Pràctica 1
Raytracing: Fase 3

GiVD - curs 2022-23

Fase 3: Aplicació de textures en el pla i ampliacions.

Índex

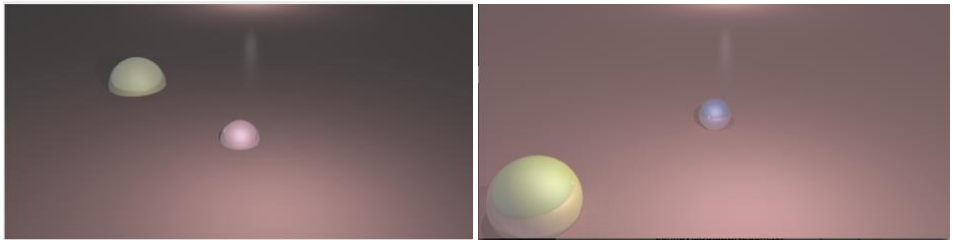
1. INTRODUCCIÓ

En aquesta fase s'inclouran les textures a les visualitzacions. Per a simplificar, només es mapejaran textures en el pla base de les dades geolocalitzades i opcionalment podràs també mapejar-les en els objectes de tipus BOX.

2. PASSOS A SEGUIR

PAS 1. Afegeix un nou material de textura al pla base

1. Per a començar bé: Comprova la teva pràctica seguint aquesta escena basada en els fitxers de dades relacionats amb Europa, dadesEuropaFase3.json, que trobaràs al campus virtual. S'ha configurat la escena per a que les esferes siguin metàl·liques amb els següents valors de propietats Ks = (0.7, 0.7, 0.7), beta = 10.0 i Ka = (0.2, 0.2, 0.2). S'ha calculat l'escena amb MAX_DEPTH = 10. Amb el fitxer de setup setupDataEuropa.json quina de les dues imatges obtens? Raona per què.



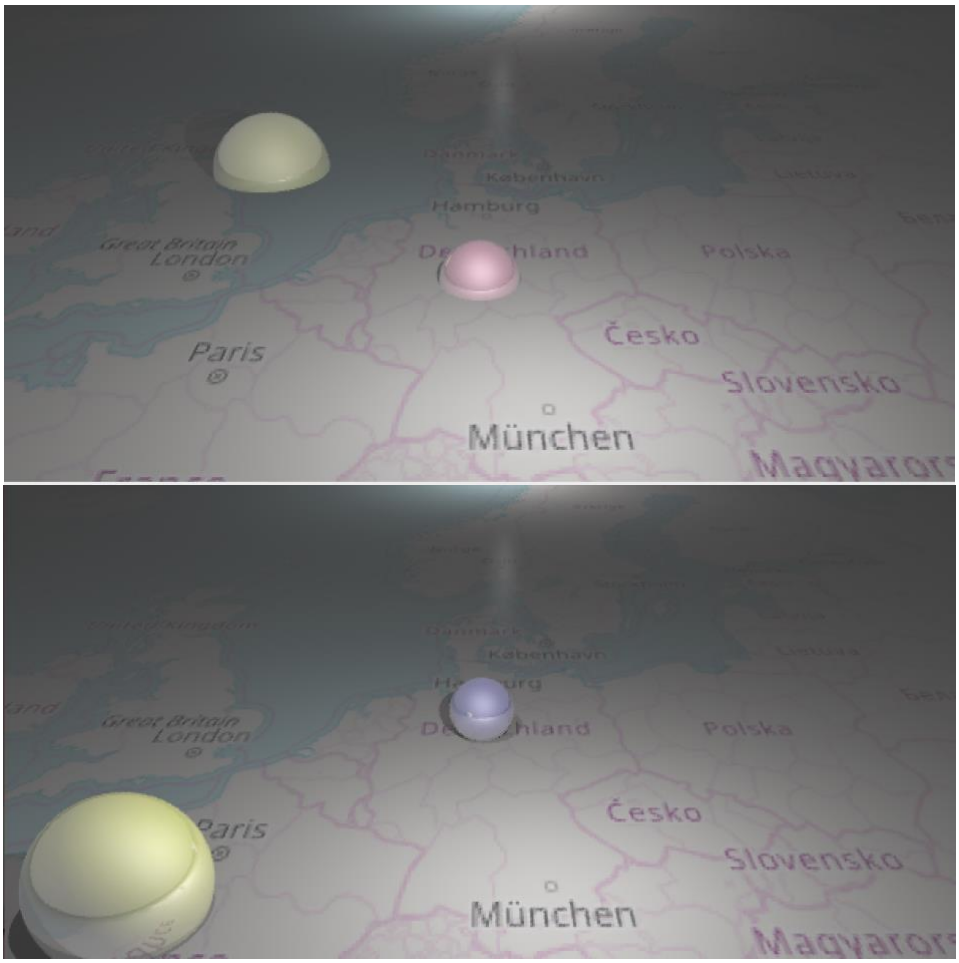
2. Afegeix un nou Material anomenat MaterialTextura que permeti tenir guardada una textura i que no calculi cap raig secundari en el mètode scatter(). Afegeix el mètode read() a la classe MaterialTextura per a poder carregar el material corresponent i, a més a més, poder llegir el fitxer de textura i assignar-lo a un atribut de la classe MaterialTextura. Aquest mètode es crida de forma automàtica des del codi del mètode read de la classe Object.

