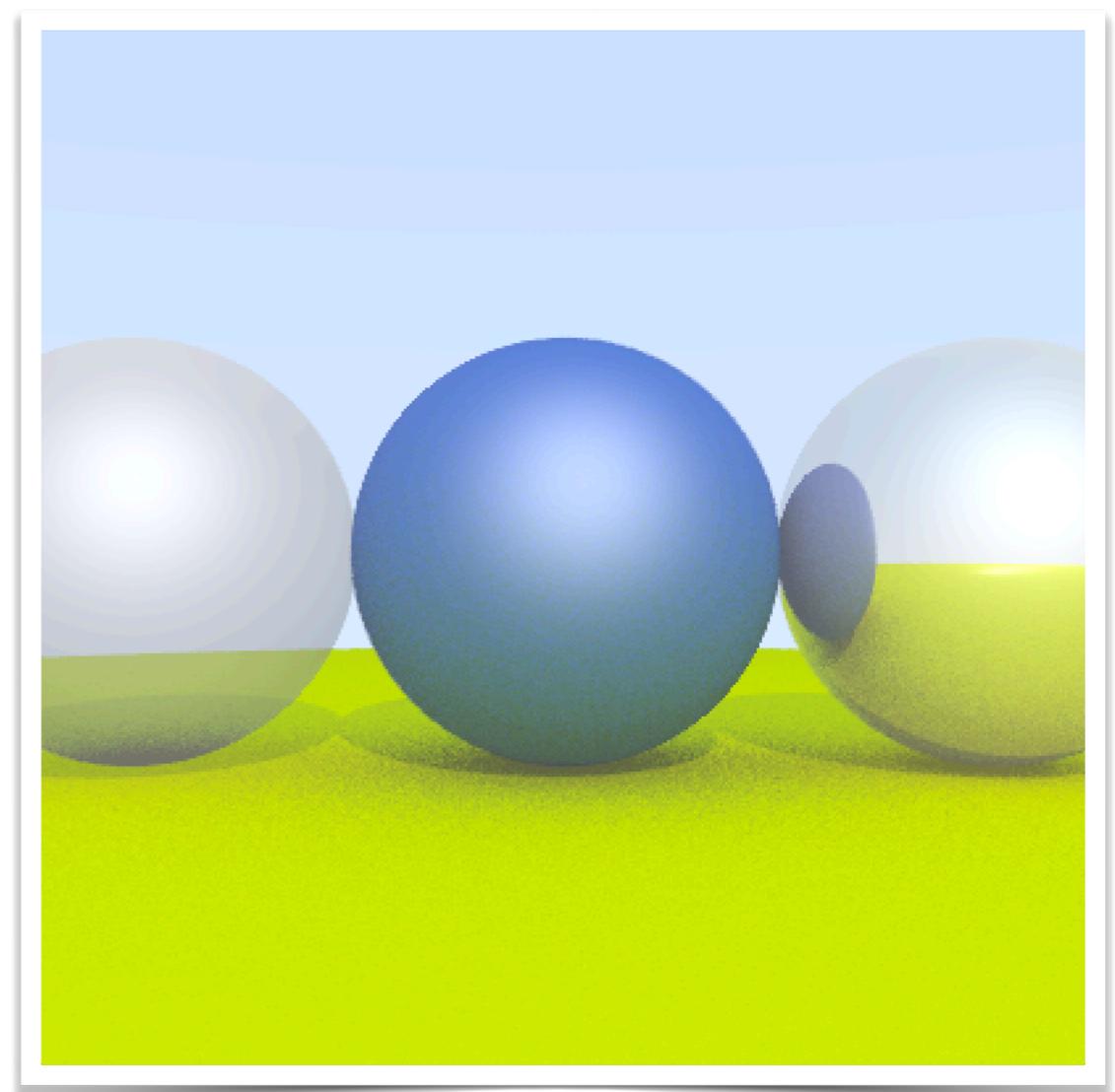


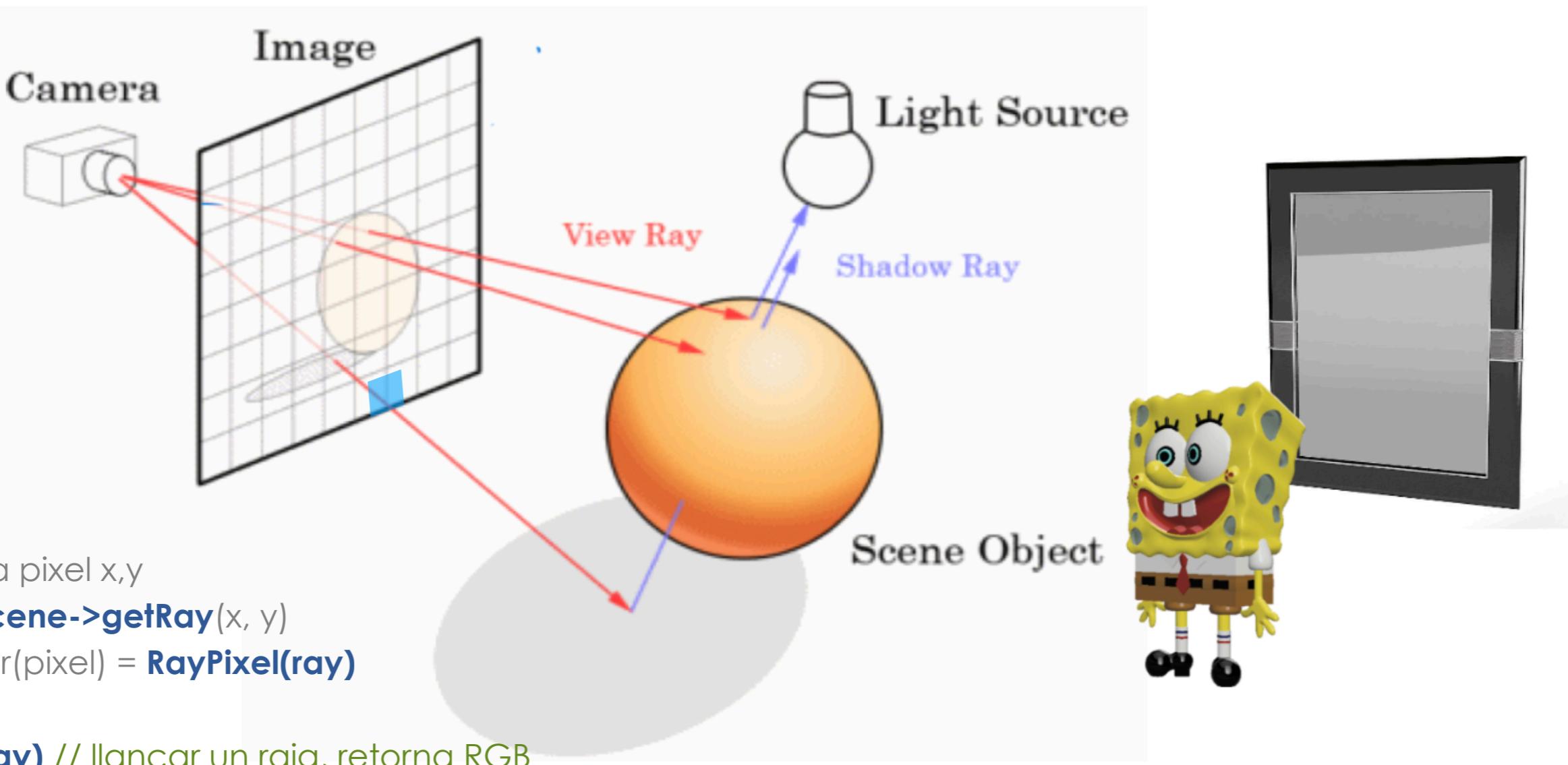
Sessió Setmana 4

GiVD 2022-23

Guió de la sessió

1. Planificació de la setmana 4
 2. *Visual Mapping a la pràctica*
 3. Representacions de malles:
repàs, activitat
 4. Exercicis extra
-

