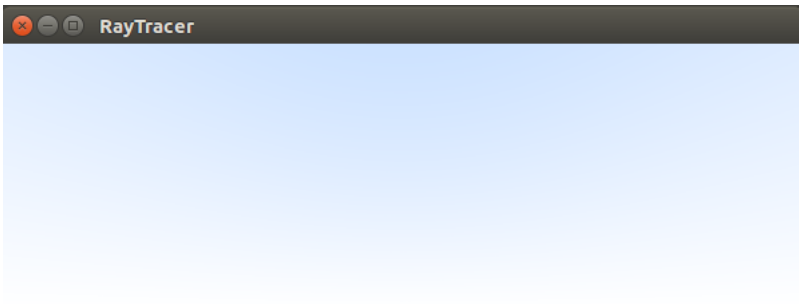
GRÀFICS I VISUALITZACIÓ DE DADES

PRÀCTICA 1

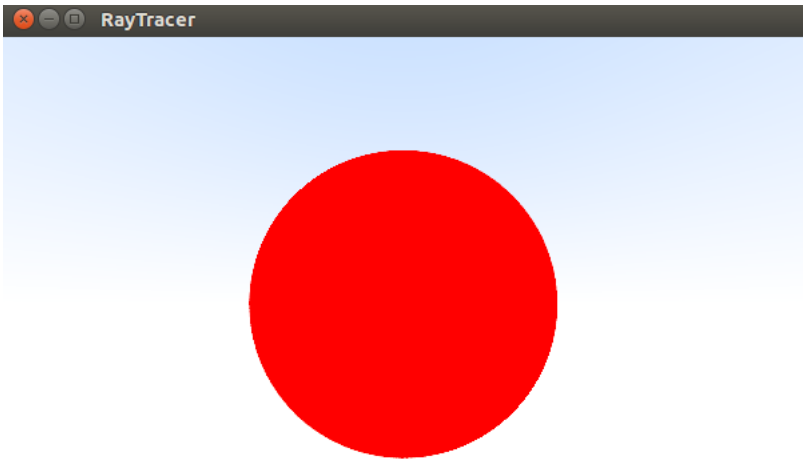
Albert Morales, Sergi Martorell i Arnau Vancells

Fase 0

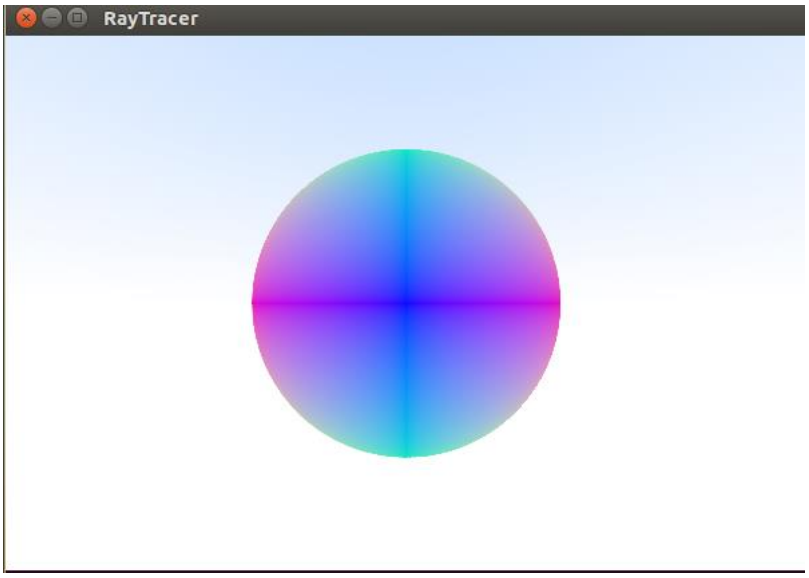
Per a la fase 0 hem modificat el mètode computeColor per a que mostri el fons de la imatge amb un degradat entre el color Blau i el Blanc, fent que els píxels de la part d'abaix de la imatge siguin de color blanc, i tendeixin a blau ascendentment.



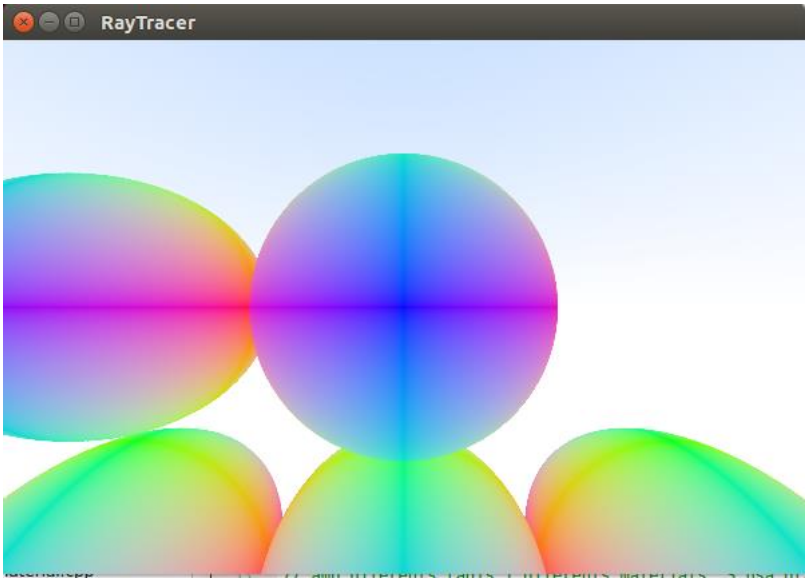
Hem afegit a l'escena una esfera que té color vermell.



Hem afegit a l'escena una esfera que té els colors de les normals calculades en els punts d'intersecció.



També hem creat un mètode que ens permet crear múltiples esferes.



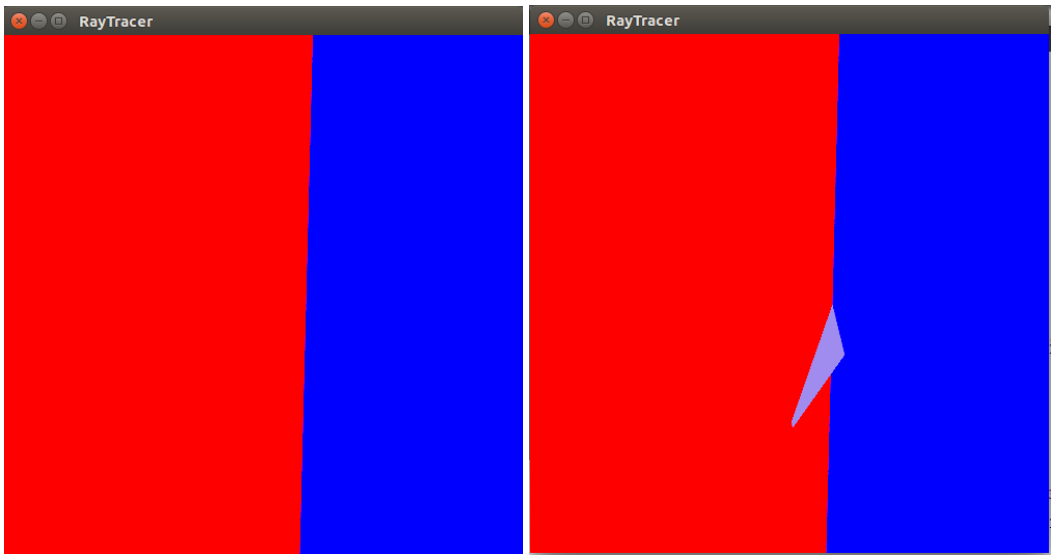
Fase 1

Per a la fase 1 hem afegit els objectes Pla, Triangle i BoundaryObject(malla).