   void MaterialTextura::***read*** (const QJsonObject &json)

   Fixa't que el mètode getDiffuse a la classe Material que permet passar-li el paràmetre *uv* (és a dir, les coordenades de textura) per accedir als píxels de textura. Tots els objectes fan servir ara aquest mètode, tinguin textura o no. Així, quan des de Blinn-Phong es vulgui accedir al material difús, es crida aquest mètode que retornarà el color difús si no es té textura i el color del píxel de la textura en cas que es tracti d'un material amb textura. Qui calcularà les coordenades (u, v) del punt d'intersecció amb el pla?

   vec3 Material::***getDiffuse***(vec2 point) const

3. En el fitxer dadesEuropaTextured.json s'inclu un mapa d'europa en format png. Per a que es pugui llegir correctament, cal que l'incloguis des del projecte a la part de resources.qrc, tal i com feies amb els .obj en el pas 3 de la Fase 1. Si a la mateixa escena d'abans es considera aquest nou material en el terra, usant el setup de setupDataEuropa.json, quina de les dues visualitzacions obtens? Fes captura en el teu README de la visualització que obtinguis.



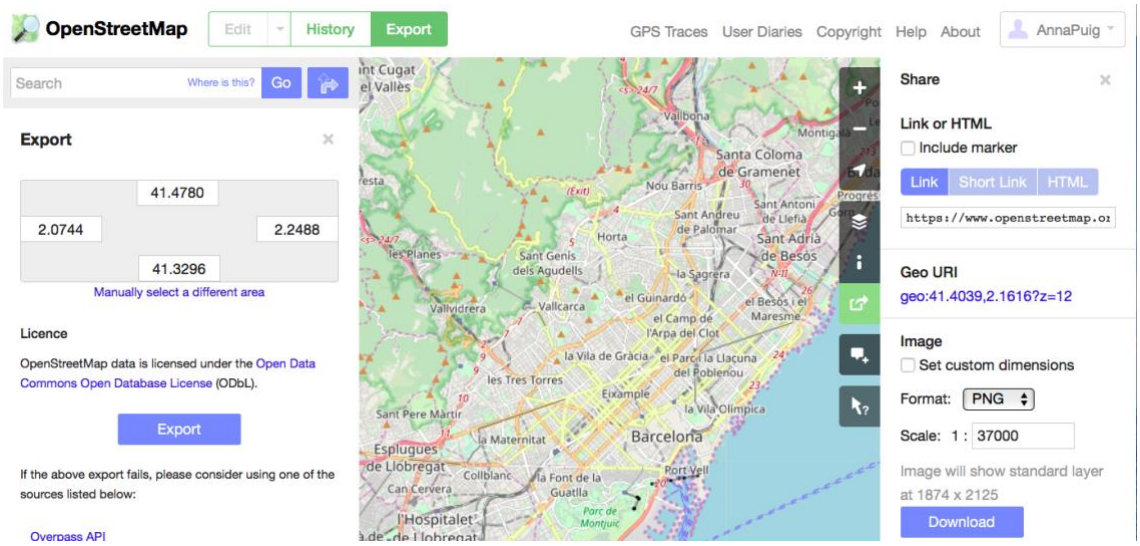
PAS 2: Prova amb diferents jocs de dades i mapes

En el fitxer dadesBCN_Zoom.json es té associada la textura mapZoom.png, i el fitxer dadesBCN.json té associada la textura també de mapBCN.png. Pots obtenir visualitzacions com les següents usant els fitxers de configuració setupDataBCN_Zoom.json i setupDataBCN.json, respectivament:

Prova també a posar les esferes metàl·liques i transparents, com feies a la Fase 2.

