación de fórmulas matemáticas.*

# Índice general

I.	Introducción	2
II.	Reglas comunes	3
III.	Parte obligatoria	4
IV.	Parte extra	10
V.	Ejemplos	12

# Capítulo I

## Introducción

Existen dos enfoques para generar imágenes en 3 dimensiones sobre un ordenador: La **Rasterización** que, dada su eficacia, es la que utilizan la gran mayoría de motores gráficos y el **Ray Tracing**.

El ray tracing apareció por primera vez en 1968 (ha mejorado desde entonces) y consume bastantes más recursos de máquina **rayterization**. Por eso no está tan adaptado a la modelización en tiempo real. Sin embargo, genera imágenes con un mayor nivel derealismo.



Figura I.1: Estas imágenes han sido modelizadas con la técnica del raytracing. Impresionante, ¿verdad?

Antes de crear gráficos de esta calidad necesitará dominar los conceptos básicos: el **miniRT** es un proyecto en C, humilde pero **funcional**.

El principal objetivo es demostrarle que puede implementar cualquier fórmula matemática o física sin tener que ser matemático.

Nos vamos a conformar con implementar únicamente las fórmulas matemáticas más sencillas.

# Capítulo II

## Reglas comunes

- Su proyecto debe estar programado respetando la Norma. Si tiene archivos o funciones extras, entrarán dentro de la verificación de la norma y, como haya algún error de norma, tendrá un 0 en el proyecto.
- Sus funciones no pueden pararse de forma inesperada (segmentation fault, bus error, double free, etc.) salvo en el caso de un comportamiento indefinido. Si esto ocurre, se considerará que su proyecto no es funcional y tendrá un 0 en el proyecto.
- Cualquier memoria reservada en el montón (heap) tendrá que ser liberada cuando sea necesario. No se tolerará ninguna fuga de memoria.
- Si el proyecto lo requiere, tendrá que entregar un Makefile que compilará sus códigos fuente para crear la salida solicitada, utilizando los flags `-Wall`, `-Wextra` y `-Werror`. Su Makefile no debe hacer relink.
- Si el proyecto requiere un Makefile, su Makefile debe incluir al menos las reglas `$(NAME)`, `all`, `clean`, `fclean` y `re`.
- Para entregar los extras, debe incluir en su Makefile una regla `bonus` que añadirá los headers, bibliotecas o funciones que no estén permitidos en la parte principal del proyecto. Los extras deben estar dentro de un archivo `_bonus.{c/h}`. Las evaluaciones de la parte obligatoria y de la parte extra se hacen por separado.
- Si el proyecto autoriza su `libft`, debe copiar sus códigos fuente y su Makefile asociado en un directorio `libft`, dentro de la raíz. El Makefile de su proyecto debe compilar la biblioteca con la ayuda de su Makefile y después compilar el proyecto.
- Le recomendamos que cree programas de prueba para su proyecto, aunque ese trabajo **no será ni entregado ni evaluado**. Esto le dará la oportunidad de probar fácilmente su trabajo al igual que el de sus compañeros.
- Deberá entregar su trabajo en el git que se le ha asignado. Solo se evaluará el trabajo que se suba al git. Si Deepthought debe corregir su trabajo, lo hará al final de las evaluaciones por sus pares. Si surge un error durante la evaluación Deepthought, esta última se parará.

# Capítulo III

## Parte obligatoria

Nombre del programa	miniRT
Ficheros de entrega	Todos los archivos necesarios
Makefile	all, clean, fclean, re, bonus
Argumentos	Una escena con el formato *.rt
Funciones externas autorizadas	<ul style="list-style-type: none"><li>• open, close, read, write, malloc, free, perror, strerror, exit</li><li>• Las funciones de la lib maths (-lm man man 3 math)</li><li>• Las funciones de la MinilibX</li></ul>
Libft autorizada	Sí
Descripción	El objetivo de su programa es generar imágenes respetando el protocolo de raytracing. Cada una de las imágenes debe representar una escena, vista desde un ángulo determinado, definida por objetos geométricos sencillos, y cada una de las escenas con su sistema de iluminación.