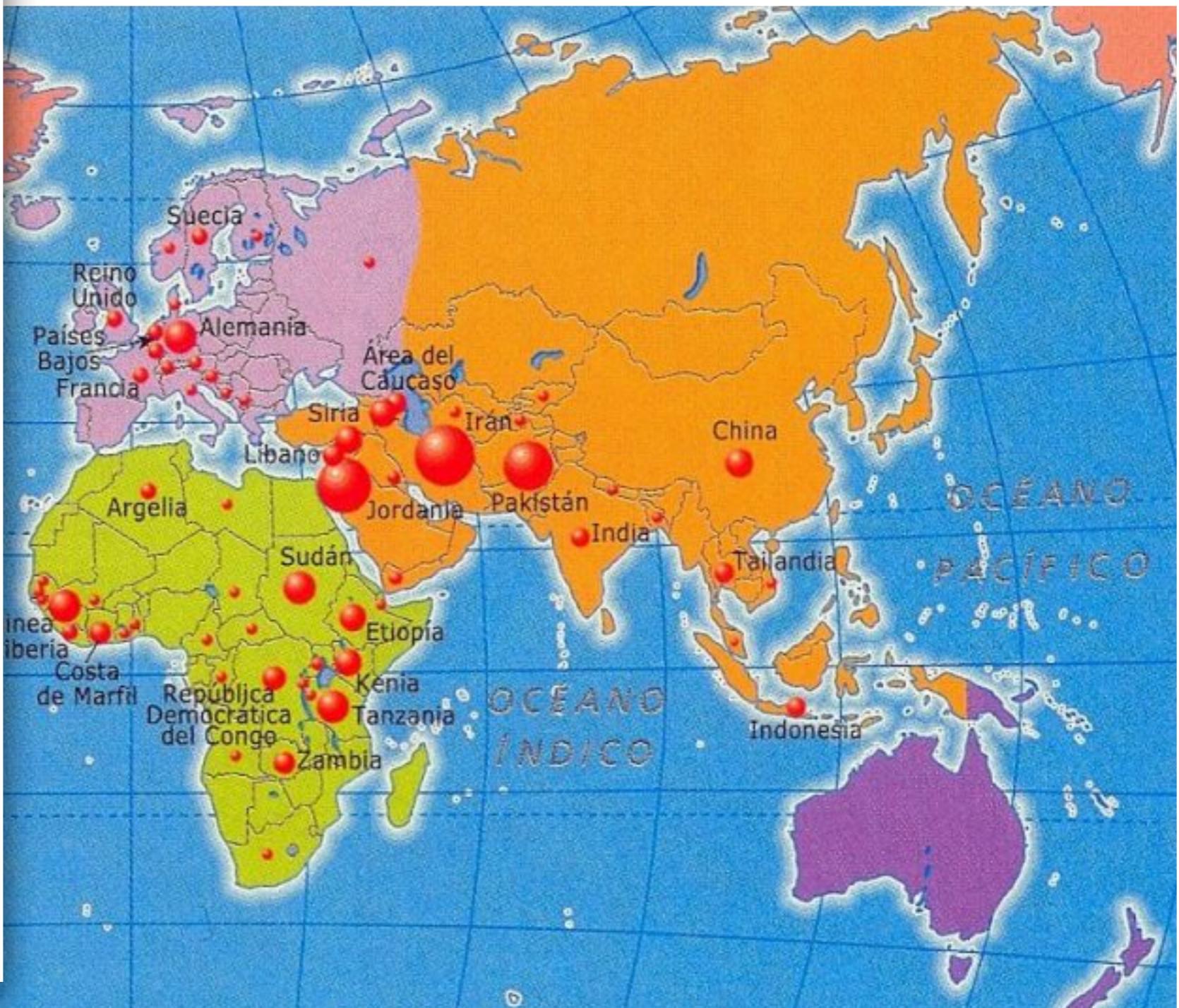




- **run()**
 - per cada pixel x, y
 - $r = \text{scene-} \rightarrow \text{getRay}(x, y)$
 - $\text{color(pixel)} = \text{RayPixel(ray)}$
- **RayPixel(ray)** // llançar un raig, retorna RGB
 - $\text{hitInfo} = \text{hit(ray)}$ // hit(ray, tmin, tmax, hitinfo)
 - si object_point retorna **Shade(hitInfo, ray)**
 - sino retorna Background_Color // o Intensitat ambient global
- **hit(ray)**
 - per cada objecte en l'escena
 - hit(ray, surface)** // hit(ray, tmin, tmax, hitinfo)
 - retorna el punt més proper a l'observador (+normal, +material)
- **Shade(hitInfo, ray)** // retorna la il·luminació en el point (strategy)
 - retorna color

