Pràctica 1
Raytracing: Fase 0

GiVD - curs 2022-23

Fase 0: Introducció al codi i primeres visualitzacions

ÍNDEX

1. INSTAL·LACIÓ del codi RayTracingToy

Baixeu del classroom github el codi base de la pràctica **RayTracingToy**.

https://classroom.github.com/a/DAylm59m

La pràctica està preparada per a desenvolupar-se en l'IDE de QtCreator. Utilitzeu el github per a mantenir el progrés del vostre projecte. **Actualitzeu el github amb commits de tot@s les estudiants de l'equip**.

Obriu el projecte en l'entorn del QtCreator i executeu el codi. Us trobareu la interfície que es mostra a la Figura 1.

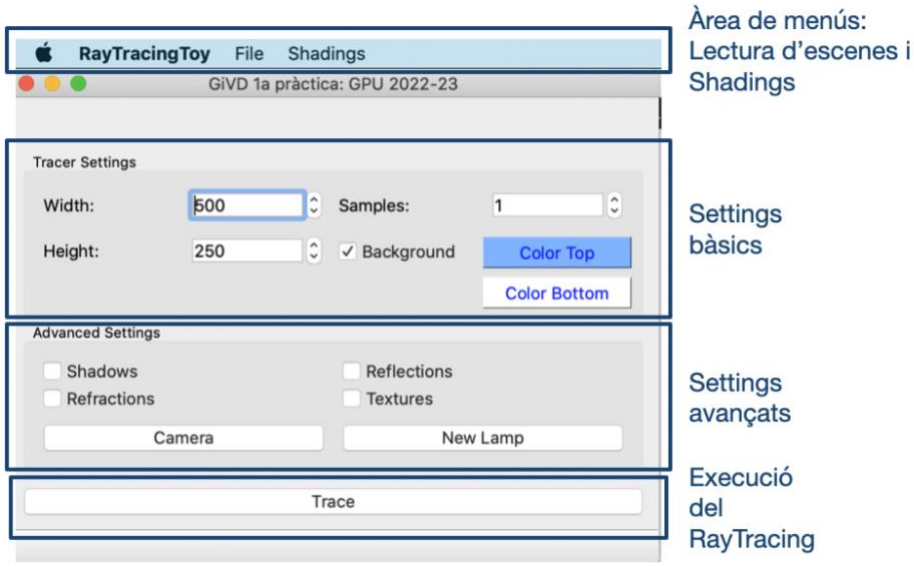


Figura 1. Finestra principal de l'aplicació RayTracingToy

Per obtenir qualsevol visualització cal prémer el botó **Trace**. La visualització inicial que apareixerà en pantalla es troba a la Figura 2.

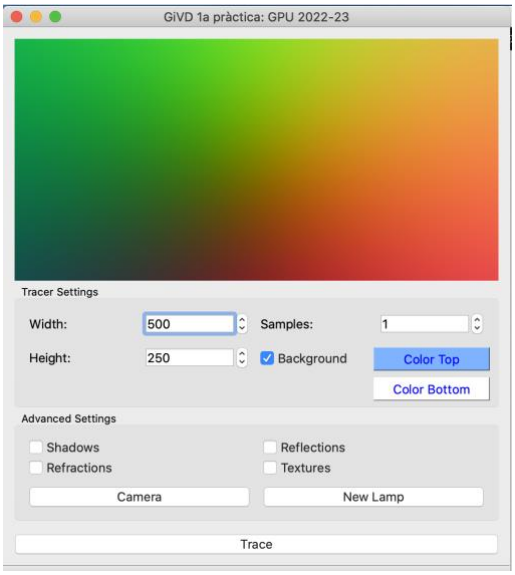


Figura 2. Primera visualització després de prémer **Trace**.