Los requisitos son los siguientes:

- Su proyecto **debe** utilizar la `minilibX`. Puede ser la versión disponible en su OS o una descarga de la versión fuente. Si trabaja con la versión fuente, tendrá que aplicar a su libft las mismas reglas que las descritas anteriormente en las instrucciones comunes.
- La gestión de las ventanas debe ser limpia: no debe haber problemas si pasamos por encima de otra ventana, si minimizamos la ventana, etc.
- Tendrá que implementar al menos 5 objetos geométricos sencillos: plano, esfera, cilindro, cuadrado y triángulo.

- Si procede, se tendrán que gestionar correctamente todas las intersecciones y el interior del objeto.

- Su programa debe ser capaz de cambiar el tamaño de las propiedades definiitoras de un objeto: diámetro para una esfera, tamaño del lado para un cuadrado, diámetro y longitud para un cilindro.
- Su programa debe poder aplicar transformaciones a los objetos, a las luces y a la cámara: traslación y rotación (salvo para las esferas, los triángulos y las luces que no se puedan rotar).
- Gestión de la luz: luminosidad, sombras, luz ambiental (los objetos nunca están totalmente a oscuras). Se tienen que gestionar correctamente las luces de colores y los focos múltiples.
- Por si algún día a Deepthought le interesa evaluar su proyecto, y si desea tener bonitos fondos de pantalla...  
Cuando el segundo argumento sea "**--save**", su programa tendrá que poder guardar la imagen modelizada en formato **bmp**.
- Si no hay segundo argumento, el programa mostrará la imagen en una ventana respetando las siguientes reglas:
  - Al pulsar la tecla **ESC** se debe cerrar la ventana y salir del programa de forma limpia.
  - Al hacer clic sobre el aspa roja de la ventana se debe cerrar la ventana y salir del programa de forma limpia.
  - Si el tamaño declarado en la escena es más grande que el display, el tamaño de la ventana tendrá que ser el mismo que el del display actual.
  - Si hay más de una cámara, se debe poder alternar entre las cámaras con la combinación de tecla que Ud. elija.
  - Se recomienda encarecidamente el uso de **imágenes** de la **minilibX**.
- Su programa debe recibir como primer argumento una descripción de la escena a través de un archivo **.rt**.
  - Este archivo incluirá el tamaño de la imagen o de la ventana, lo que implica que su **miniRT** debe poder utilizar cualquier tamaño  $>0$ .
  - Cada tipo de elemento está separado por una o varios salto(s) de línea.
  - Se puede separar cada tipo de información de un elemento por uno o más espacio(s).
  - Los elementos pueden ir colocados en cualquier orden dentro del archivo.
  - Los elementos que comiencen con una mayúscula solo podrán ser declarados una vez en la escena.

- La primera información de cada elemento será su tipo (uno o dos caractere(s)), seguida de la información específica del elemento.

- Resolución:

```
R 1920 1080
```

- ◊ identificador: **R**
- ◊ x render size
- ◊ y render size

- Luz ambiental:

```
A 0.2 255,255,255
```

- ◊ identificador: **A**
- ◊ Luz ambiental - ratio dentro del rango [0.0,1.0]: **0.2**
- ◊ Colores R,G,B dentro del rango [0-255]: **255, 255, 255**

- Cámara:

```
c -50.0,0,20 0,0,1 70
```

- ◊ identificador: **c**
- ◊ coordenadas x,y,z del punto de vista: **0.0,0.0,20.6**
- ◊ Vector de orientación 3d dentro del rango [-1,1] para cada eje x,y,z **0.0,0.0,1.0**
- ◊ FOV : Campo de visión horizontal en grados en el rango [0,180]

- Luz:

```
l -40.0,50.0,0.0 0.6 10,0,255
```

- ◊ identificador: **l**
- ◊ coordenadas x,y,z del punto de Luz: **0.0,0.0,20.6**
- ◊ ratio de luminosidad dentro del rango [0.0,1.0]: **0.6**
- ◊ Colores R,G,B dentro del rango [0-255]: **10, 0, 255**