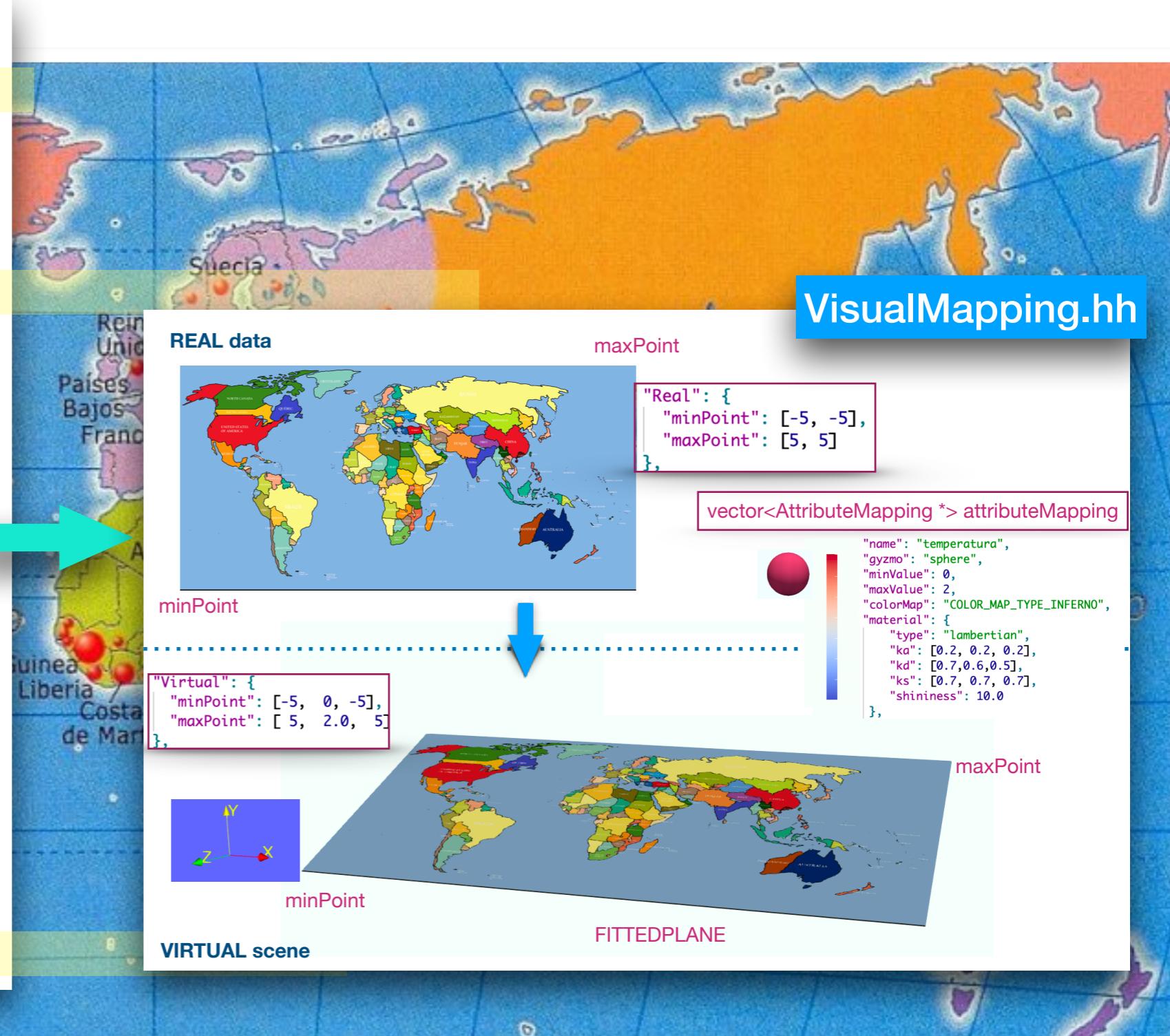
2. Visual Mapping

```
{  
  "scene": "Data0",  
  "typeScene": "REALDATA",  
  "Real": {  
    "minPoint": [-5, -5],  
    "maxPoint": [5, 5]  
  },  
  "Virtual": {  
    "minPoint": [-5, 0, -5],  
    "maxPoint": [5, 2.0, 5]  
  },  
  "attributes": [  
    {  
      "name": "temperatura",  
      "gizmo": "sphere",  
      "minValue": 0,  
      "maxValue": 2,  
      "colorMap": "COLOR_MAP_TYPE_INFERNO",  
      "material": {  
        "type": "lambertian",  
        "ka": [0.2, 0.2, 0.2],  
        "kd": [0.7, 0.6, 0.5],  
        "ks": [0.7, 0.7, 0.7],  
        "shininess": 10.0  
      },  
      "data": [  
        [-2, -1, 0.5]  
      ]  
    }]  
}
```



