GiVD - curs 2015-2016 – Pràctica 2

Pràctica número 2

Tema: RayTracing: Implementació d'un RayTracing que permeti simular ombres i reflexions en una escena bàsica que conté primitives bàsiques, com esferes i plans.

Objectiu:
Implementar en C++ tècniques per visualitzar imatges realistes. El principal objectiu és crear un RayTracing capaç de visualitzar un conjunt de primitives bàsiques.

El RayTracing és una de les diferents tècniques per visualització d'imatges per computador. La idea en la que es basa és que les imatges generades estan composades per llum (colors) i que aquesta llum vé de les fonts de llum i rebota per tota l'escena fins a arribar a l'ull (o a la càmera). En els 80, Turner Whitted proposa l'algorisme de RayTracing que simplifica el problema anterior tenint en compte només els rajos que arriben als ull després d'uns quants rebots, podent modelar reflexions directes, indirectes i transparències.

L'algorisme bàsic de RayTracing determina la visibilitat que l'observador té de l'escena a través de cada píxel. Per a cada píxel es calcula un raig de visió que s'intersecta amb l'escena (Figura 1). Aquest raig s'anomena **raig primari**. El color a assignar a cada píxel és la intensitat calculada al punt d'intersecció visible, és a dir, la intensitat del punt intersecció amb l'escena més proper a l'observador. Podeu trobar més informació a la referència [1].

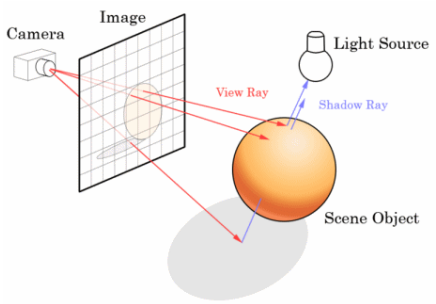


Figura 1: Tècnica de RayTracing: El raig de visió (View Ray) s'anomena **raig primari** i es calcula a partir de la posició de l'observador i el punt de l'àrea de projecció que es rasteritza en el píxel. El raig d'**ombra** (Shadows Ray) és el raig que va des del punt d'intersecció fins a la llum.

L'algorisme principal per a una escena en el buit es pot resumir com:

```
per a cada pixel del viewport
        calcular raig primari
        intersectar Raig-Escena
        si existeixIntersecio
                calcular il·luminació en el punt d'intersecció més proper a l'observador
                assignar al píxel la intensitat calculada
        sino
                assignar al píxel la intensitat de fons (color de background)
        fsi
fper
```

El càlcul de la il·luminació és un dels punts clau del mètode. En principi, en el punt d'intersecció es calcula la intensitat segons el model de Blinn-Phong, però es poden simular efectes com ombres, reflexions i transmissions, traçant rajos d'ombres i rajos secundaris.

Els **rajos d'ombra** estan definits des del punt d'intersecció fins a la font de llum. Es calculen les interseccions dels rajos d'ombra contra els objectes que formen l'escena. En el cas que hi hagi alguna intersecció amb algun objecte, el punt d'intersecció estarà a l'ombra, sinó la il·luminació es calcularà segons el model de Blinn-Phong.

Per a cada punt d'intersecció p_A també es poden generar dos **rajos secundaris**: el raig reflectit i el raig transmès (Figura 2(b)). Per a cada raig secundari es torna a comprovar si hi ha intersecció amb l'escena. En el cas que hi hagi intersecció, p_B, es calcula la contribució segons Blinn-Phong en el segon punt d'intersecció i s'acumula en el color de p_B en el color calculat a p_A.

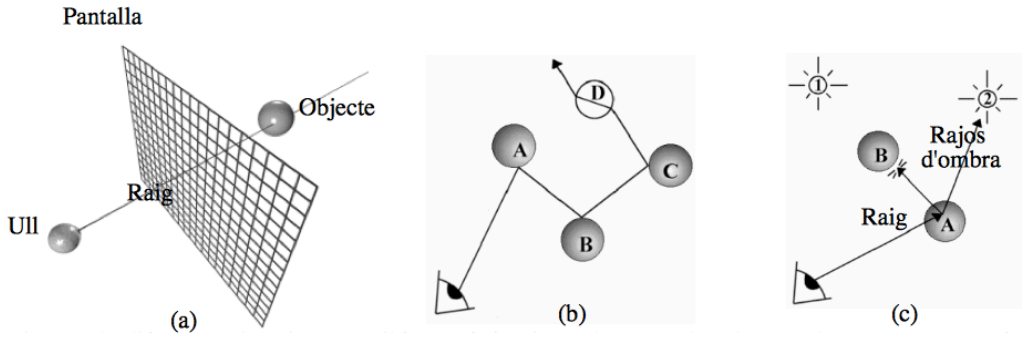


Figura 2: Figures de diferents situacions possibles en l'algorisme de traçat de rajos. La figura (a) mostra el concepte bàsic (raig **primari**), la figura (b) representa també els rajos **secundaris**, quan el raig rebota en els objectes opacs (A,B i C) i la refracció en els objectes transparents (D). La figura (c) mostra com es calculen les ombres segons RayTracing (**rajos d'ombra**), si des de la superfície de l'objecte es pot veure la llum aleshores aquesta li afecta.