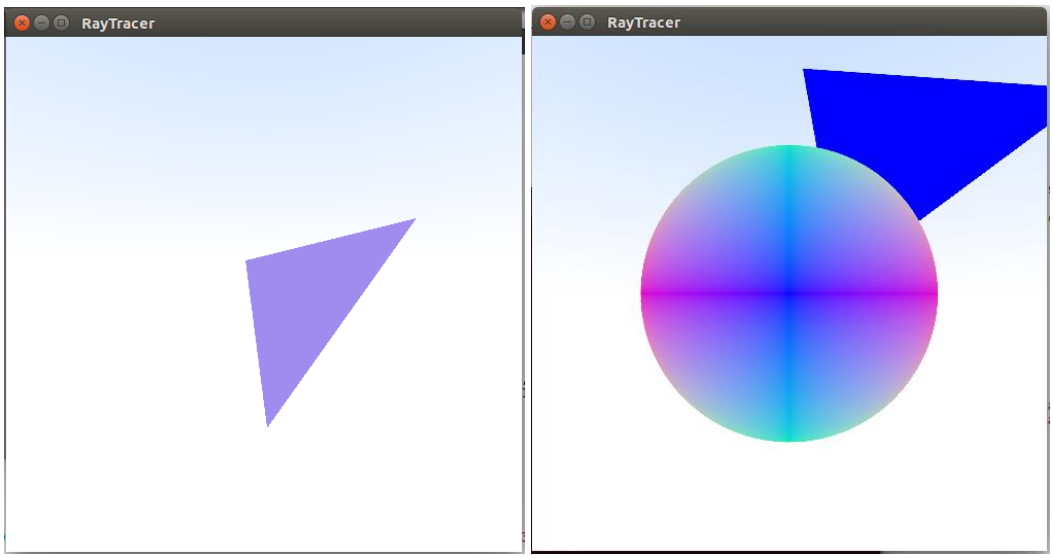
Per a cada objecte nou hem implementat el mètode hit, per tal de calcular si l'objecte es troba en el camí d'un raig, o sigui, calculem la intersecció entre un raig i un objecte.

A mesura que hem implementat les diferents classes hem anat fent escenes amb els diferents objectes:

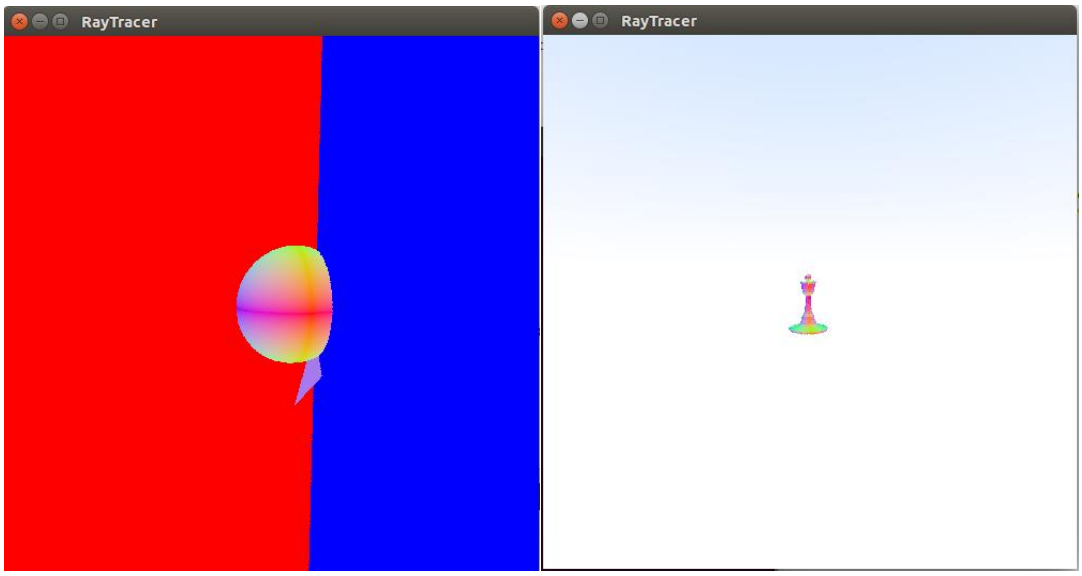
Representació de dos plans / Dos plans i un Triangle



Un triangle / Triangle i Esfera



Dos plans + triangle i Esfera / Peo1K sense transformacions geomètriques



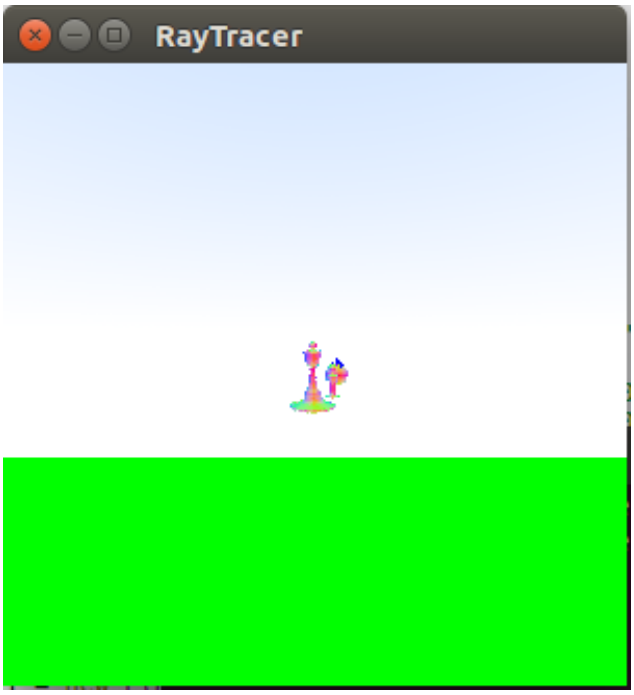
Un cop fetes les classes, hem implementat els mètodes per a aplicar transformacions geomètriques. Per fer això hem modificat els mètodes aplicaTG dels diferents objectes per a que apliquin una transformació geomètrica a partir d'una matriu de transformació 4x4.