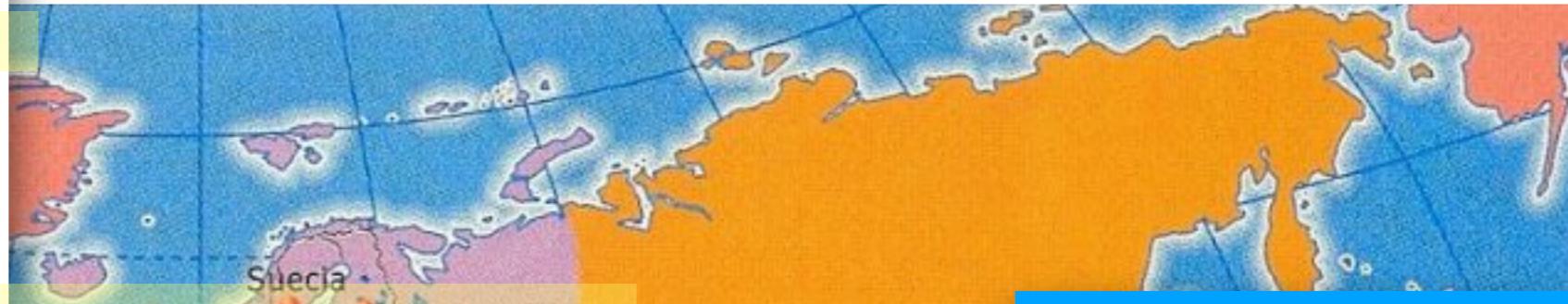
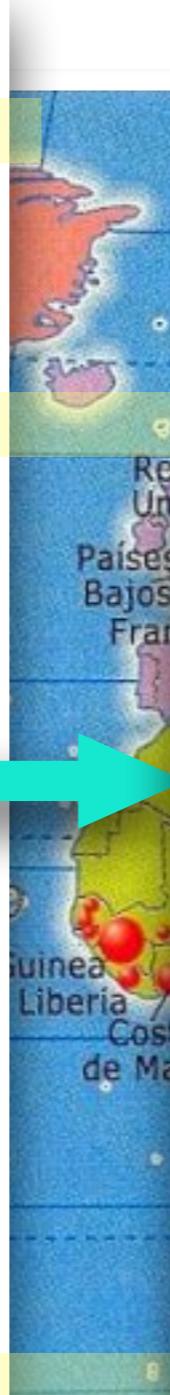
2. Visual Mapping

```
{
  "scene": "Data0",
  "typeScene": "REALDATA",
  "Real": {
    "minPoint": [-5, -5],
    "maxPoint": [5, 5]
  },
  "Virtual": {
    "minPoint": [-5, 0, -5],
    "maxPoint": [5, 2.0, 5]
  },
  "attributes": [
    {
      "name": "temperatura",
      "gyzmo": "sphere",
      "minValue": 0,
      "maxValue": 2,
      "colorMap": "COLOR_MAP_TYPE_INFERNO",
      "material": {
        "type": "lambertian",
        "ka": [0.2, 0.2, 0.2],
        "kd": [0.7, 0.6, 0.5],
        "ks": [0.7, 0.7, 0.7],
        "shininess": 10.0
      },
      "data": [
        [-2, -1, 0.5]
      ]
    }
  ]
}
```

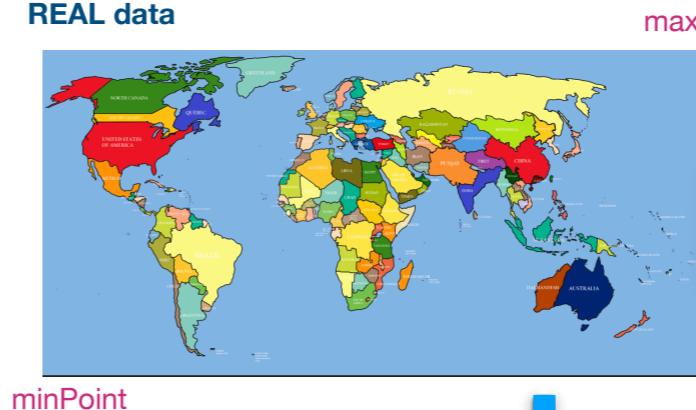


2. Visual Mapping

```
{
  "scene": "Data0",
  "typeScene": "REALDATA",
  "Real": {
    "minPoint": [-5, -5],
    "maxPoint": [5, 5]
  },
  "Virtual": {
    "minPoint": [-5, 0, -5],
    "maxPoint": [5, 2.0, 5]
  },
  "attributes": [
    {
      "name": "temperatura",
      "gizmo": "sphere",
      "minValue": 0,
      "maxValue": 2,
      "colorMap": "COLOR_MAP_TYPE_INFERNO",
      "material": {
        "type": "lambertian",
        "ka": [0.2, 0.2, 0.2],
        "kd": [0.7, 0.6, 0.5],
        "ks": [0.7, 0.7, 0.7],
        "shininess": 10.0
      },
      "data": [
        [-2, -1, 0.5]
      ]
    }
  ]
}
```