Input:

Es dóna un codi bàsic per a desenvolupar el RayTracing (veure el fitxer `RayTracer.cpp`) amb les classes `Scene`, `Object` i `Ray` que cal completar per a calcular el color final de cada píxel a la imatge. Es dóna també ja implementat el codi d'OpenGL per a visualitzar la imatge final (en el mètode `Render` del fitxer `RayTracer.cpp`). El projecte es dóna amb un Makefile bàsic i amb un projecte preparat per a fer el seu desenvolupament en l'IDE QtCreator. Reviseu l'enunciat i el codi per a veure més detalls de les classes que es donen i els punts a desenvolupar.

Output:

Heu d'implementar un RayTracing per visualitzar l'escena formada per un conjunt d'objectes bàsics que inclogui com a mínim una esfera i un pla. Això implica que heu d'afegir codi per tractar les interseccions d'aquests objectes bàsics amb els rajos que escombrin la vostra escena. Heu d'utilitzar l'equació de Blinn-Phong que heu desenvolupat a la pràctica 1 per a calcular la il·luminació dels objectes. Cal que es vegin les ombres i les reflexions en la visualització final.
Addicionalment, cal que lliureu un informe on expliqueu les tècniques que heu usat donant captures de pantalla dels resultats finals que aconseguiu.

Punts obligatoris de la pràctica 2:

En la implementació de la pràctica cal que:
1. Implementeu els test d'intersecció per esferes i plans.
2. Implementeu l'algorisme de RayTracing bàsic en la funció anomenada `CastRay` del fitxer `RayTracer.cpp`. Aquesta funció cal que per a cada píxel de l'escena llenci un raig des de la posició de l'observador contra tots els objectes de l'escena.
3. Afegiu la il·luminació directe basant-se en l'equació de Blinn-Phong i afegint llums puntuals.
4. Afegiu ombres testejant la influència de les llums de l'escena llençant rajos cap als punts de llum.
5. Afegiu Reflexions calculant els rajos de reflexió a l'escena

6. Afegiu Transparències calculant els rajos transmesos en l'escena.

Etapes a seguir:

1. Instal·lació i exploració del codi

Baixeu del campus virtual el codi base del RayTracing (practica2Test). Hi han dues possibilitats per desenvolupar la pràctica: (1) la possibilitat de desenvolupar el projecte en l'IDE de QtCreator i (2) la possibilitat d'utilitzar un Makefile en el desenvolupament utilitzant qualsevol editor. El codi base és el mateix i està escrit en C++. El requeriment principal és que s'executi en la versió Linux de l'aula IA. Utilitzeu el github per a mantenir el progrés del vostre projecte.

Per executar la versió basada en Makefile, obriu una consola i executeu les comandes següents:

```
> make
> ./RayTracer
```

La visualització inicial que apareixerà en pantalla és la següent imatge:

En el cas de treballar amb la versió de QtCreator, obriu el projecte en l'entorn QTCreator i executeu el codi.

A continuació es descriuen les principals classes del projecte:

- `RayTracer.h`: fitxer que inclou les capçaleres i el codi d'utilitats. Podeu canviar el que considereu necessari.
- `RayTracer.cpp`: fitxer que conté el `main` del programa, inicialitza la finestra d'OpenGL i els mètodes per a visualitzar la imatge final. Implementa també l'algorisme bàsic principal del RayTracing explicat al principi de l'enunciat en el mètode `Render`.
- `Scene.h` i `Scene.cpp`: fitxers que contenen les dades d'una escena (similar a la classe `Mon` de la pràctica 1). Conté la informació inicial de l'escena: el vector d'objectes i les dades de la càmera. Caldrà afegir el vector de llums quan calculeu la il·luminació a cada punt. En aquesta classe es defineix el mètode `CastRay` en el què es comprova la intersecció del raig amb l'escena i en cas d'haver intersecció, calcula el color en el punt d'intersecció
- `Ray.h`: fitxer que defineix la interfície de la classe `Ray`. La classe `PayLoad` té informació de la informació del estat actual del raig. També es té la classe `IntersectInfo` que emmagatzema els detalls de la intersecció entre el raig i un objecte.
- `Object.h`: fitxer que defineix la interfície de la classe `Object` pels objectes de l'escena. Els objectes obligatoris d'aquesta pràctica són paramètrics (esferes i plans) i no cal guardar explícitament els punts de la frontera. En el cas que es faci l'ampliació de triangles caldrà guardar els tres vèrtexs que formen el triangle.
- `Object.cpp`: fitxer on s'implementen les classes `Object`, `Material`, `Sphere`, `Plane` i `Triangle`. Cal sobrecarregar el mètode `Intersect` per a cada tipus d'objecte.
- `Camera.h` i `Camera.cpp`: fitxer que conté els atributs de la càmera i el mètode `iniViewProjMatrices()` que inicialitza les matrius `viewMatrix` i `projMatrix`