Per a fer les transformacions hem creat les classes:

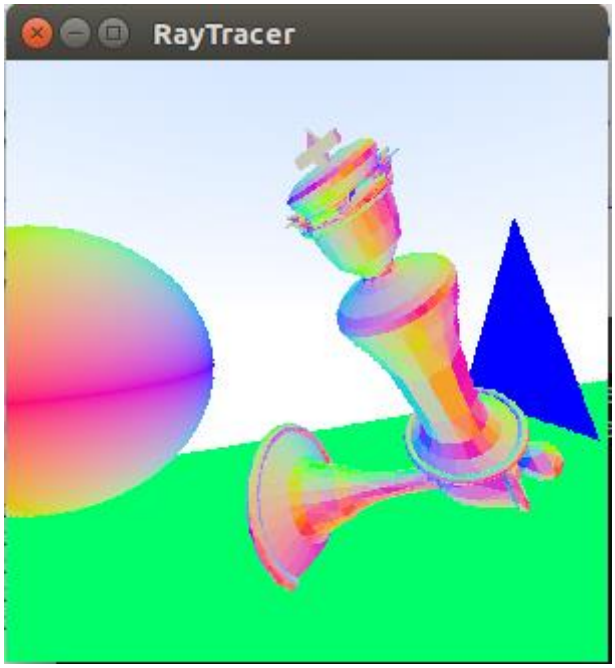
- ***Rotate***: genera una matriu de transformació respecte el centre de l'objecte(giraran sobre si mateixos).
- ***RotateBoundary***: genera una matriu de transformació respecte la posició 0,0,0(necessari per a transformar boundaryObjects).
- ***Scale***: genera una matriu de transformació respecte el centre de l'objecte(creixerà a partir del punt on està).
- ***ScaleBoundary***: genera una matriu de transformació respecte la posició 0,0,0(necessari per a escalar boundaryObjects).

Pels boundaryObjects, abans d'aplicar la transformació s'aplica, sobre cada objecte del boundary, una translació respecte el centre del boundaryObject.

Escena abans de les transformacions



Escena després de les transformacions geomètriques

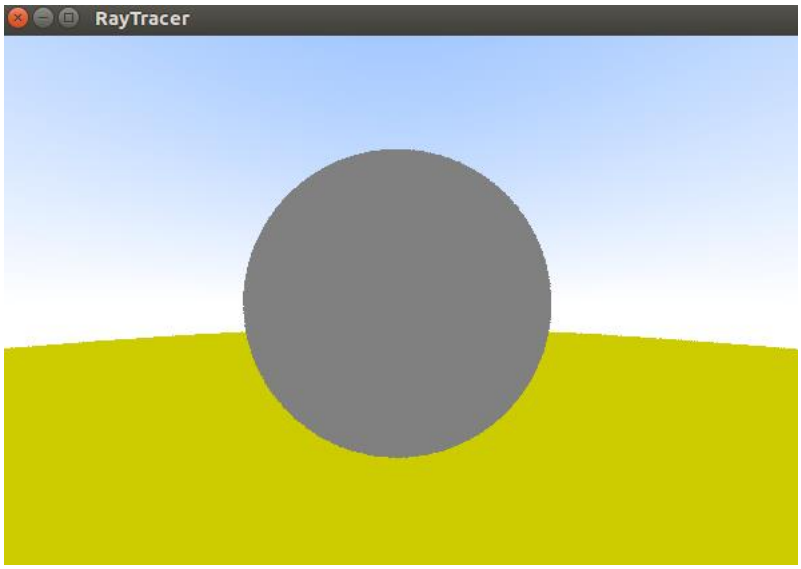


Nota: les imatges de l'última escena tenen una resolució baixa degut a que hem reduït el temps d'execució per a aquesta escena, també té antialiàsing amb 1 sample, que fa que els contorns dels objectes estiguin escalonats.

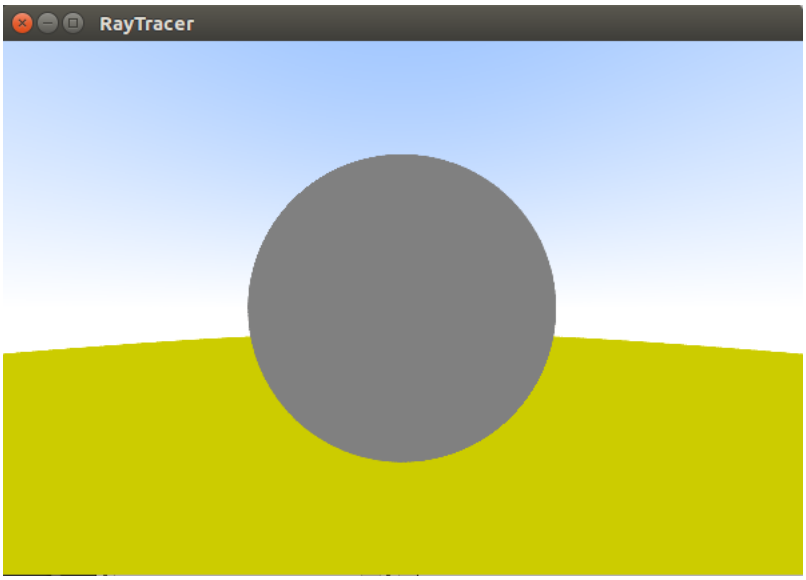
Fase 2

Abans de començar amb la matèria de la Fase 2, hem implementat anti-aliasing per a evitar l'escalonat dels contorns dels objectes en la imatge. Per a controlar això, hem modificat la classe Render, per a que enlloc de fer un mostreig amb un sol raig, faci numSamples mostreigs, aleatòriament, al voltant de la mostra original. D'aquesta forma aconseguim homogeneïtzar els objectes de l'escena.

Sense Antialiasing

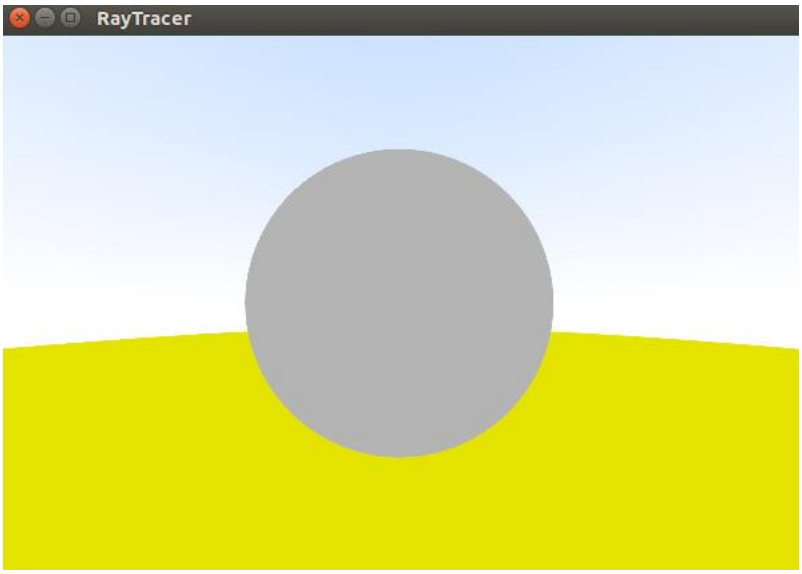


Amb Antialiasing



També hi hem afegit Gamma Correction, fent l'arrel quadrada de cada canal del color abans de pintar-lo, aquesta correcció també la fem a la classe Render, just després d'haver fet les samples de l'antialiasing.

