Trabajo práctico Nº2

Integrantes: Casermeiro Maria Silvia - 2013430, Videla Agustin - 1702629 y Vietto Santiago - 1802890

Docente: John Coppens

Institución: UCC

Año: 2022

Desarrollo

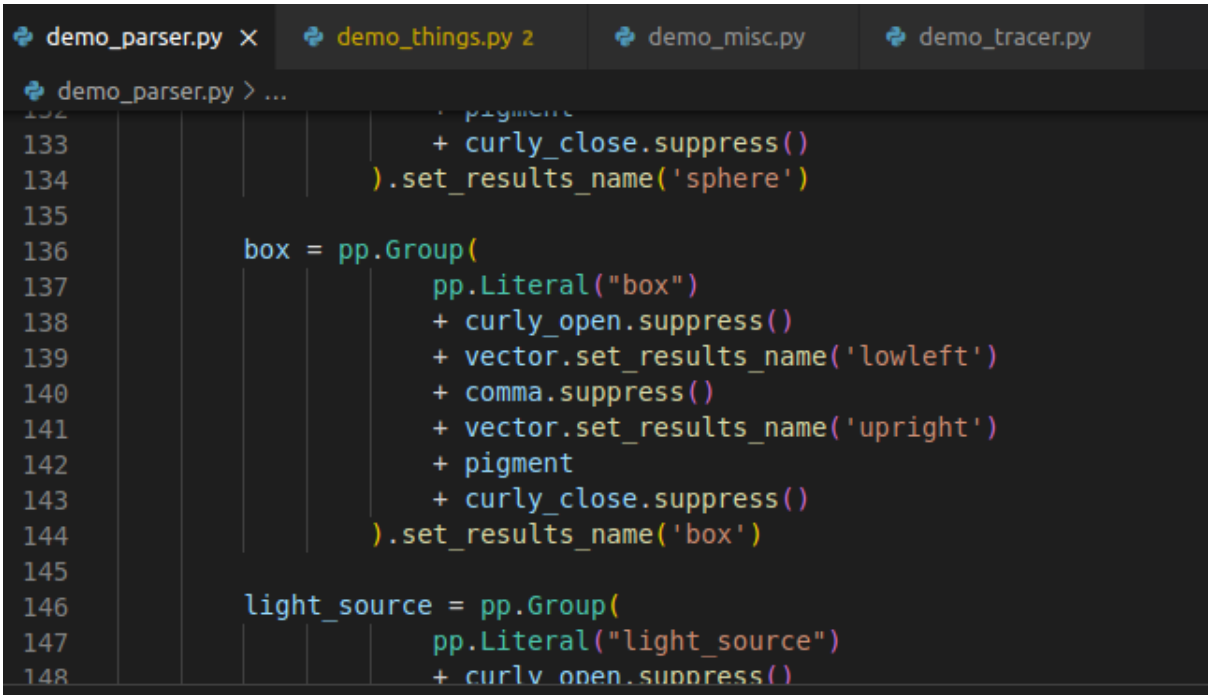
Análisis del código

La ejecución del archivo **demo_tracer.py** permite dibujar una figura en una determinada posición, indicando el color de la misma como así también la luz de la escena, que hace que se ilumine la figura, y un rayo de luz, que hace que se genere una intersección con dicha figura.