En les classes `Raytracer`, `Scene`, `Object` i `Ray` implementareu la majoria dels mètodes necessaris per a fer el RayTracing.

2. Càlcul de test d'intersecció raig-esfera, raig-pla i raig-triangle

Implementeu els mètodes d'intersecció raig-esfera i raig-pla en les classes corresponents sobrecarregant el mètode `Intersect` de la classe `Object`. Utilitzeu els mètodes de càlcul d'intersecció descrits a [2], secció 10.3.

Modifiqueu la classe `Scene` per a que creï objectes quan es construeix, i canvieu el mètode `CheckIntersection` per a que els tingui en compte. En el cas de trobar varis objectes que intersecten amb el raig, es té en compte sempre la intersecció més propera a l'observador.

Finalment, per veure si les vostres interseccions funcionen correctament, en el mètode `CastRay` poseu a la variable `PayLoad` un color, com pot ser el vermell, quan es trobi que el raig intersecta amb un objecte.

Tingueu en compte que inicialment la càmera està a la posició (0,0,10) i apunta al VRP (0,0,0) amb un vector de verticalitat (0,1,0). Poseu els vostres objectes inicials per a que l'observador els vegi des d'aquesta posició.

Proveu a posar diferents objectes i comproveu que la intersecció que es veu per pantalla és la de l'objecte intersecat pel raig més proper a l'observador.

3. Càlcul del raig primari

Per a calcular el raig primari que intersecarà amb l'escena, cal tenir dos punts en el sistema de coordenades de món. El punt de l'observador en la classe `Camera` està en coordenades de món per definició. En canvi, els píxels pels que passen els rajos estan en coordenades de viewport. Cal calcular les coordenades de píxel en coordenades de món per a poder trobar la direcció de la recta que forma el raig.

Per això necessitaràs calcular les matrius `viewMatrix` i `projMatrix` que són la implementació de les matrius modelView i projection vistes a teoria. Implementa en la classe Camera el mètode `IniViewProjMatrices` ajudant-te dels mètodes `lookAt` i `perspective` implementats al fitxer `glm/gtc/matrix_transform.hpp`

Calcula el raig primari per a qualsevol posició de la càmera i per a qualsevol angle d'obertura de la càmera. El raig primari es calcula en el mètode `Render` del fitxer `RayTracer.cpp`. Canvia les dues línies marcades en el codi pel càlcul del nou raig. Com a material de reforç matemàtic, pots veure la referència [3].

Actualment, en el codi, s'està calculant la direcció del raig tenint en compte que les coordenades de càmera i les coordenades de món coincideixen, donat que l'observador està posicionat en el punt (0, 0, 10), el VRP cap a on mira és el punt (0,0,0) i el seu vector de verticalitat és el (0,1,0). Si es considera que l'angle d'obertura de la càmera és arcos (1/10.) i que l'aspect ratio del viewport és 1, es pot calcular el raig tal i com està implementat en el codi.

Com a dades d'exemple per comprovar que el càlcul del raig primari funciona correctament pots considerar `l'observador` en el punt -10, 10, 10, amb `VRP` al 0,0,0 i `VUP` a (0,1,0). considera com a angle d'obertura 45º i com a `zNear` a distància 1 i `ZFar` a 1000. Aquests valors estan posats per defecte en la constructora buida de la classe `Camera`, excepte per l'observador.

4. Càlcul d'il·luminació i ombres

Per a la intersecció del raig més propera a l'observador, s'ha de calcular la seva il·luminació basant-se amb la fórmula que heu implementat per Blinn-Phong a la pràctica 1. Quan creeu un objecte, li podeu passar el material que desitgeu. Fixeu-vos que la classe `Material` està

inclosa en el fitxer de la classe `Object`.

En aquest apartat podeu considerar només llums puntuals. Afegiu una nova classe `Light` (com feieu a la pràctica 1) i un vector de llums en la classe `Scene`. Inicialitzeu les llums en la constructora de l'escena.