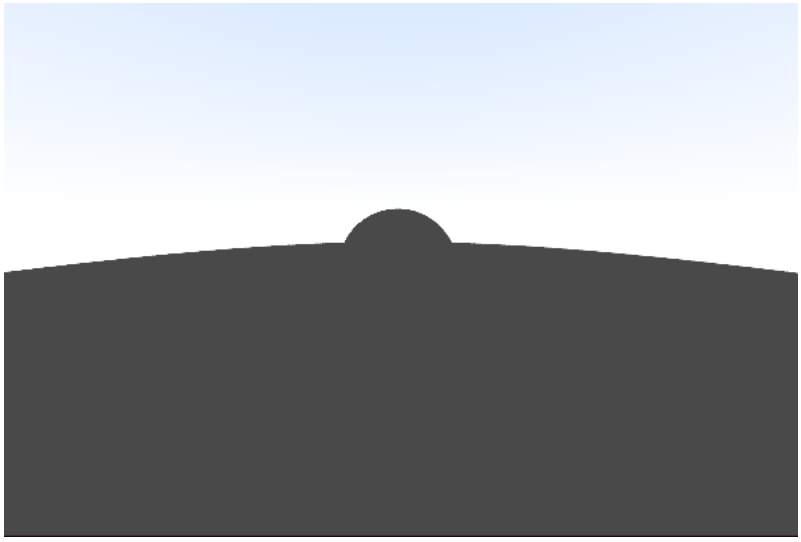
Antialiasing + Gamma Correction



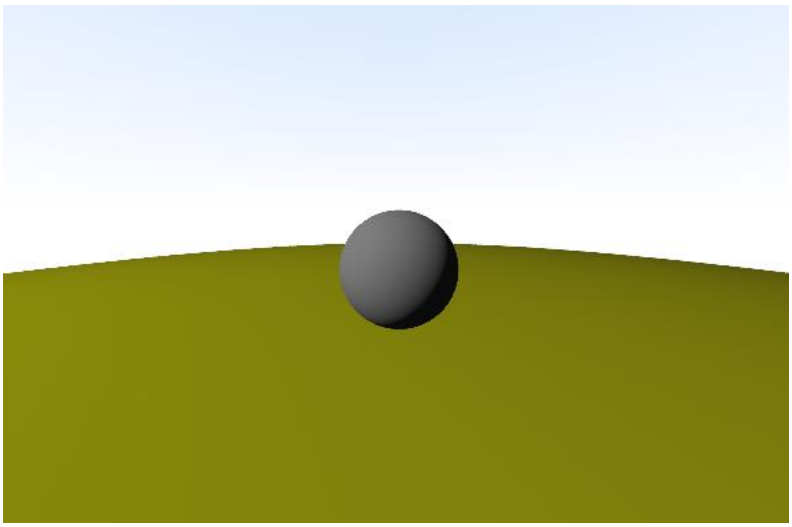
Un cop feta aquesta part "prèvia" a la fase 2, hem afegit llums puntuals i implementat el mètode Blinn-Phong. Per a afegir les llums puntuals hem creat la classe Light, que conté la informació necessària per a representar una llum puntual, la qual farem servir per a calcular els colors en els objectes degut a aquesta llum.

El mètode Blinn-Phong el podem separar en tres càlculs diferents, els càlculs de les diferents components de lluminositat. Hem calculat la component ambient, la difosa i l'especular, i les hem ajuntat per a obtenir els colors que corresponen seguint el mètode Blinn-Phong.

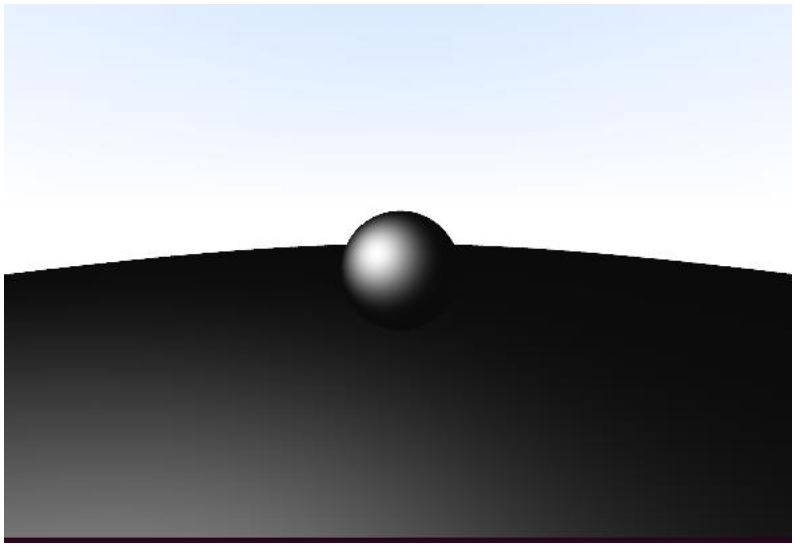
Si només es calcula la component ambient, s'obté:



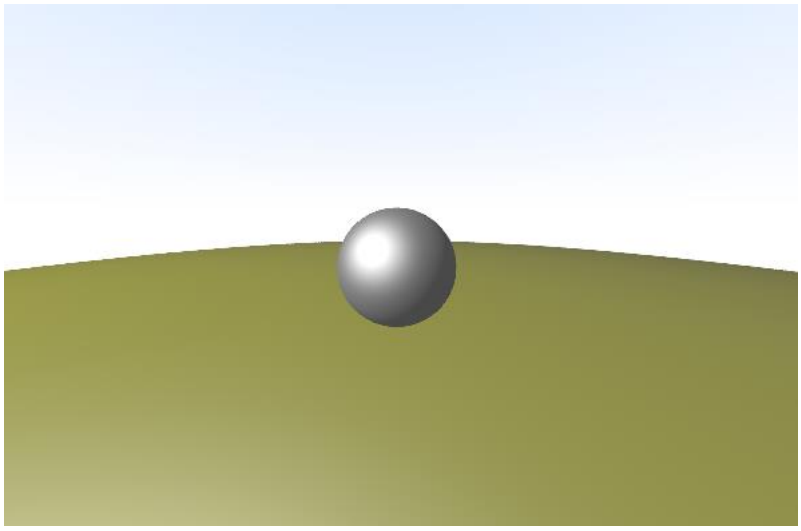
Amb només la component difosa:



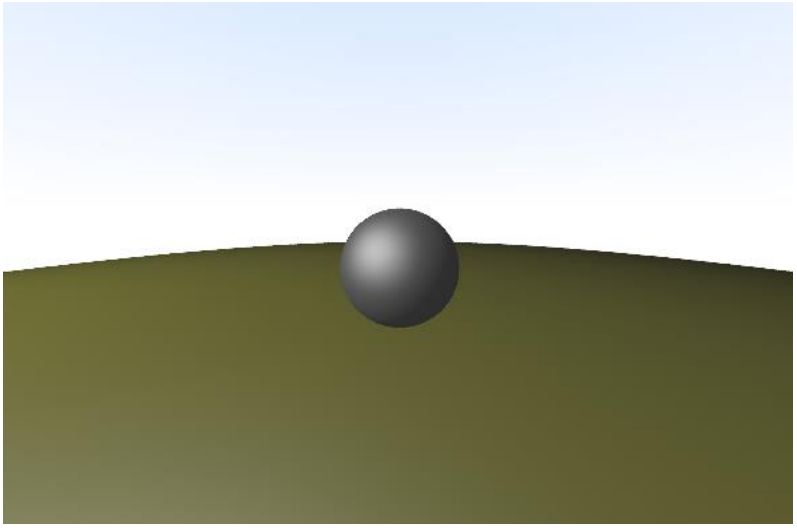
Amb només l'especular:



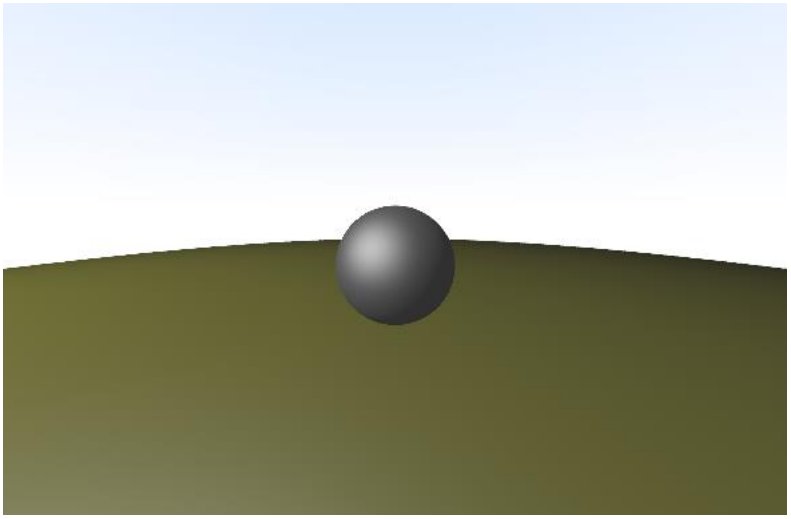
Ara les tres juntes:



I afegint atenuació amb profunditat:

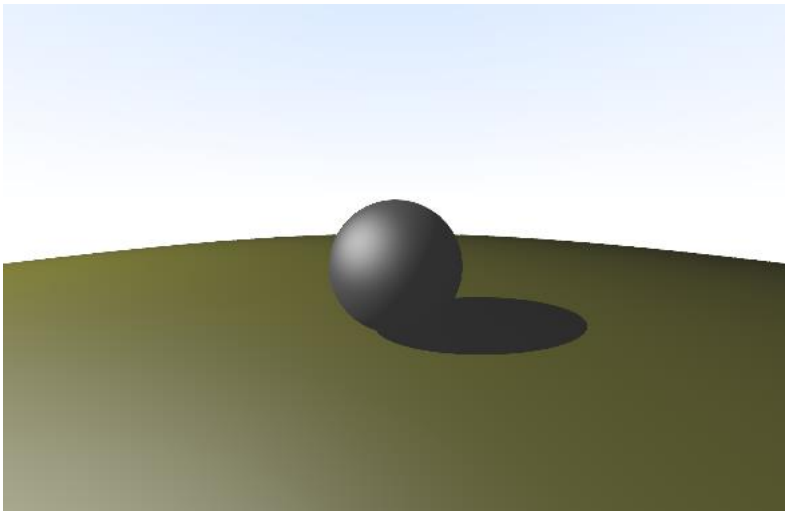


I afegint l'ambient global:



Seguint les transparències de la teoria, hem modificat el mètode Blinn-Phong que teníem per a que es tinguin en compte les ombres que projecten els objectes en l'escena.

Blinn-Phong amb implementació de la ombra:



Escena amb tres plans, dues esferes i un peó

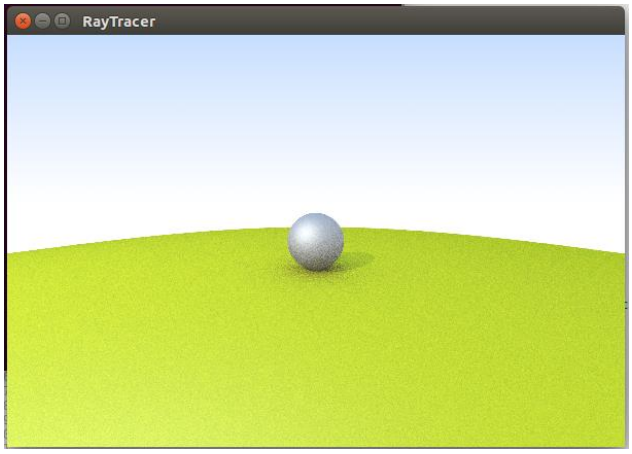
Nota: per a l'escena anterior hem reduït la mida de la imatge a 300x200 i MAXDEPTH a 3, per a evitar una execució molt llarga degut al peó(sense multithread o acceleració per hardware tarda molt).

Nota: per a les imatges anteriors hem modificat el punt de vista de la càmera per tal que el resultat fos lo més semblant possible als enunciats de la pràctica.

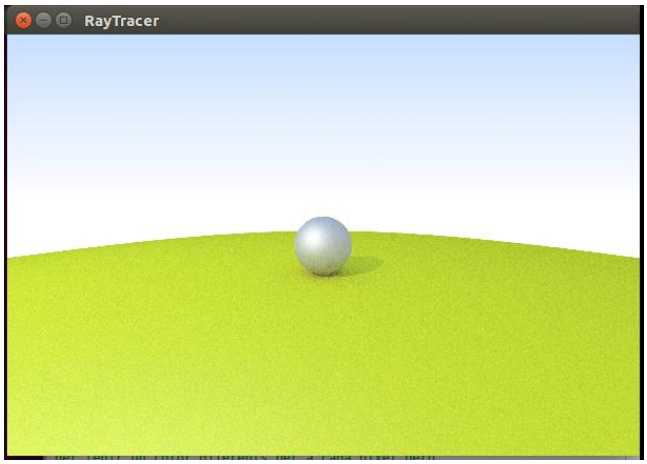
Un cop implementat el mètode Blinn-Phong, hem modificat el mètode ComputeColor de Scene per tal d'afegir recursivitat. Per a fer això hem afegit l'ús del mètode scatter dels materials, i una constant MAXDEPTH per tal de limitar la recursivitat.

Considerem que els rajos secundaris que no intersequen amb l'escena prenen el valor del background.