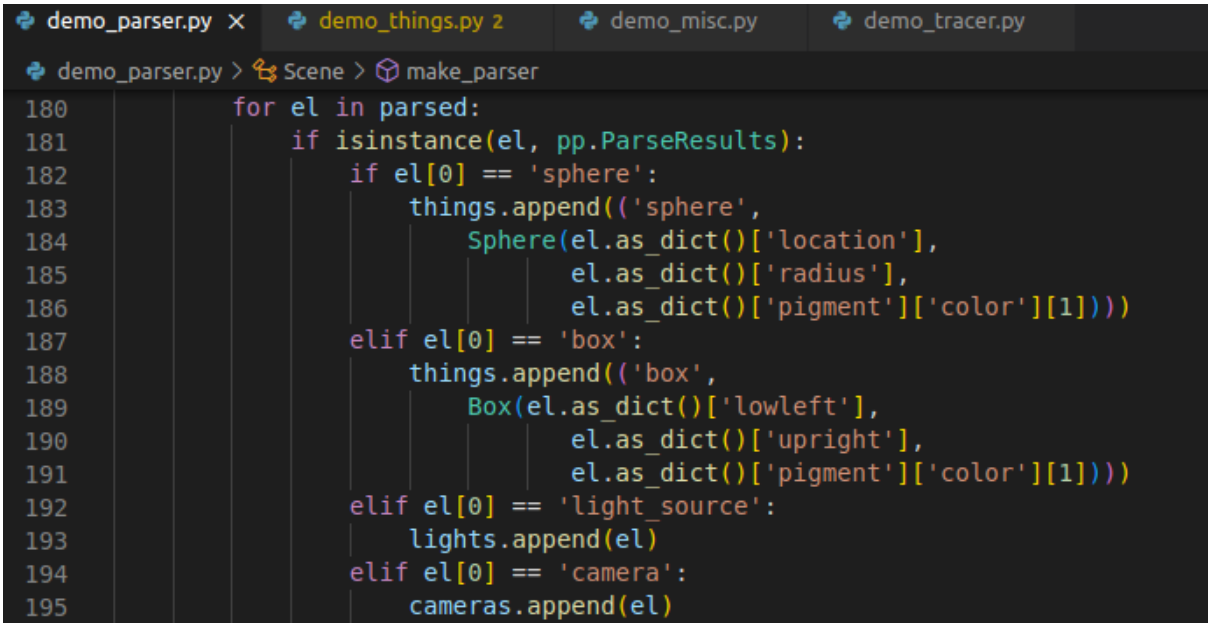
```
silvia-ubuntu@silviaubuntu-Aspire-E5-573:~/Documentos/2° Parcial/demo_tracer-Agus-cubo/demo_tracer$ /bin/python3 "/hom
e/silvia-ubuntu/Documentos/2° Parcial/demo_tracer-Agus-cubo/demo_tracer/demo_tracer.py"
['#include', '"colors.inc"', ['camera', ['location', <demo_misc.Vec3 object at 0x7fc83f579e20>], ['look_at', <demo_mis
c.Vec3 object at 0x7fc83f51f2b0>], ['angle', 50.0]], ['box', <demo_misc.Vec3 object at 0x7fc83f51f790>, <demo_misc.Vec
3 object at 0x7fc83f51f1c0>, ['pigment', ['color', <demo_misc.RGB object at 0x7fc83f5c46a0>]]], ['box', <demo_misc.Vec
3 object at 0x7fc83f58cee0>, <demo_misc.Vec3 object at 0x7fc83f58cd00>, ['pigment', ['color', <demo_misc.RGB object at
 0x7fc83f5c45e0>]]], ['sphere', <demo_misc.Vec3 object at 0x7fc83f58c160>, 1.0, ['pigment', ['color', <demo_misc.RGB o
bject at 0x7fc83f5c4a60>]]], ['light_source', <demo_misc.Vec3 object at 0x7fc83f51f9d0>, ['color', <demo_misc.RGB obje
ct at 0x7fc83f5c4580>]]]
```

Entonces, en el resultado de la terminal, podemos observar como se separa cada término, y si se devuelven todos los archivos significa que el mismo está correcto. De lo contrario va a mostrar el error que encuentra o existe.

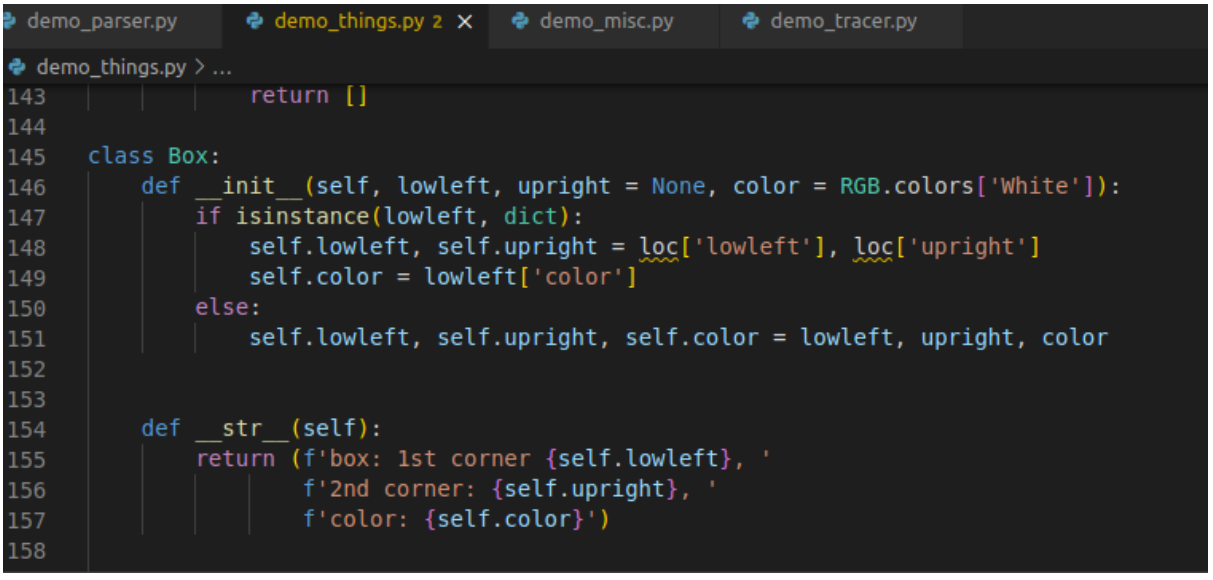
El archivo **demo_parser.py** contiene la definición de la figura que elegimos, el cubo, con sus características. Como aclaración, este mismo archivo contiene la definición de la fuente de luz que agrega iluminación a la escena completa mostrando más o menos brillante.