PAS 3: Genera diferents jocs de dades i mapes

En l'enllaç [1] pots extreure mapes que tenen les coordenades mínimes i màximes de latitud i longitud. Només cal que seleccionis l'àrea de la que vols el mapa i amb el botó de *share* pots obtenir el .png o directament fent la captura de pantalla. El que és interessant aquí es el poder tenir els límits en x i en z del món real del mapa.



Posa en el teu fitxer de dades json els límits geogràfics de la teva imatge com a valors de “Real”:

```
"Real": {
  "minPoint": [2.0744, 41.3296],
  "maxPoint": [2.2488, 41.4780]
},
```

Ara ja pots inserir dades geolocalitzades en aquesta àrea en el teu fitxer de dades. Un bon lloc per trobar dades referents a Barcelona el pots trobar a [2].

Cal incloure com a mínim un nou joc de dades amb un mínim d'una propietat, amb un mínim de 15 dades i amb algun altre mapa.

PAS 4: Genera noves escenes i resultats finals de la pràctica

Genera noves escenes amb tot el que has generat en la pràctica. Detalla els .json de les escenes virtuals o de dades i els seus fitxers de configuració (o setups) corresponents. Penja'ls al padlet: https://padlet.com/anna3143/xafajnzngebonnt1 per a que d'altres equips puguin reproduir-los.

RESUM D'EXTENSIONS OPCIONALS

Aquí es detallen totes les extensions opcionals que han anat sorgint en les altres fases de la pràctica i se n'inclouen algunes més.

Les parts opcionals poden ajudar a pujar la nota final de la pràctica 1. Cadascuna, segons les hores de dedicació que comporten, tenen una certa qualificació o ponderació dels punts obtinguts. En el següent llistat les trobareu ordenades de més fàcils a més difícils.

- Es poden afegir **cilindres** com a nous objectes.
- Es poden afegir altres objectes que representin altres tipus **d'objectes paramètrics** o **funcions implícites** i que es visualitzin segons alguna característica de la superfície que representen. Mira la referència [3] per obtenir idees.
- Llegir dades geolocalitzades amb **més d'una propietat** usant diferents *gizmos*.
- **Mapeig de les dades en una esfera** (i no en un pla com a terra)
- **Ús d'arbres CSG** per estructurar l'escena i poder fer operacions booleanes entre objectes.
- **Diferents tipus de llums.** Ara que ja tens les fots de llum puntuals, pots ampliar el ventall de llums amb llums direccionals i llums de tipus spot-light. Afegeix també la GUI per a modificar-les
- **Llums amb àrea i penombres**. Es vol produir l'efecte de penombra i per això pots afegir llums amb àrea com un conjunt de punts de llum. Com faries el càlcul de les penombres? (mira la transparència 9 del tema 2c).
- **Ombres segons el color de l'objecte transparent**. Pots tenyir les ombres segons l'objecte transparent pel què passa la llum. Prova a fer efectes amb aquesta opció.
- **Ambient Occlusion:** Generar les ombres que produeixen els mateixos objectes sobre la llum ambient
- **Textures en objectes de tipus BOX.** Utilitza els mateixos conceptes que has treballat a la Fase 3 per a mapejar les textures en una BOX
- **Textures en objectes de tipus Sphere.** Utilitza els mateixos conceptes que has treballat a la Fase 3 per a mapejar les textures en una esfera
- **Textures per a tenyir les llums.** Utilitza les transparències en objectes transparents per posar un pla amb una textura que segons el color, "tenyeixi" la llum que passa pel pla.

- **Animacions de dades temporals.**

Per a activar les animacions al projecte, has d'activar des del fitxer .json l'etiqueta TEMPORALVW. Aquesta opció no está implementada. En el codi base trobaràs que a la part dels Renders, es contempla fer visualitzacions temporals si s'activa l'opció RENDER_TYPES::TEMPORAL.