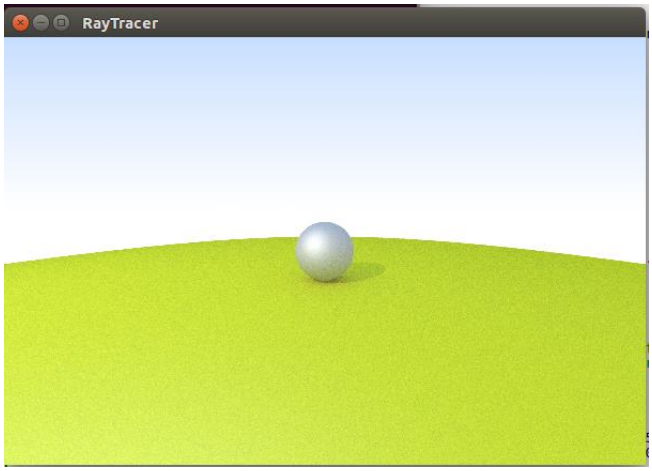
Esfera Lambertiana amb MAXDEPTH=1



Esfera Lambertiana amb MAXDEPTH=3

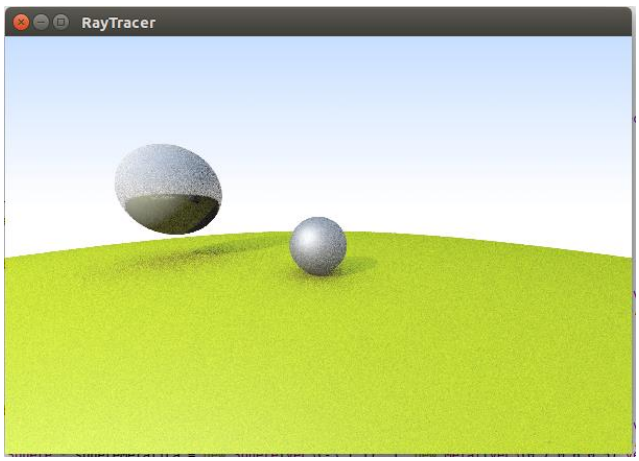


Esfera Lambertiana amb MAXDEPTH=10



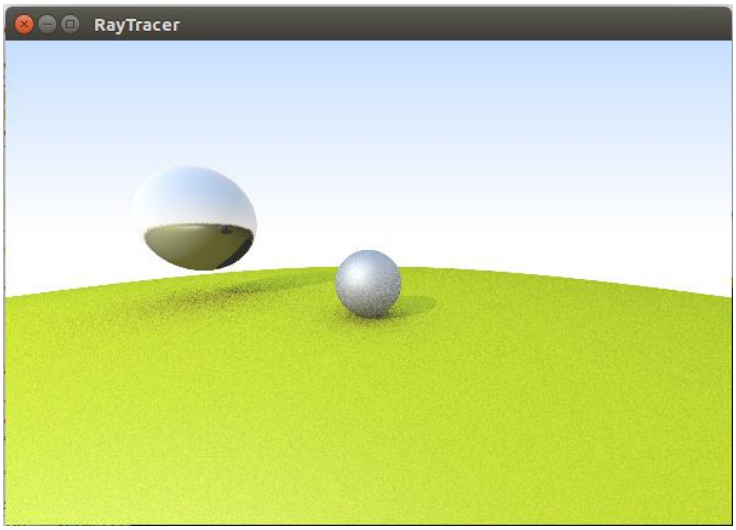
Hem creat una nova classe derivada de Material que codifica el tipus Metal. Aquest material simula la reflexió especular, tal com s'ha explicat a les classes de teoria. Així, hem modificat el mètode scatter per a complir tal fi, calculant el raig reflectit i el color amb el que contribueix el color del raig reflectit a l'escena.

Esfera Lambertiana + metàl·lica amb MAXDEPTH=1 amb acne



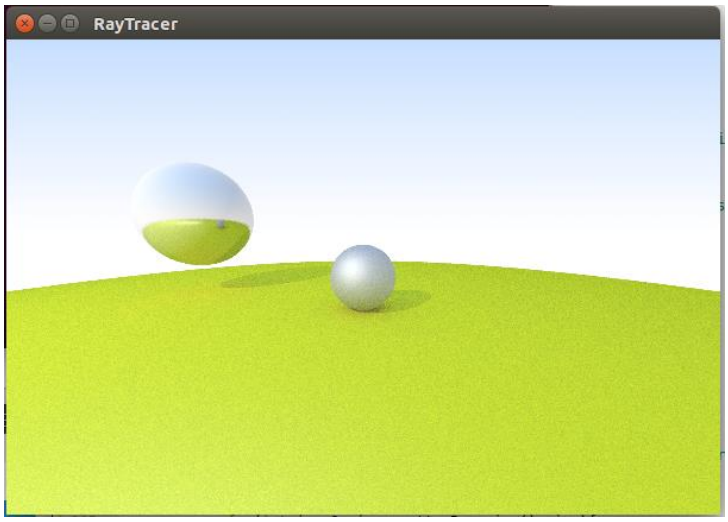
Per a reduir l'acné de les imatges, hem modificat el mètode computeColor per tal de que a l'hora de calcular el color es tingui en compte una petita epsilon, a l'hora de decidir si hi ha intersecció o no amb un objecte. Degut a que hi ha poca precisió, hi haurà casos en que els objectes s'intersecaran amb si mateixos, per això fent servir epsilon desplacem en una mesura molt petita el punt d'intersecció per tal d'evitar el self-shadowing, i per tant, evitar l'aparició de shadow acne.

Esfera Lambertiana + metàl·lica amb MAXDEPTH=1 (fuzzy = 0.05)



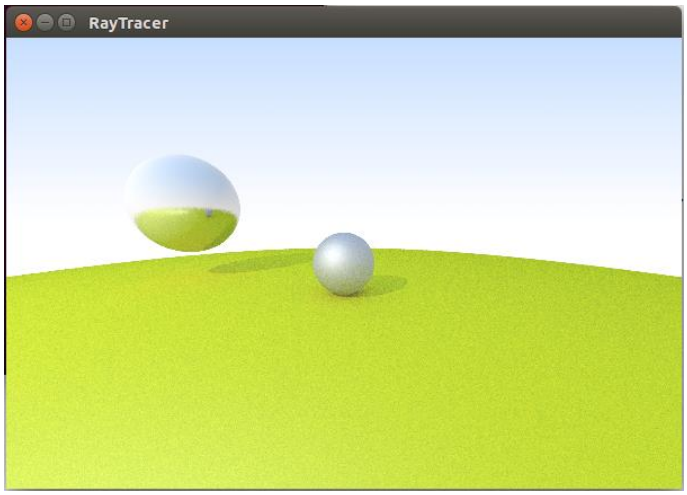
Esfera Lambertiana + metàl·lica amb MAXDEPTH=3 (fuzzy = 0.05)

Esfera Lambertiana + metàl·lica amb MAXDEPTH=10 (fuzzy = 0.05)



Podem modificar el paràmetre fuzzy que estarà entre 0 i 1, i el farem servir per a modificar l'aleatorietat de randomInSphere(). Així, amb fuzzy molt baixa la reflexió serà molt més nítida, i amb fuzzy més alta la reflexió serà més difosa.