SceneFactoryData.hh

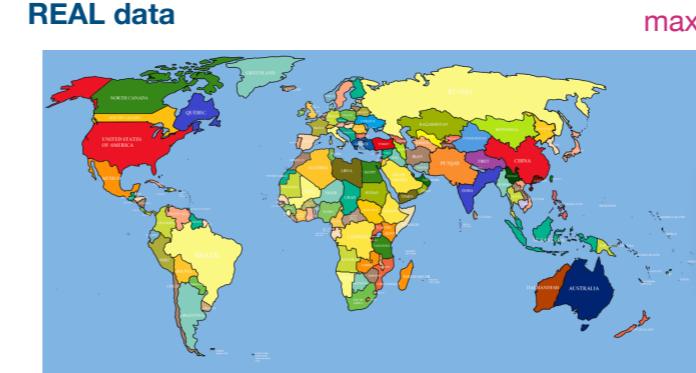


minPoint

```
"Virtual": {
  "minPoint": [-5, 0, -5],
  "maxPoint": [5, 2.0, 5]
},
```



VIRTUAL scene

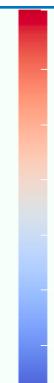
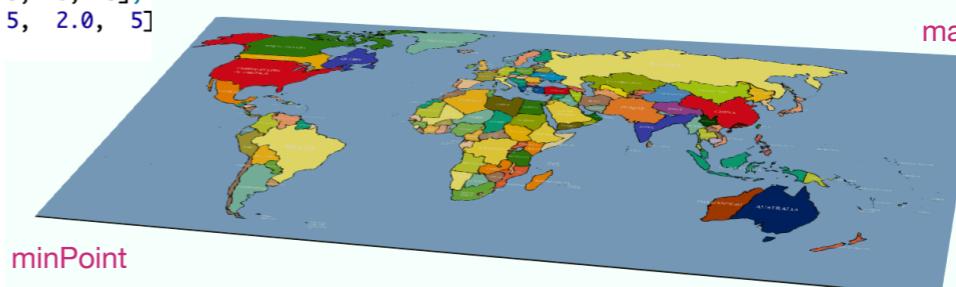


maxPoint

```
"Real": {
  "minPoint": [-5, -5],
  "maxPoint": [5, 5]
},
// dades: vector de parells <string,
// vector de dades d'aquell atribut>
vector<pair<QString, vector<vec3>>> dades;
shared_ptr<VisualMapping> mapping;
```

first	second		
name	x	z	valor
temperatura	-2	-1	0.5
	1	2	1.4

maxPoint



2. Visual Mapping

```
shared_ptr<Scene> SceneFactoryData::createScene(QString nameFile) {
    scene = make_shared<Scene>();
    load(nameFile);
    print(0);
    return visualMaps();
}
```

```
float Rxmin, Rxmax, Rzmin, Rzmax;
float Vxmin, Vxmax, Vymin, Vymax, Vzmin, Vzmax;
vector<AttributeMapping *> attributeMapping;
```

name	gizmo	min/max	Material	colorMap
temperatura	i			
humitat				

mapping

```
// dades: vector de parells <string,
// vector de dades d'aquell atribut>
vector<pair<QString, vector<vec3>>> dades;
shared_ptr<VisualMapping> mapping;
```

first	second			
name	x	z	valor	
i temperatura	j -2	-1	0.5	
humitat	1	2	1.4	

255

0

// Metode que mapeja les dades llegides a una escena virtual segons la informació del Visual Mapping

```
shared_ptr<Scene> SceneFactoryData::visualMaps() {
    // T0 D0: A partir de les dades carregades, ca
```

```
    shared_ptr<Material> SceneFactoryData::materialMaps(int i, int j) {
```

```
        AttributeMapping *propinfo = mapping->attributeMapping[i];
```

```
        auto tCM = propinfo->colorMapType;
        auto cm = make_shared<ColorMapStatic>(tCM);
```

```
        float valorDada = dades[i].second[j][2];
```

```
        auto tMat = MaterialFactory::getInstance().getIndexType(propinfo->material);
```

```
        // Calcul de l'index de la paleta
```

```
        int idx = (int)(255.0*(valorDada-propinfo->minValue)/(propinfo->maxValue-propinfo->minValue))
```

```
        return MaterialFactory::getInstance().createMaterial(propinfo->material->Ka,
                                                               cm->getColor(idx),
                                                               propinfo->material->Ks,
                                                               propinfo->material->Kt,
                                                               1.0, propinfo->material->shininess, tMat);
```

```
        auto o = objectMaps(i);
        o->setMaterial(materialMaps(i, j));

        // Afegir objecte a l'escena virtual
        scene->objects.push_back(o);
    }

    return scene;
}
```

2. Visual Mapping

```
shared_ptr<Scene> SceneFactoryData::createScene(QString nameFile) {
    scene = make_shared<Scene>();
    load(nameFile);
    print(0);
    return visualMaps();
}
```

```
float Rxmin, Rxmax, Rzmin, Rzmax;
float Vxmin, Vxmax, V ymin, Vymax, Vzmin, Vzmax;
vector<AttributeMapping *> attributeMapping;
```

name	gizmo	min/max	Material	colorMap
temperatura	i			
humitat				

mapping

```
// dades: vector de parells <string,
// vector de dades d'aquell atribut>
vector<pair<QString, vector<vec3>>> dades;
shared_ptr<VisualMapping> mapping;
```

first	second			
name	x	z	valor	
i temperatura	j -2	-1	0.5	
humitat	1	2	1.4	

// Metode que mapeja les dades llegides a una escena virtual segons la informació del Visual Mapping

```
shared_ptr<Scene> SceneFactoryData::visualMaps() {

    // T0 D0: A partir de les dades carregades, cal construir l'escena virtual amb tot colocado al seu lloc
    // i a la seva mida

    // T0 D0: Fase 1: PAS 5. Recorregut de les dades:
    for (unsigned int i=0; i< dades.size(); i++) {

        // Per cada valor de l'atribut, cal donar d'alta un objecte (gizmo) a l'escena
        for (unsigned int j=0; j<dades[i].second.size(); j++) {
            auto o = objectMaps(i); ← j TAMBE ES NECESSITA
            o->setMaterial(materialMaps(i, j));

            // Afegir objecte a l'escena virtual ja amb el seu material corresponent
            scene->objects.push_back(o);
        }
    }

    return scene;
}
```