Fixa't en la classe MainWindow, el mètode runanimation() detalla com es crida repetidament al Raytracing segons un número de Frames. Cal que la teva escena els contingui. Fixa't les classes Animation per a veure com funcionen les animacions.

Quan es crea l'escena cal que els objectes que es volen animar tinguin animacions. En el següent exemple, s'afegeix l'animació a l'esfera per a traslladar-la (0.2, 0.2, 0.2) a cada frame. Teniu aquest codi a la classe Controller (línia 46):

```
46  bool Controller::createScene(int nFrames) {
47
48      //TO DO Fase 3 opcional: Codi exemple amb animacions però que es pot canviar
49      // pel que creguis convenient
50
51      auto sphere = make_shared<Sphere>(vec3(0, 0, -1), 0.5, 1.0);
52      sphere->setMaterial(make_shared<Lambertian>(vec3(0.5, 0.2, 0.7)));
53
54      shared_ptr<Animation> anim = make_shared<Animation>();
55      anim->transf =  make_shared<TranslateTG>(vec3(0.2));
56      sphere->addAnimation(anim);
57
58      return true;
59  }
```

Fixeu-vos que la classe Animation conté el frame inicial on es vol activar l'animació (frameIni) i el frame final fins al que es vol l'animació activa (frameFinal). El nombre màxim de frames es defineix a la constant MAXFRAMES.

```
8   #include <memory>
9
10  using namespace std;
11
12  class Animation
13  {
14  public:
15   int frameIni;
16   int frameFinal;
17   shared_ptr<TG> transf;
18
19   Animation(): frameIni(0), frameFinal(10), transf(NULL) {}
20
21   //  "operator =" per la classe  Animation
22   Animation &operator =(const Animation &rhs) {
23     frameIni = rhs.frameIni;
24     frameFinal = rhs.frameFinal;
25     transf = rhs.transf;
26     return *this;
27   }
28  };
```

Tot objecte es pot animar i hereta de la classe Animable. Aquesta classe conté un vector d'animacions que defineix totes les animacions que pot tenir un objecte. El codi base pressuposa que tot objecte tindrà definida una animació i el mètode update() definit (fixa't en el mètode update de la classe Animable).

Afegeix la lectura de les transformacions geomètriques a la classe TG i les seves derivades per poder llegir les transformacions geomètriques des de fitxer.

Ara, per a afegir sèries temporals als teus objectes, has de modificar el fitxer de l'escena .json per a que indiqui que es llegirà un fitxer amb animacions i el número d'instants de temps a llegir:

"numInstants": 10

Conseqüentment, hauràs de modificar la classe SceneFactoryVirtual per a guardar aquests valors per poder llegir fitxers de dades amb animacions.

Modifica la classe SceneFactoryVirtual per a poder llegir les animacions o transformacions associades a un objecte virtual. Després de la línia que defineix un objecte, hauràs d'afegir tantes línies com instants de temps vulguis animar l'objecte. Per exemple, si tens una esfera i la vols animar amb una translació, un escalat i una rotació, definiràs les línies:

```
"translation": [ -1, -2, 3],          // x, y, z a traslladar
"scale": [2.0],                       // escala uniforme en els tres eixos
"rotation": [30, 1, 0, 0]             // rotació de 30 graus al voltant de l'eix x (1, 0, 0)
```

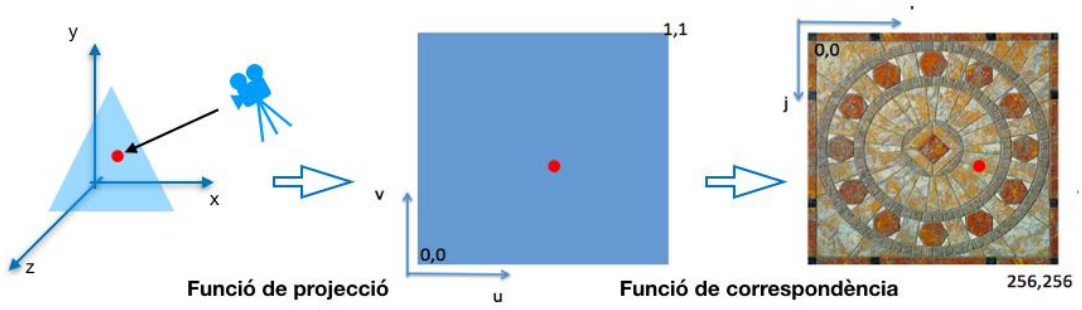
Per una altra banda pots modificar el fitxer de dades reals per afegir més dades agafades en diferents instants de temps i la modifica la seva lectura a la classe SceneFactoryData.