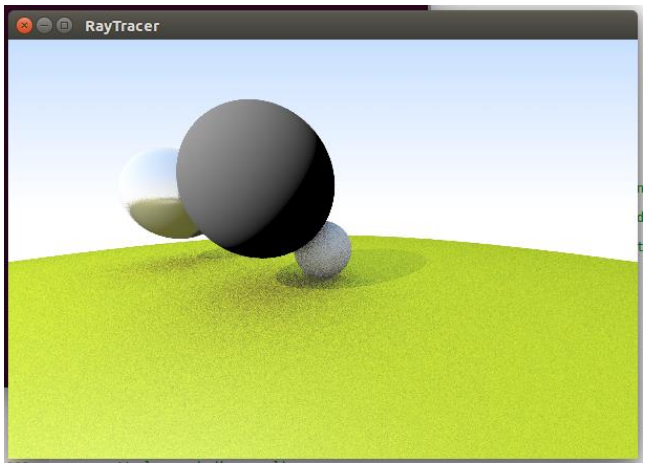
Esfera Lambertiana + metàl·lica amb MAXDEPTH=10 (fuzzy = 0.1)



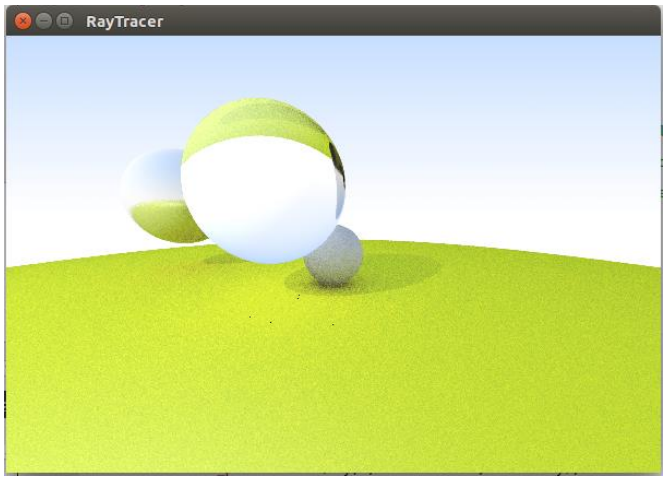
Podriem evitar objectes lambertians rugosos fent servir un paràmetre com hem fet servir a metall, fuzzy, per tal de modificar l'aleatorietat multiplicant el valor de randomInSphere() obtingut per aquest paràmetre. També podríem afegir més rajos reflectits afegint al mètode scatter més rajos aleatoris, i des de computeColor ja que treballem amb un vector de rajos obtingut de scatter no hauríem de modificar-hi res.

Hem afegit un nou material anomenat Transparent. Aquest material permet fer transparències amb el seu mètode scatter. Aquest mètode té en compte si el raig entra o surt de l'objecte, per tal de calcular l'angle del raig refractit, o el raig reflexat, depenent de la situació del raig que s'està tenint en compte.

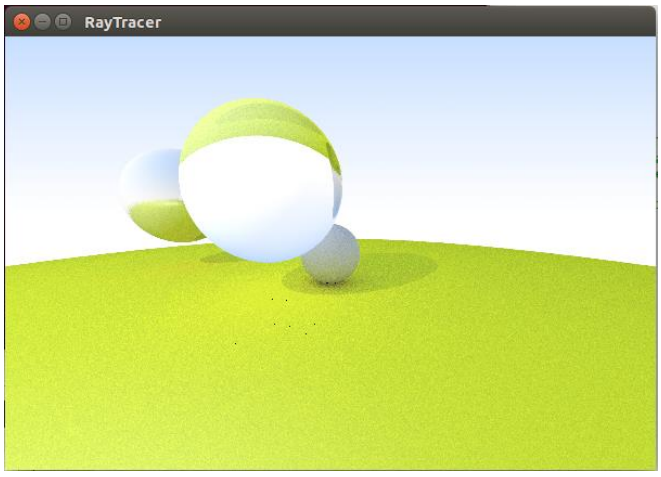
Esfera Lambertiana + metàl·lica + transparent amb MAXDEPTH=1



Esfera Lambertiana + metàl·lica + transparent amb MAXDEPTH=3



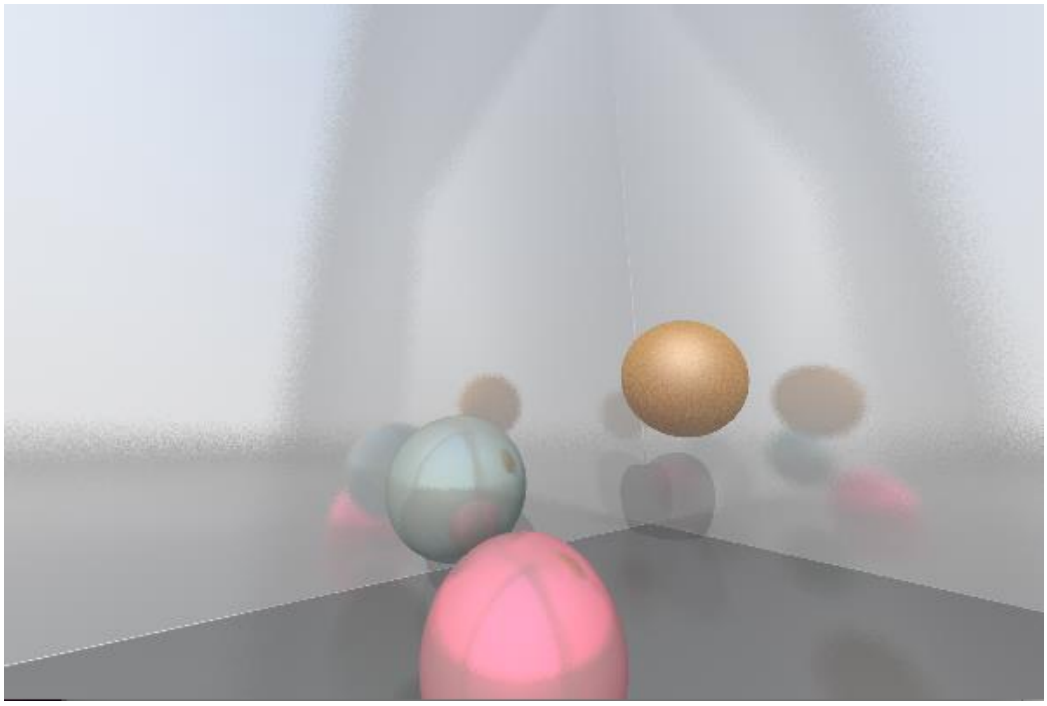
Esfera Lambertiana + metàl·lica + transparent amb MAXDEPTH=10