OCEÀN
PACÍFIC

POBLACIÓN RE
O DESPLAZADA

2.000.000
1.000.000
500.000

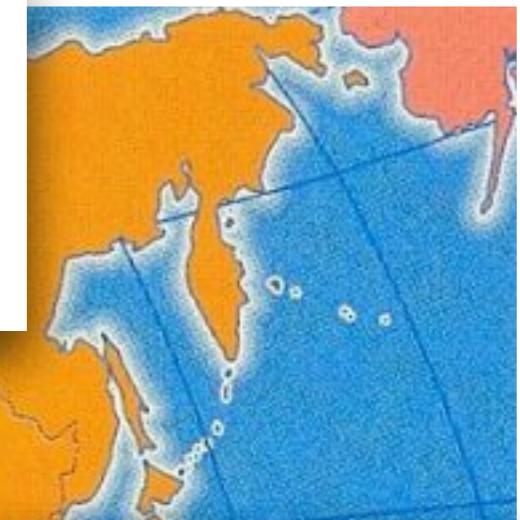
2. Visual Mapping

```
{  
  "scene": "Data0",  
  "typeScene": "REALDATA",  
  "Real": {  
    "minPoint": [-5, -5],  
    "maxPoint": [5, 5]  
  },  
  "Virtual": {  
    "minPoint": [-5, 0, -5],  
    "maxPoint": [5, 2.0, 5]  
  },  
  "attributes": [  
    {  
      "name": "temperatura",  
      "gyzmo": "sphere",  
      "minValue": 0,  
      "maxValue": 2,  
      "colorMap": "CIE1976",  
      "material": {  
        "type": "lambertian",  
        "ka": [0.2, 0.2, 0.2],  
        "kd": [0.7, 0.7, 0.7],  
        "ks": [0.7, 0.7, 0.7],  
        "shininess": 100  
      },  
      "data": [  
        [-2, -1, 0.5]  
      ]  
    }]  
  ]}  
  500.000
```

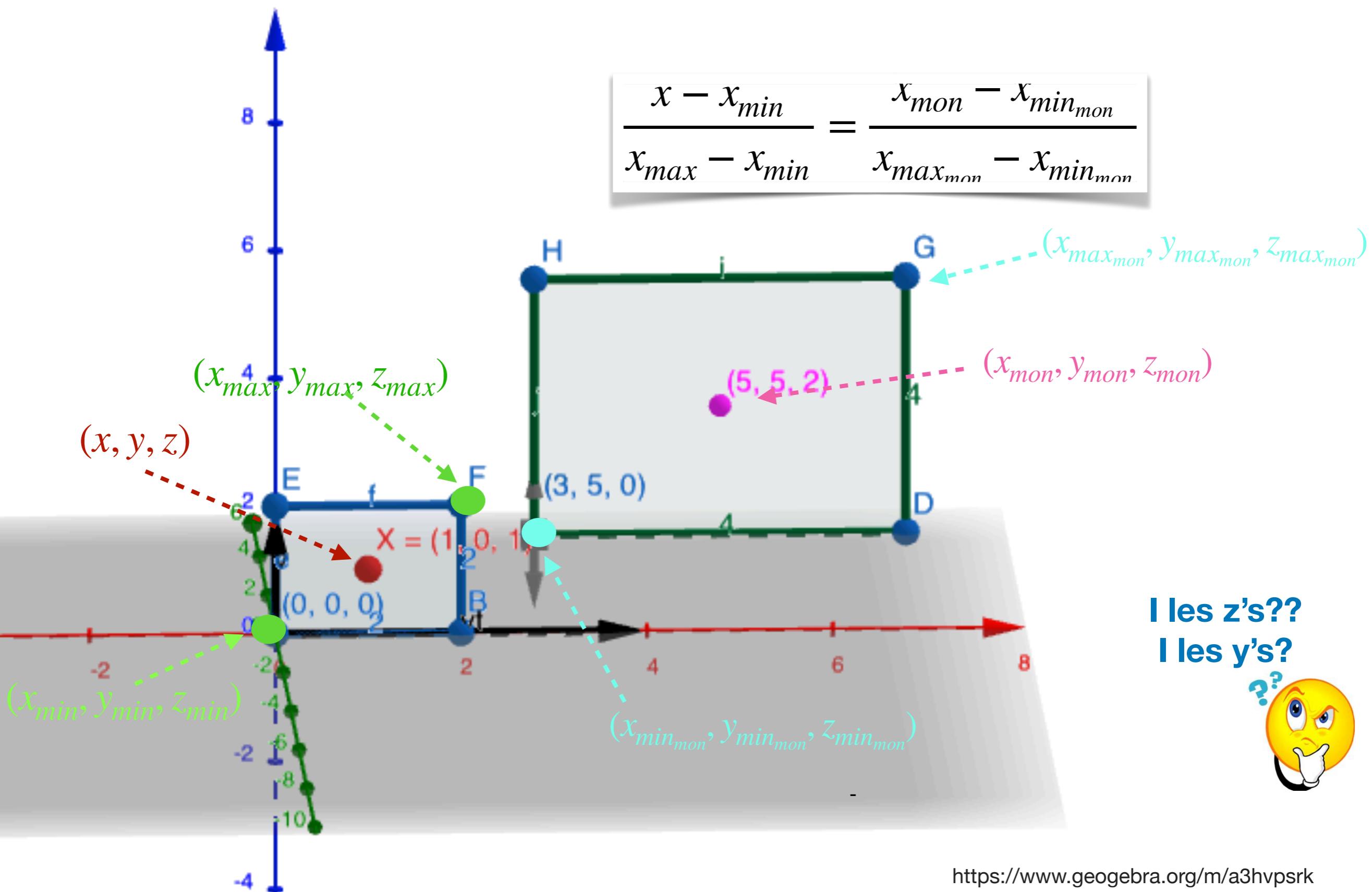
```
shared_ptr<Scene> SceneFactoryData::createScene(QString nameFile) {  
  scene = make_shared<Scene>();  
  load(nameFile);  
  print(0);  
  return visualMaps();  
}
```



```
shared_ptr<Object> SceneFactoryData::objectMaps(int i) {  
  // Gyzmo és el tipus d'objecte  
  shared_ptr<Object> o;  
  // Crea Objecte unitari  
  o = ObjectFactory::getInstance().createObject(mapping->attributeMapping[i]->gyzmo);  
  
  // TODO: Fase 1. Cal situar l'objecte unitari creat al (0,0,0) a escala proporcional  
  // monReal-monVirtual (valors de mapping) i el valor de la dada, i a la posició corresponent segons  
  // les coordenades donades a la dada (corresponen a x, z de mon virtual)  
  // Dades (x, y, z) --> Escena Virtual (x_v, 0, z_v) i l'objecte escalat segons  
  // la relació de y a escala amb el mon virtual  
  
  // a. Calcula primer l'escala  
  // b. Calcula la translació  
  // c. Aplica la TG a l'objecte usant  
  //     o->aplicaTG(transformacio)  
  
  return o;  
}
```