2. DESCRIPCIÓ DEL CODI

A continuació es descriuen les principals classes del projecte i podeu trobar en el campus virtual el Diagrama de Classes del projecte. L'esquema que es segueix és un Model-Vista-Controlador:

- Main: **classe** que inicia l'aplicació i crea la finestra principal MainWindow que conté tots els botons.
- A la vista es tenen les classes:
    - MainWindow: Aquesta finestra ha estat generada directament amb el Designer de QtCreator. Des d'allà, s'usen altres finestres o serveis de la interfície (contingudes a la carpeta View).
    - Builder: A la carpeta View, és la classe a partir de la qual es llegeixen les escenes o es demana al controlador de crear-ne via codi. Aquesta classe crida al Controller amb els fitxers de configuració i les dades que s'entren per la interfície. Aquests fitxers són de tipus JSON. Alguns d'ells contenen noms de fitxers, com les textures o fitxers que contenen malles de tipus .obj. Cal que aquests fitxers interns estiguin a la carpeta resources del projecte i donats d'alta als recursos (resources.qrc).

- Controller és una classe *singleton* per la que passen totes les peticions de la vista. Ofereix tots els serveis externs del model.

- El model (carpeta Model) està dividit en dues carpetes:
    - la carpeta Modelling, que inclou tots els elements de l'escena virtual i les seves factories corresponents. Dins d'aquesta carpeta trobaràs d'altres carpetes i classes, de les quals destaquem:
        - Scene: classe que conté tots els objectes d'una escena. En aquesta classe es defineix el mètode *hit* en el què es comprova la intersecció del raig amb l'escena més propera a l'observador i en cas d'haver intersecció, calcula la informació necessària (veure classe HitInfo) per calcular la il·luminació en el punt d'intersecció.
        - Object: classe que defineix la interfície de la classe Object pels objectes de l'escena. Aquesta classe hereta de la interfície Hitable (classe abstracte que obliga a la implementació del mètode *hit*) En la versió inicial, es proporcionen les classes Sphere, Plane i Mesh que són de tipus Object. També hereta de la interfície Serializable per poder llegir-se de disc i de la interfície Animable per

poder-se moure mitjançant animacions predefinides.

- La carpeta Rendering, que inclou totes les classes relacionades amb la visualització directament, des de l'algorisme de *raytracing* (classe RayTracer), la seva configuració (classe SetUp) i les estratègies per calcular diferents il·luminacions (o *shadings*). Com a classes a destacar:
  - RayTracer: Conté l'escena i el setup de visualització així com l'algorisme de raytracing (mireu el mètode run)
  - SetUp: Classe que conté tota la configuració per fer una visualització: càmera, llums, *flags* com el de *background*, d'ombres, etc.
  - Ray: La classe Ray conté la definició geomètrica d'un raig. La classe HitInfo té informació de la informació del estat actual del raig quan interseca amb algun objecte.
  - Camera: classe que conté totes les utilitats per a definir una càmera. En aquesta pràctica **NO** hauràs de modificar aquesta classe.

- La part de persistència de sortida i las propietats que ajudaran a realitzar el visual *mapping* es codifiquen a les classes de la carpeta DataInOut.

- Interfícies del projecte: Hitable, Serializable i Animable. Tota classe que necessiti calcular la intersecció amb un raig hereta de la interfície Hitable. Tota classe que ha de ser llegida de fitxer o part d'un JSON hereta de la interfície Serializable. Tota classe que es vulgui animar en les extensions, caldrà que hereti de la interfície Animable. Això permetrà que el *rendering* temporal pugui cridar al mètode update() per poder-les animar *frame* rere *frame*.

3. ACTIVITATS

a. Obre la classe MainWindow i explora com es connecten els menús i els events de la interfície amb els serveis de l'aplicació mitjançants els *signals* i els *slots* de Qt. Fixa't també que hi ha events que directament venen de la interfície com els anomenats on_*_valueChanged.

b. Explora el codi fent una primera execució usant el *debugger* (o navegant pels mètodes) i veient com des del botó de Trace, s'arriba al mètode run() de la classe RayTracing. Quina escena es té carregada? Qui la crea? Quin setup té la classe? Des d'on es crea?