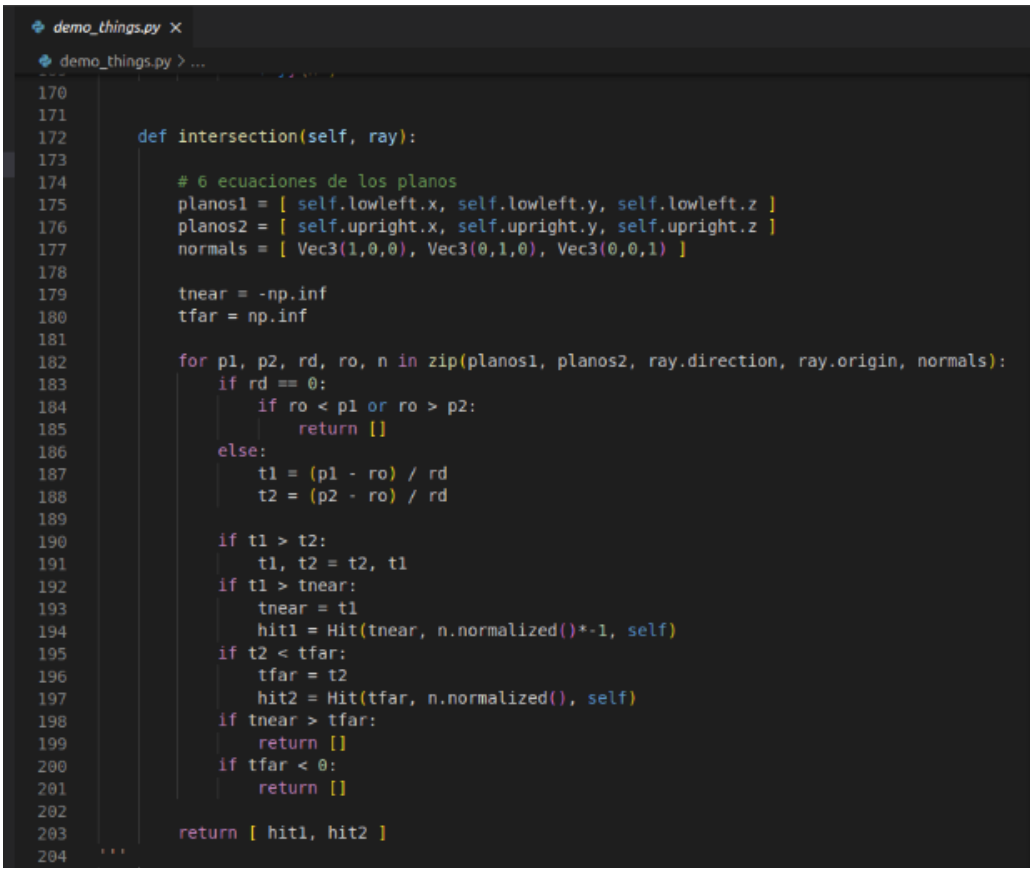
Luego, continúa parseando la imagen del cubo:



Continuando con el archivo **demo_things.py** define la instancia del objeto cubo indicando la posición de la figura en la pantalla:

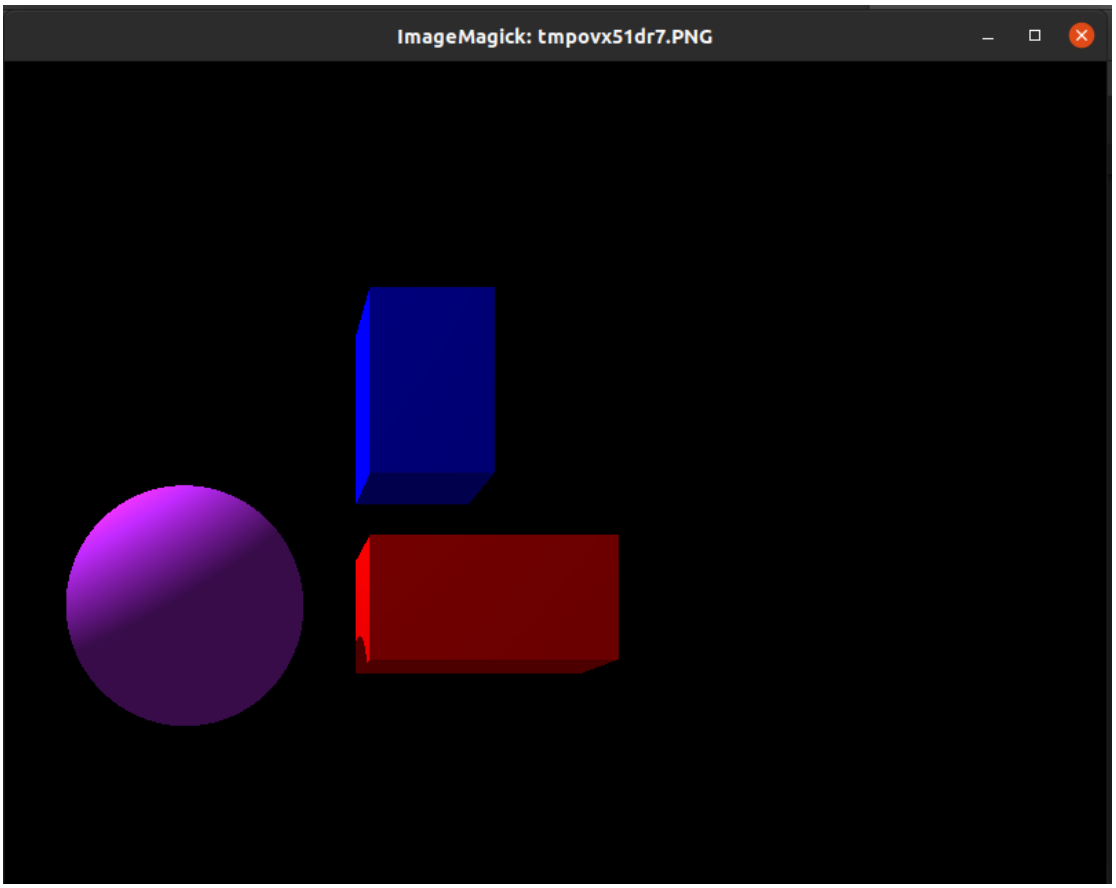


Para determinar si la luz incide sobre la superficie de la figura, hemos seguido las indicaciones del documento *cg_math.pdf* en donde se incluyen las fórmulas que a continuación vamos a explicar:



Comenzamos primero en crear los 6 planos necesarios para determinar la posición del cubo y creamos también la normal (las mismas se definen bajo la referencia de “6 ecuaciones de los planos”), para luego calcular los puntos de impacto del rayo. Luego como podemos observar, se definen dos variables (tnear y tfar) que se utilizan para determinar la orientación del plano y la ubicación del cubo y de la cámara, y como se encuentran uno respecto al otro. Aclaramos que las variables rd y ro es en donde termina y comienza un rayo, y las variables t hacen referencia a la distancia. Y de esta forma continuamos desarrollando el algoritmo tomando como guía el mismo que aparece en el documento *cg_math.pdf* (página 13 - 14).

Para poder corroborar el trabajo, procedemos a ejecutar el archivo **demo_tracer.py**, ya que este es desde donde se ejecuta el test de la imagen que definimos. Como resultado, obtuvimos lo siguiente:



Es importante aclarar que en el archivo **test_mini.pov** se definieron la posición y ángulo de la cámara para poder ilustrar dos figuras de tipo cubo (box), en donde para cada figura se definieron sus colores y posiciones respectivamente:

demo_parser.py | demo_things.py 2 | test_mini.pov | demo_misc.py

test_mini.pov

```
#include "colors.inc"
camera {
    location <-1.7, -1.7, -6>
    look_at <0, 0, 0>
    angle 50
}

box {
    <-0.5,1.0,3.5>, <0.5,2.5,4.5>
    pigment { color Blue }
}

box {
    <-0.5,-0.5,3.5>, <1.5,0.5,4.5>
    pigment { color Red }
}

sphere {
    <-2, 0, 4>, 1
    pigment { color Med_Purple }
}


light_source {
    <-5, 5, 2.5>, color White
}
```