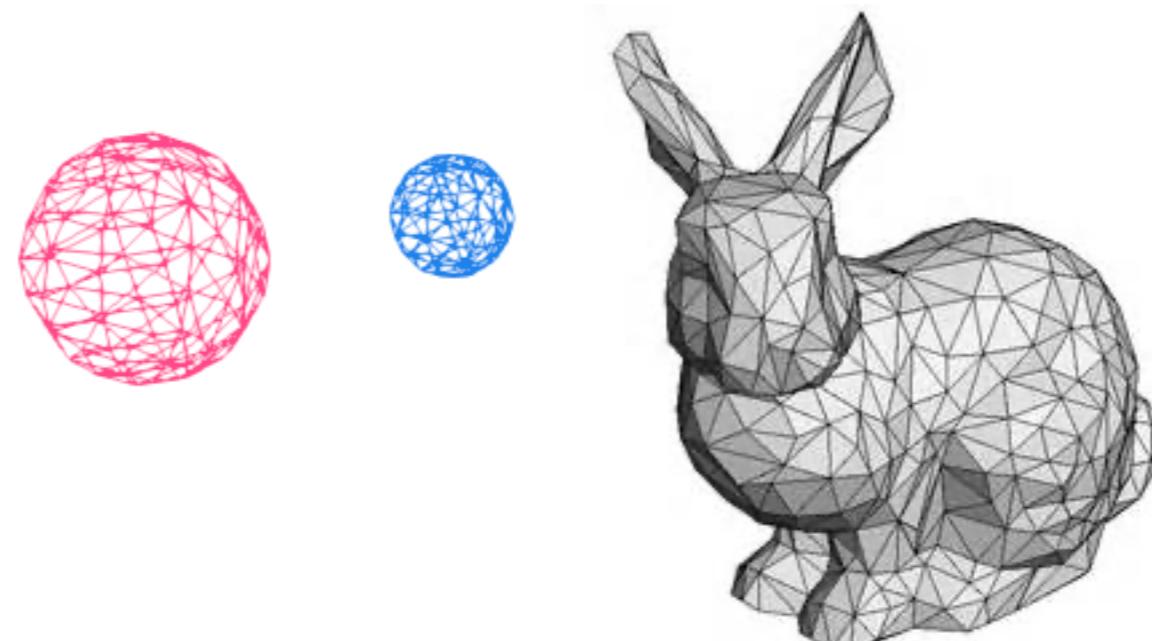
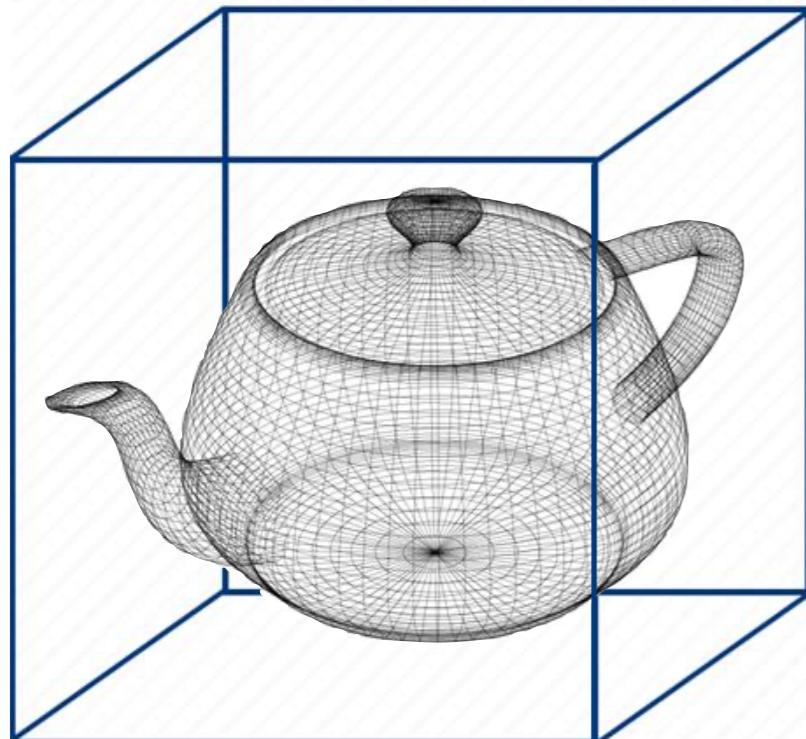
Transformaciones Geomètriques



2. Visual Mapping

Com tractem els .obj?