Recorda de pujar les teves millors visualitzacions finals al padlet https://padlet.com/anna3143/xafajnzngebonnt1 i així veurem tots els efectes aconseguits. Votarem els millors!

APÈNDIX: COM APLICAR UNA TEXTURA A UN PLA AFITAT?

Fins ara, quan es defineix el color difús d'un objecte, o es té el material directament per què ve d'una escena virtual o bé prové d'un material creat a partir del color de la paleta corresponent al valor que es defineix en els fitxers de dades. El color difús, doncs, és constant en tots els punts de l'objecte.

Quan s'utilitzen textures per a definir el material de l'objecte, es considera que cada punt de l'objecte es correspon a un píxel de textura (o téxel). Així, quan es troba una intersecció del raig amb l'objecte, a partir del punt d'intersecció es calculen les coordenades de textura (definides en un espai 2D entre el (0,0) i el (1,1)). Aquest pas és la **funció de projecció**. Aquestes noves coordenades de píxel entre 0 i 1, s'anomenen **coordenades de textura (u, v)**. A partir de les coordenades de textura es troben el píxel de la imatge corresponent usant la funció de correspondència.



Tot i que hi han diferents funcions de projeccions, a la pràctica obligatòria usarem només la **funció de projecció** (o funció de mapeig) en un pla. Es definirà la textura sobre el pla del terra de l'escena (un pla acotat per un paral·lelepípede, amb dos punts donats sobre el pla, pmin i pmax). La textura vindrà donada en el fitxer de dades com a material de l'objecte base dadesEuropa.json

```
"scene": "dadesEuropa",
"typeScene": "REALDATA",
"base" : {
   "type": "FITTEDPLANE",
   "normal": [0.0, 1.0, 0.0],
   "point": [0.0, -1.0, 0.0],
   "pmin": [-10.0, -10.0],
   "pmax": [2.0, 2.0],
   "material": {
       "type": "MaterialTextura",
       "ka": [0.2, 0.2, 0.2],
       "kd": [0.7,0.6,0.5],
       "ks": [0.7, 0.7, 0.7],
       "shininess": 10.0,
       "textureFile": ":/resources/map.png"
    }
```

Per a aquesta part, es disposa d'una classe anomenada Texture a la pràctica base. Aquesta classe facilita la lectura de la imatge de la textura (el fitxer png o jpg) i també l'accés al color del píxel de la textura a partir de les coordenades de textura, és a dir, es dóna ja implementada la **funció de correspondència** però no és de tipus Material. Cal crear un nou material MaterialTextura que usi aquesta classe.

Nota que el MaterialTexture NO llegeix automàticament quan es carrega un material des del .json. El codi base només et deixa llegir les propietats del material però no la textura. Hauràs d'afegir el codi a la classe MaterialTexture per a que pugui llegir el camp "textureFile" del fitxer .json i poder delegar a la classe Texture, la lectura del fitxer .png.

Per una altra banda, serà necessari que, en el moment de trobar la intersecció amb el pla acotat del terra de l'escena, calculis les coordenades de textura (u, v) associades al punt d'intersecció (la funció de projecció). Fixa't que a la classe hitInfo tens un atribut que guarda el valor de (u, v) associat a la intersecció, per poder-se utilitzar quan es calculi el color d'aquest punt intersecat amb la fórmula de Blinn-Phong. Recorda que el color de la textura, en aquest cas, serà el color difús del pla.

Temps estimat de cada estudiant

10 hores de treball setmanal fora de l'aula

Material addicional

[1] https://wiki.openstreetmap.org/wiki/Using_OpenStreetMap Com obtenir diferents mapes: https://www.openstreetmap.org/export#map=6/40.002/1.362

[2] https://opendata-ajuntament.barcelona.cat/data/es/dataset Dades geolocalitzades de Barcelona en obert.

[3] https://www.iquilezles.org/www/articles/intersectors/intersectors.htm Idees d'interseccions amb diferents objectes.