- Esfera:

```
sp 0.0,0.0,20.6 12.6 10,0,255
```

- ◊ identificador: **sp**
- ◊ coordenadas x,y,z del punto de la esfera: **0.0,0.0,20.6**
- ◊ diámetro de la esfera: **12.6**
- ◊ Colores R,G,B dentro del rango [0-255]: **10, 0, 255**

- o Plano:

```
pl  0.0,0.0,-10.0  0.0,1.0,0.0  0,0,225
```

- ◊ identificador: **pl**
- ◊ coordenadas x,y,z del punto **0.0,0.0,-10.0**
- ◊ Vector de orientación 3d dentro del rango [-1,1] para cada eje x,y,z: **0.0,0.0,1.0**
- ◊ Colores R,G,B dentro del rango [0-255]: **0, 0, 255**

- o Cuadrado:

```
sq  0.0,0.0,20.6   1.0,0.0,0.0  12.6  255,0,255
```

- ◊ identificador: **sq**
- ◊ coordenadas del punto x,y,z: **0.0,0.0,20.6**
- ◊ Vector de orientación 3d dentro del rango [-1,1] para cada eje x,y,z: **1.0,0.0,0.0**
- ◊ Altura: **12.6**
- ◊ Colores R,G,B dentro del rango [0-255]: **255, 0, 255**

- o Cilindro:

```
cy  50.0,0.0,20.6   0.0,0.0,1.0  10,0,255  14.2  21.42
```

- ◊ identificador: **cy**
- ◊ coordenadas x,y,z del punto **50.0,0.0,20.6**
- ◊ Vector de orientación 3d dentro del rango [-1,1] para cada eje x,y,z: **0.0,0.0,1.0**
- ◊ diámetro del cilindro: **14.2**
- ◊ altura del cilindro: **21.42**
- ◊ Colores R,G,B dentro del rango [0,255]: **10, 0, 255**

- o Triángulo:

```
tr  10.0,20.0,10.0  10.0,10.0,20.0  20.0,10.0,10.0  0,0,255
```

- ◊ identificador: **tr**
- ◊ coordenadas x,y,z del primer punto **10.0,20.0,10.0**
- ◊ coordenadas x,y,z del segundo punto **10.0,10.0,20.0**
- ◊ coordenadas x,y,z del tercer punto **20.0,10.0,10.0**
- ◊ Colores R,G,B dentro del rango [0,255]: **0, 255, 255**

- Ejemplo minimalista de un archivo .rt:

```
R 1920 1080
A 0.2
c -50,0,20    0,0,0    70
l -40,0,30      0.7
pl 0,0,0        0,1,0,0   255,0,225
sp 0,0,20       20
sq 0,100,40     0,0,1,0   30
cy 50.0,0.0,20.6 0,0,1.0 14.2 21.42 10,0,255
tr 10,20,10     10,10,20 20,10,10 0,0,255
```

- Si su programa encuentra cualquier problema de configuración en el archivo, tendrá que cerrarse correctamente y devolver un “Error\n” seguido de un mensaje explícito que Ud. decidirá.
- Para el examen, lo ideal es que proporcione su propio conjunto de escenas con el enfoque de cada uno de los elementos para que resulte más fácil.

# Capítulo IV

## Parte extra

Por supuesto, la técnica RT gestiona bastantes más cosas, como la reflexión, la transparencia, la refracción, al igual que objetos más complejos, sombras ligeras, luminosidad global...

Pero para este `miniRT` hemos preferido no complicar demasiado las cosas. He aquí una serie de tareas extra y sencillas que podrá implementar. Si desea ir más lejos, le recomendamos que en su futura vida de desarrollador programe su propio RT. Eso sí, cuando haya terminado y entregado este.