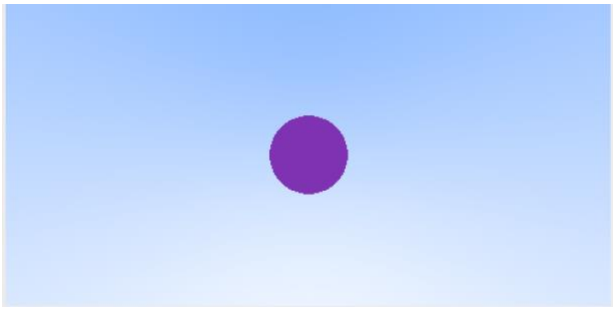
c. Si prems els botó de Trace, es veu el degradat de la figura 2. Per què es veu aquest degradat? A quin mètode es calcula aquest color?

d. Si desactives el *flag* de *background*, i tornes a fer Trace, quin efecte et trobes? Per què? Pots veure com ha arribat el *flag* de *background* al teu RayTracer?

e. Amb el *flag* de *background* activat, intenta fer que el fons es canviï segons es mostra a la figura 3, tenint en compte que es vol un degradat segons les Y's de forma que es vol pintar de color més blanc quan estigui més avall de les Y's de pantalla (RGB = (1.0, 1.0, 1.0)) i més blavós quan estiguis més amunt (per exemple, RGB = (0.5, 0.7, 1.0)). Fes-ho primer directament en el codi i després intenta veure com llegir els colors que pots posar a la interfície. En quina variable els trobes?

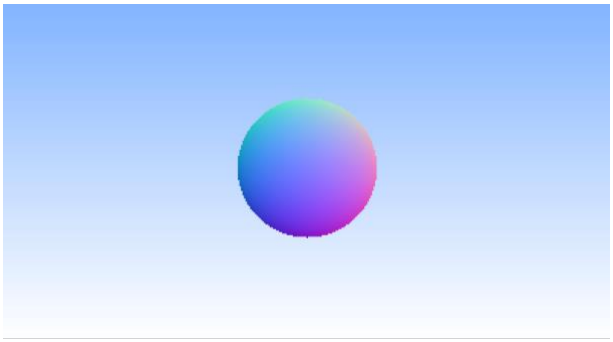
f. I si volguessis canviar els colors del degradat? On els canviaries?

g. Des del menú de File pots crea una esfera unitària amb centre al punt (0, 0, 0). Si prems **Trace**, no la veuràs, per què? Modifica el codi per a que es tingui en compte la intersecció amb l'esfera i que es pinti de color lila (es l'atribut Kd del material contingut a HitInfo). Fixa't amb el codi del mètode hit de la classe Scene i el codi de RayPixel de la classe RayTracer. Gradua la càmera per veure l'esfera amb més zoom i et cal. Intenta resseguir des del MainWindow, passant pel Controller i veient que s'està creant i a on.

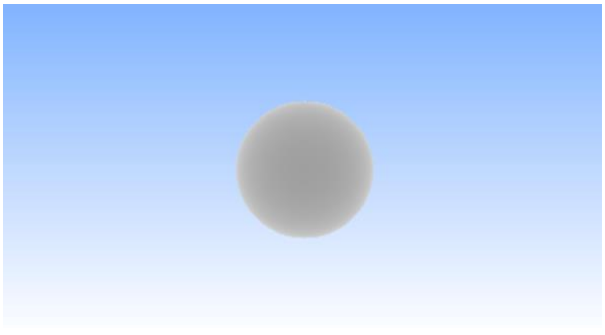


h. Canvia ara per a que el color es calculi des del ColorShading, es a dir, enlloc de pintar l'esfera amb el color del seu material Kd, es vol usar el càlcul fet a la classe ColorShading. Fixa't que des del **Menu Shadings** pots activar aquesta estratègia però cal que la usis en calcular el color de sortida de cada píxel. Com hi pots accedir? A quina variable pots aconseguir l'estratègia a cridar? Com es crea aquesta estratègia des del menú? Resegueix el codi des del MainWindow.

i. Canvia per a que ara per a que l'esfera es vegi segons els colors de les normals calculades en els punts d'intersecció, i així veuràs aquesta figura. Com faràs per a crear una nova estratègia de shading?