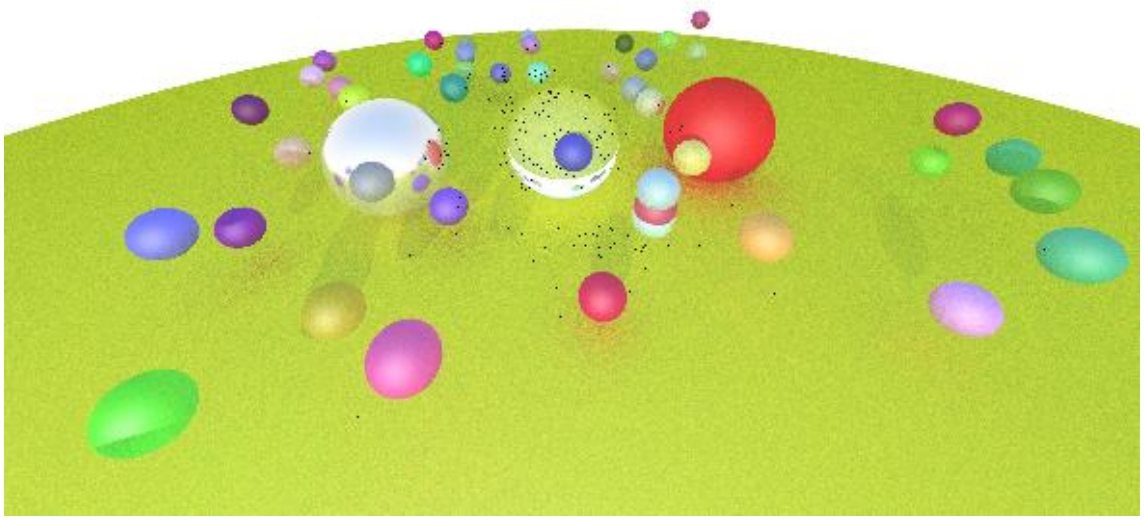
<u>Nota:</u> el punt de vista és diferent del de l'enunciat, ja que les posicions proporcionades no es corresponen amb les imatges de la pràctica, hem assignat un punt de vista per obtenir resultats semblants als de l'enunciat de la pràctica.

Per acabar hem representat diferents escenes amb múltiples esferes:

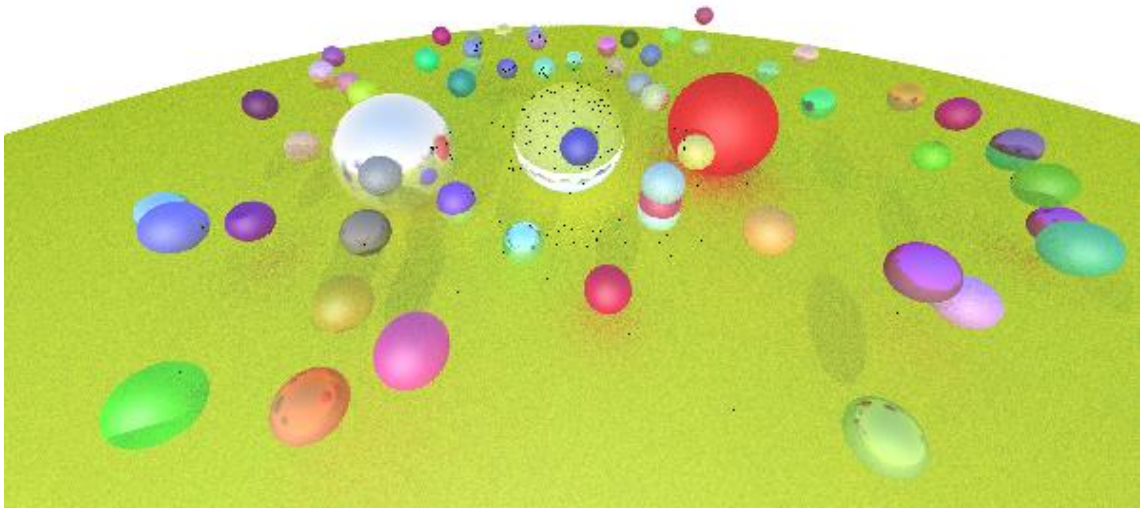
Escena amb 3 plans metàl·lics i 3 esferes(2 metàl·liques i una lambertiana)



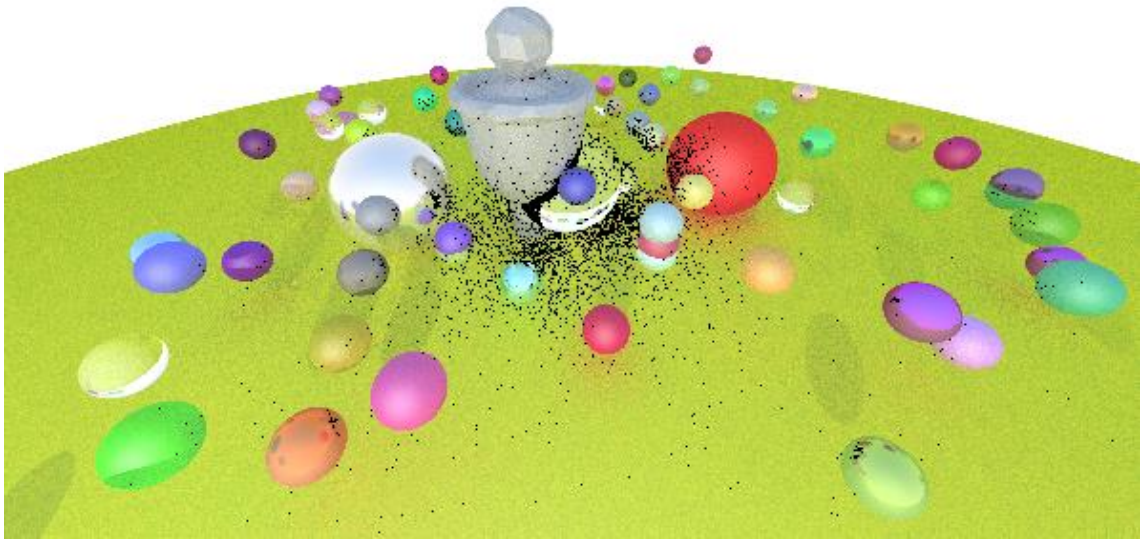
Escena amb 3 esferes(1 metàl·lica, 1 transparent i 1 lambertiana), una esfera gran, i esferes lambertianes aleatòries



Escena amb 3 esferes(1 metàl·lica, 1 transparent i 1 lambertiana), una esfera gran, i esferes lambertianes i metàl·liques aleatòries



Escena amb 1 peó(interseca amb les esferes) 3 esferes(1 metàl·lica, 1 transparent i 1 lambertiana), una esfera gran, i esferes lambertianes aleatòries



En les 3 útlimes imatges podem veure que hi ha acné de color negre. Això és degut a que al intersectar un objecte amb una esfera transparent, hi ha valors que s'assignen a 0. És molt més notable en la imatge del peó. Es podria haver mogut el peó per tal d'obtenir un millor resultat, però el temps d'execució per a mostrar aquesta imatge és molt llarg(fet als ordinadors de la universitat) i no hem pogut fer les comprovacions necessàries per a posicionar-lo bé.

Podríem calcular el color de la ombra dels objectes transparents, de forma que aquesta prengués color de l'objecte transparent. Per a fer això, podríem modificar el mètode Blinn-Phong, per tal de que a l'hora de comprovar si hi ha hit o no amb un objecte per a projectar una ombra, hauríem de comprovar si el material amb el que fa hit és transparent. En cas de ser transparent, podríem sumar a la ombra el color de l'objecte transparent però amb una mica d'atenuació per a que segueixi veient-se com a una ombra(i que no es confongui amb una reflexió).