Figura IV.1: un área, una ventana desde el espacio, y una esfera brillante con textura de globo terráqueo con orografía



Las tareas extra se mirarán si la parte obligatoria está IMPECABLE. Es decir completa, sin errores, ni siquiera con errores menores tales como error de usuario. Si en la parte obligatoria no ha obtenido TODOS los puntos, la parte extra será ignorada por completo.

Lista de tareas extras:

- Normal disruption: por ejemplo utilizar `sine` para obtener un efecto ola.
- Color disruption: por ejemplo, un damero
- Color disruption: un efecto arco iris.
- Luz paralela: la luz sigue una dirección precisa
- Elemento compuesto: Cubo (6 cuadrados)
- Elemento compuesto: Pirámide (4 triángulos, un cuadrado)
- Un tapón en un cilindro de talla limitada
- Otro objeto de segundo grado: Cono, hiperboloide, paraboloide...
- Filtro de colores: Sepia, filtro R/G/B...
- Anti solapamiento
- Estereoscopía simple (como las gafas verdes/rojas)
- Acabado multihilo
- Textura sobre las esferas (uv mapping)
- Gestionar los relieves de la textura (bump map)
- Una preciosa skybox
- Interacciones con el teclado (traslación/rotación) para la cámara
- Interacciones con el teclado (traslación/rotación) para los objetos
- Conseguir que la cámara gire gracias al ratón



Tiene derecho a utilizar otras funciones para completar la parte extra, siempre y cuando su uso esté justificado. También puede modificar el archivo de la escena y sus reglas para las tareas extra.  
¡Sea listo!



Solo necesita validar 14 tareas extra para conseguir el conjunto de los puntos. Escoja bien y no pierda el tiempo

# Capítulo V

## Ejemplos

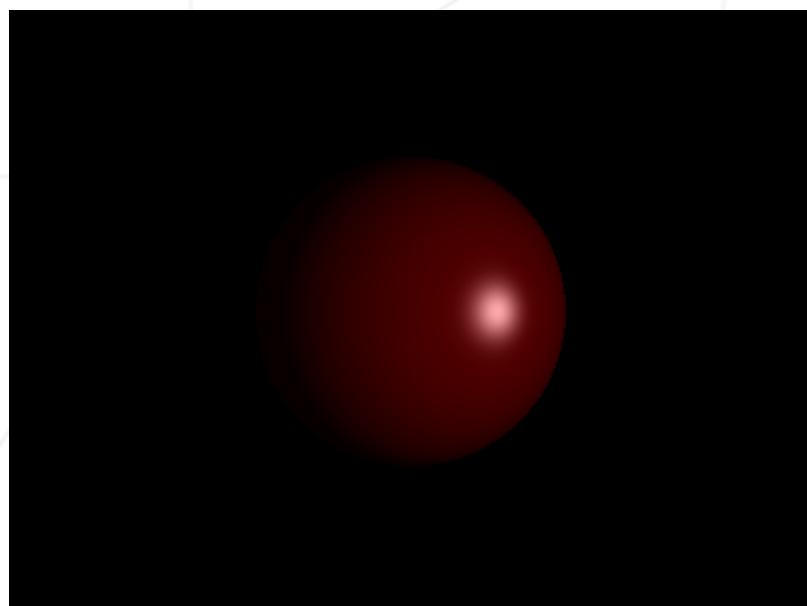


Figura V.1: Una esfera, un foco, un reflejo (opcional)