Modifiqueu el càlcul de la il·luminació per incloure ombres. Per saber si un punt és a l'ombra, cal llençar un raig des del punt fins a la llum i comprovar si hi ha algun objecte de l'escena que estigui entre el punt i la llum (veure el cas (c) de la figura 2). Utilitzeu el mètode `CheckIntersection` que heu implementat en els apartats anteriors.

En el cas que hi hagi un objecte entre la llum i el punt on s'està calculant la il·luminació, quina component de la fórmula de Blinn-Phong s'haurà de tenir en compte?

5. Càlcul de reflexions

Per a calcular les reflexions cal que feu que el vostre algorisme de `CastRay` sigui recursiu. La recursivitat acaba quan s'arriba un nombre màxim de reflexions MAX_REFLECT o bé de forma adaptativa a cada raig, és a dir, quan la contribució de la nova intersecció contribueix molt poc al color final del píxel.

Per a un punt d'intersecció p_A cal calcular el raig secundari reflectit R, que es calcula seguint la fórmula

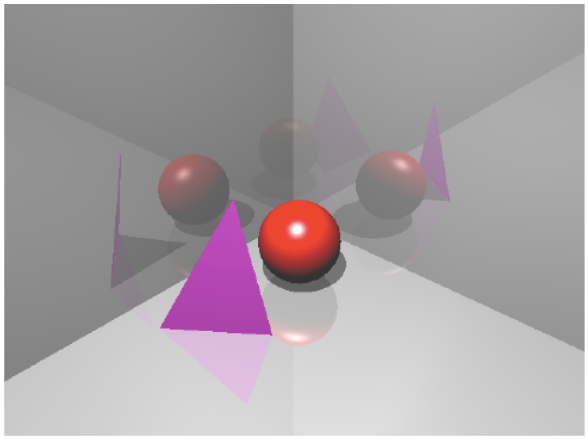
$$R = 2\,(N \cdot L) \cdot N - L$$

Aquest nou raig haurà de llençar-se contra tota l'escena de nou i veure si interseca amb altres objectes. Si interseca amb algun objecte, s'agafa la intersecció més propera al punt p_A, p_B, i es calcula la seva il·luminació segons la fórmula de Blinn-Phong, I_{pB}. Aquest color s'acumularà al color de p_A multiplicat per la ks del punt p_A.

$$I_{pA} = I_{pA} + Ks_A * I_{pB}$$

OUTPUT:

Possible imatge final de la pràctica, amb tres plans, una esfera i un triangle. El cas del triangle es fa en el bonus extra de la pràctica.



Material de referència:

[1] http://www.ics.uci.edu/~gopi/CS211B/RayTracing tutorial.pdf- Teoria bàsica de RayTracing.
[2] http://www.cs.utah.edu/~shirley/books/fcg2/rt.pdf- Capítol de llibre que explica els bàsics de RayTracing i com es deriven alguns test d'intersecció.
[3] http://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/geometry/ - Geometria bàsica per realitzar transformacions geomètriques.

[4] http://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/how-does-it-work- Síntesi de l'algorime de RayTracing.

Ampliacions (bonus) de la pràctica 2:

6. Afegir objectes de tipus triangles a l'escena, amb el càlcul d'intersecció basant-se en coordenades baricèntriques.

7. Afegir Transparències calculant els rajos de transmissió. Igual que has afegit els rajos reflectits, pots calcular a cada intersecció el raig transmès en el cas que l'objecte sigui transparent. Això implica definir un nou atribut en el materia k_t i calcular el raig de transmissió amb la llei de Snell (veure las transparències de teoria del Tema 5). Veure les referències [1] i [4].

8. Afegir d'altres tipus de llums com direccionals i spot lights. Incorpora les llums direccionals i de spot light que tenies en la pràctica 1 en el teu càlcul de la il·luminació.

9. Afegir efectes de textures als objectes. Quan es calcula la intersecció amb el raig primari, es calcula la seva il·luminació principal accedint a una textura.

10. Afegir altres tipus d'objectes a tenir en compte en el RayTracing, com Capses alineades als eixos (Axis-aligned Bounding Boxes).

També es tindran en compte com ampliacions altres propostes que penseu.

Planificació temporal de la pràctica:

Abril	25	26	27	28	29	Punts 1 i 2: Instal·lació i tests d'intersecció
Maig	2	3	4	5	6	Punt 3: Càlcul del raig primari i Punt 4: Blinn-Phong
	9	10	11	12	13	Punt 4: Ombres
	16	17	18	19	20	Punt 5: Reflexions
	23	24	25	26	27	Entrega P2