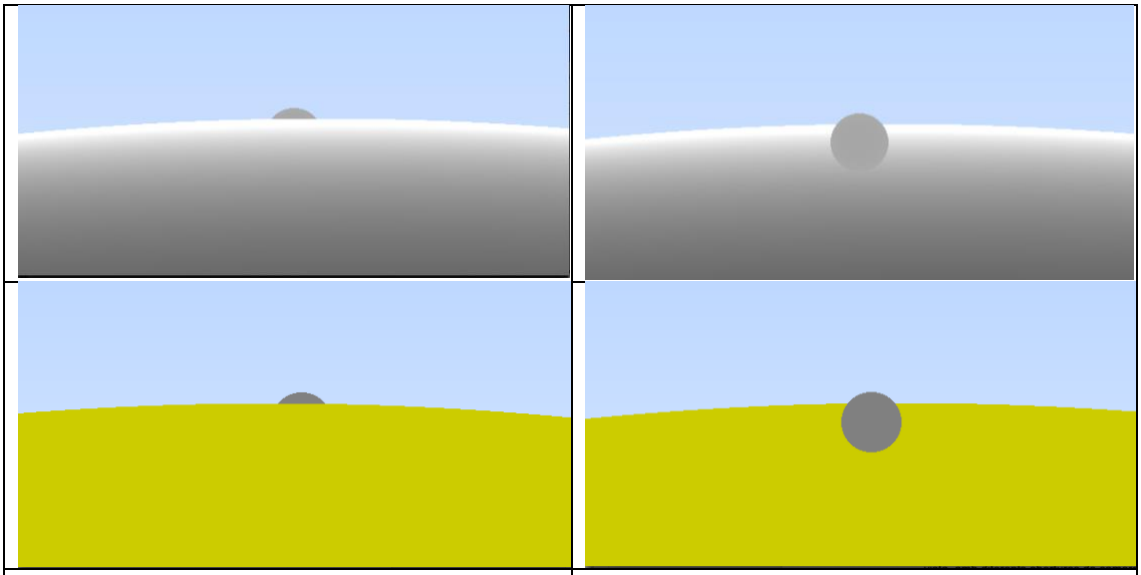


j. Canvia ara per a que l'esfera es vegi segons una nova estratègia, usant el valor de la seva distància a l'observador. Com aconsegueixes que els colors estiguin normalitzats? Fixa't on tens situada la càmera i on està situada l'esfera a la teva escena. Pots provar diferents gradacions.



k. Canvia els paràmetres inicials de la càmera per visualitzar des de diferents punts de vista, amb diferents obertures de càmera.

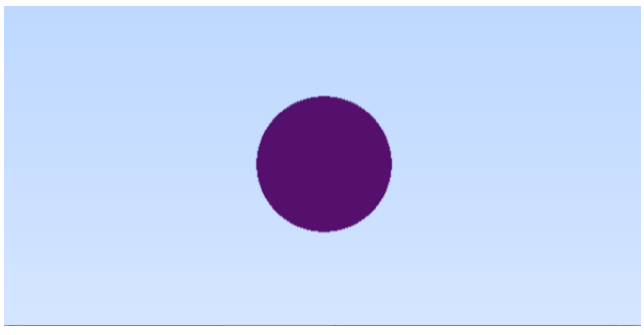
l. Carrega ara el fitxer de dues esferes que tens en la carpeta dels recursos resources del projecte twoSpheres.json amb el fitxer de configuració setupOneSphere.json. Observa que el material difús es carrega des del mateix fitxer de dades (són els valors kd del material associat a cada esfera). Codifica el mètode hit de la classe Scene per a que es facin les interseccions amb totes les esferes i es calculin bé les seves visibilitats. Juga amb les posicions de la càmera i amb les distàncies per normalitzar la "depth" per a poder obtenir imatges semblants a les de les figures.



Càlcul incorrecte	Càlcul correcte

m. Obre un vim o un editor senzill de text i intenta generar un fitxer .json amb més esferes col·locades a diferents punts de la teva escena virtual per testejar el teu mètode de hit.

n. Prova a carregar ara el fitxer de dades reals data0.json que conté en l'array de data amb un punt situat a -2, -1 del món real i mostreja el valor de 0.5. Manté el mateix fitxer de setup de la visualització que has usat en el punt anterior. Per què veus només una esfera? On està situada a la teva escena? Amb quin radi? Per què? Per què és lila i no de color "kd": [0.7,0.6,0.5] com posa el fitxer? Resegueix amb el debugger la creació de l'escena en aquest cas on el fitxer conté dades.



Temps estimat: 3:00 - 4:00 hores