Figura V.2: Un cilindro, un foco

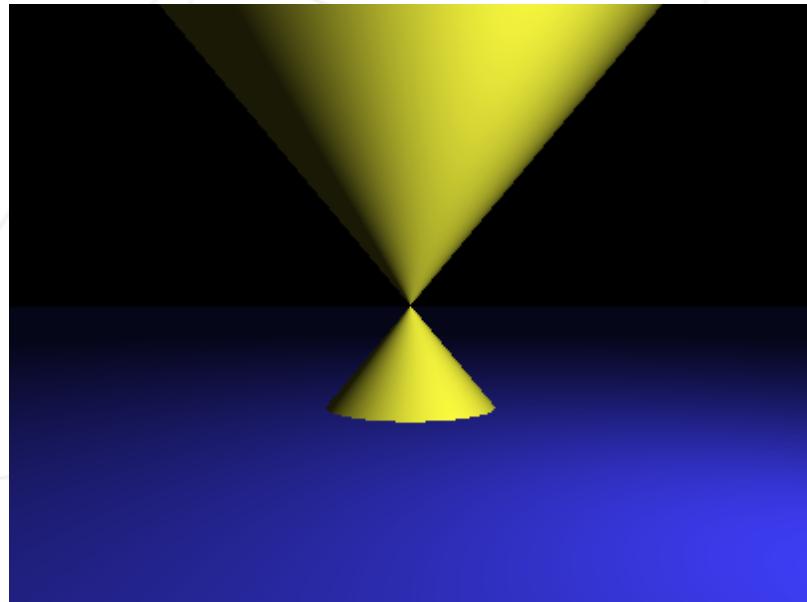


Figura V.3: Un cono (opcional), un plano, un foco

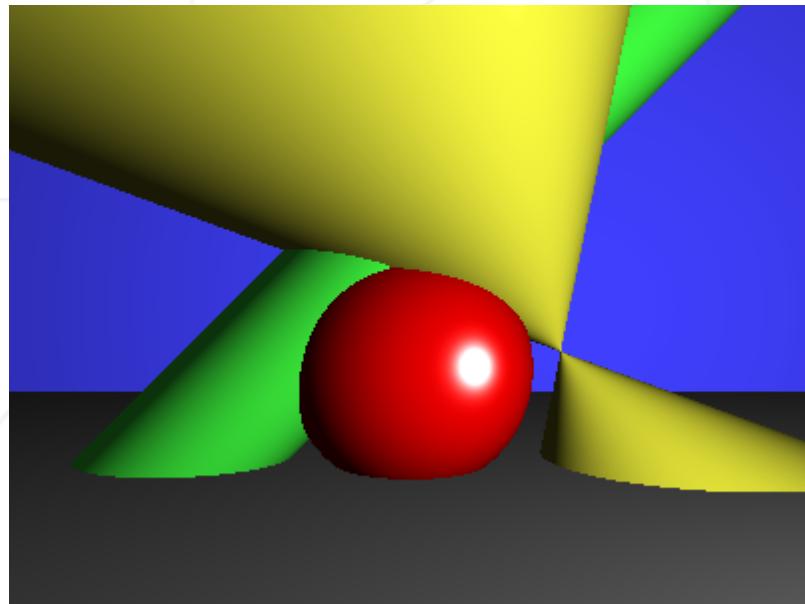


Figura V.4: Un poco de todo, incluidos 2 planos

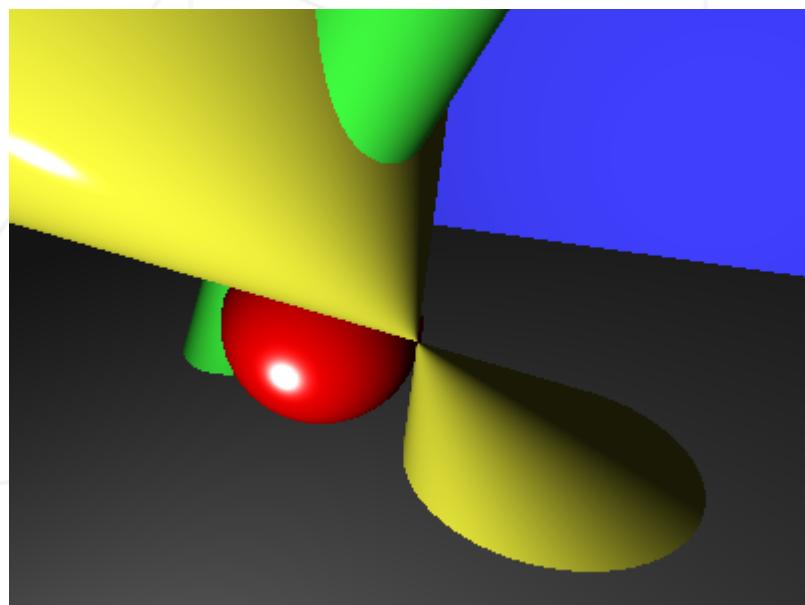


Figura V.5: La misma escena, pero con una cámara distinta

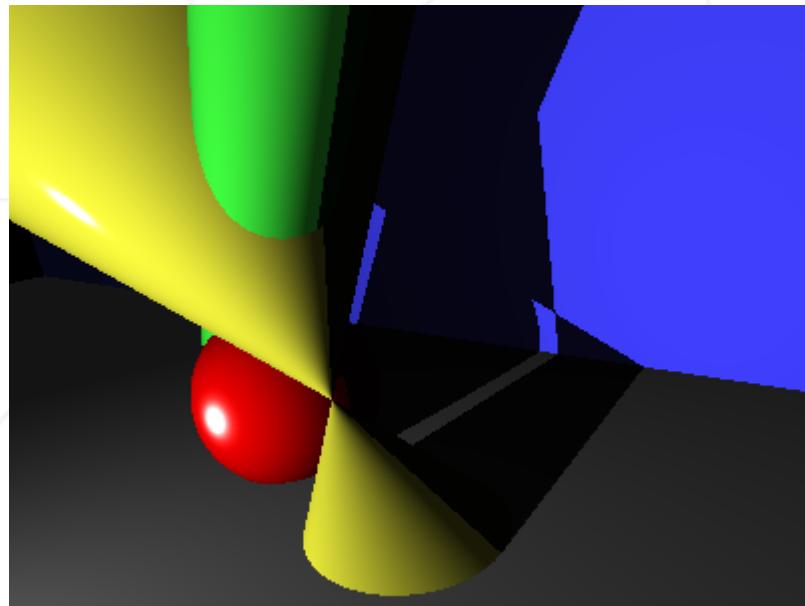


Figura V.6: Ahora, con sombras

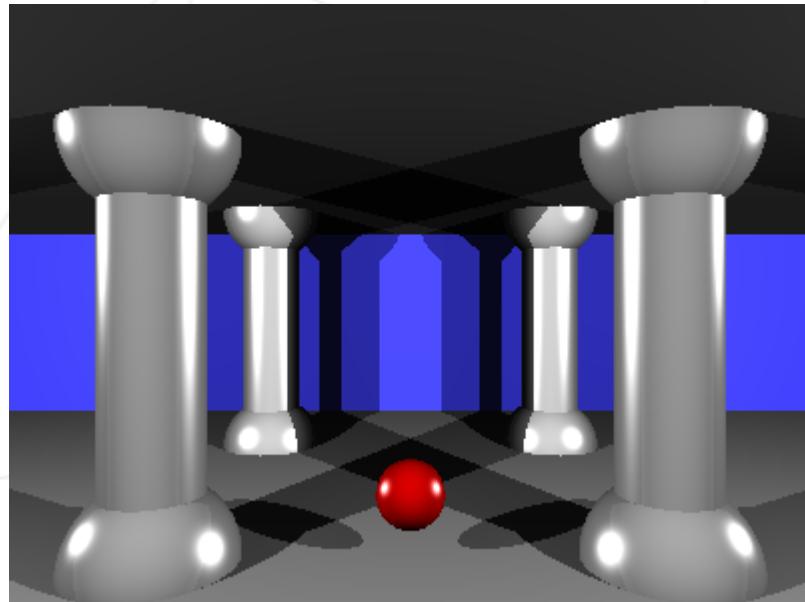


Figura V.7: Varios focos

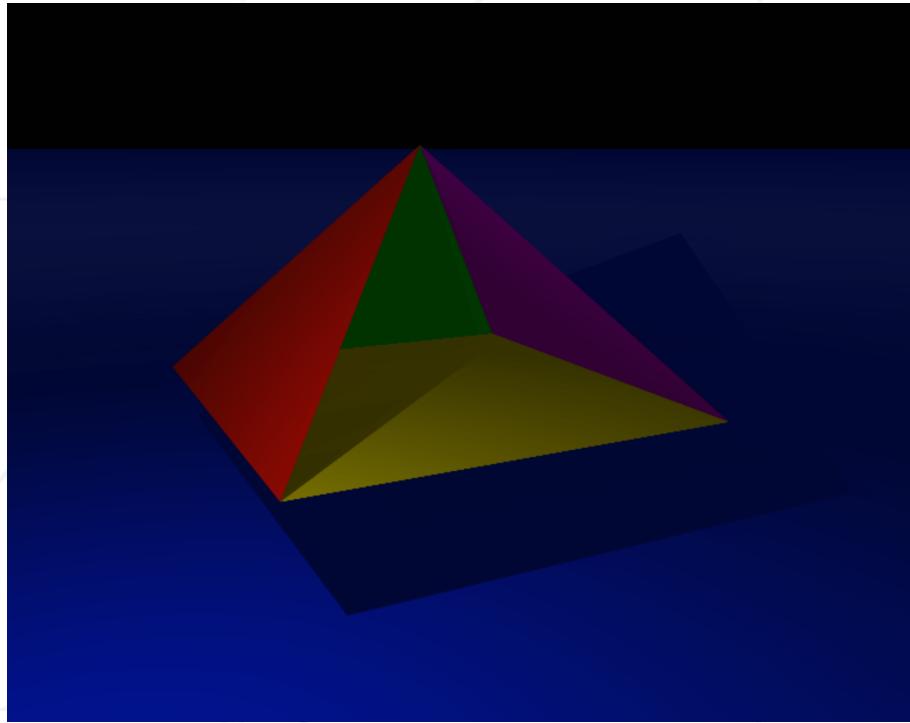


Figura V.8: Un plano, 3 triángulos, 1 cuadrado, ratio de luminosidad

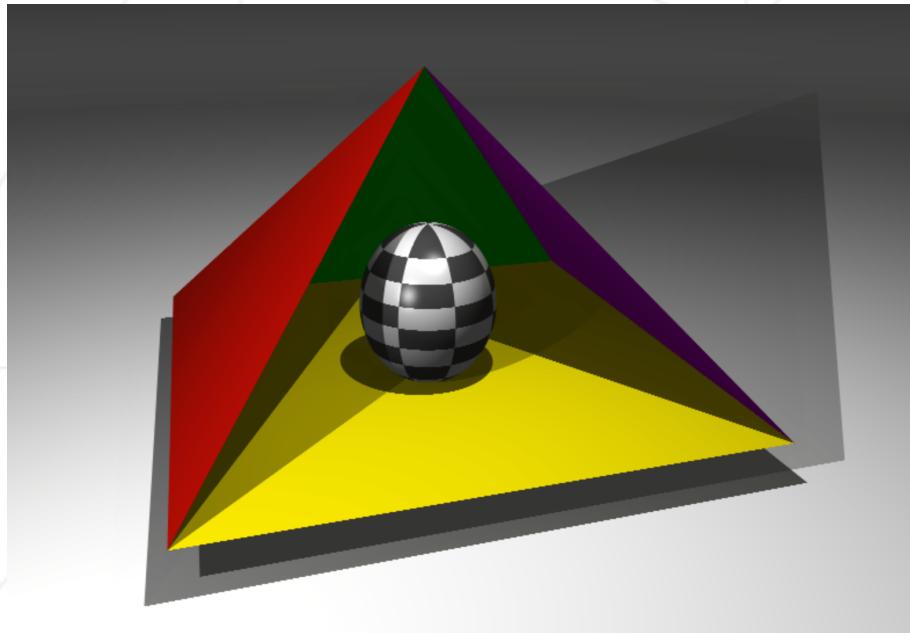


Figura V.9: Por último, varios focos y, en el centro, una esfera ajedrezada reflectante (opcional)