Problema: no se sap l'escala dels .obj carregats de fitxer



Com es calcula la capsula contenidora?

Com el transformem en unitari i centrat al $(0, 0, 0)$ sense deformacions?



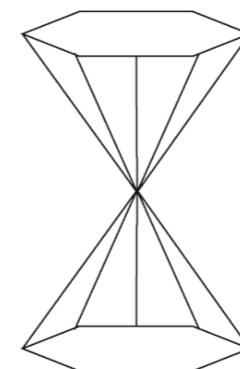
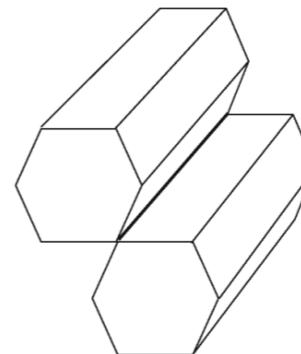
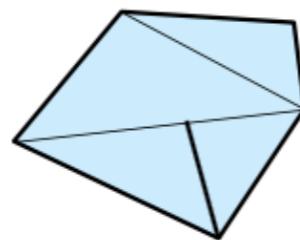
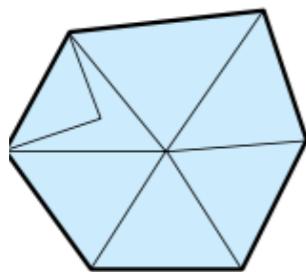
3. Malles o meshes

Quan es defineix una malla de polígons, quina de les següents propietats pot ser en algun cas CERTA?

Trieu-ne una:

- a. La intersecció d'un polígon de la mesh amb un altre pot ser directament un altre polígon.
- b. La intersecció de dos polígons pot ser buida
- c. Un vèrtex de la malla pot no pertànyer ni a una aresta ni a un polígon de la mesh
- d. Una aresta pot ser compartida per més de 2 polígons

[Esborra la meva selecció](#)



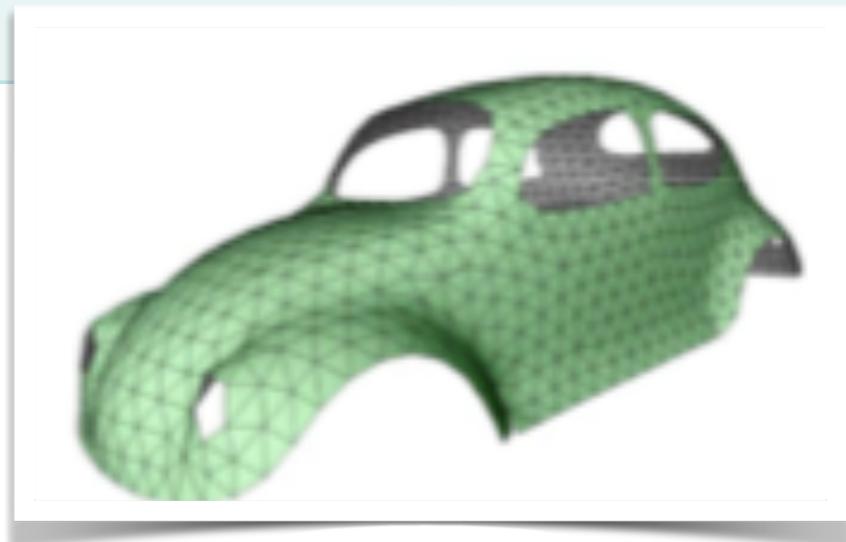
3. Malles o meshes

Si es té una malla poligonal tancada o no, quina de les següents propietats **NO** cal que compleixi?

Trieu-ne una:

- a. Totes les opcions anteriors s'han de satisfir.
- b. Totes les arestes han de ser compartides per dues cares tan en malles obertes com tancades.
- c. La intersecció de dos polígons qualsevol de la malla, ja sigui oberta o tancada, ha de ser buida o una aresta comú o un vèrtex comú.
- d. Tots els vèrtexs tenen un disc continu al seu voltant (situacions 2-manifold) si són malles tancades.

[Esborra la meva selecció](#)



3. Malles o mesh

Representacions de malles poligonals:

1. explícita
2. vèrtex indexats
3. adjacència de cares
4. winged edge

3. Malles o mesh

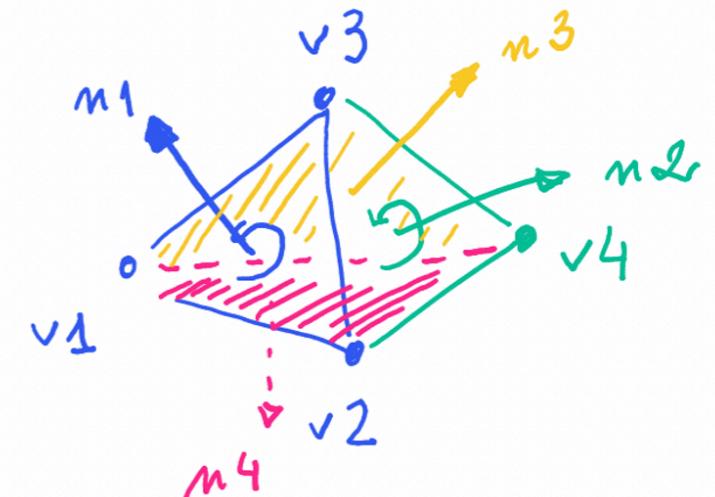
Representacions de malles poligonals:

1. explícita

2. vèrtex indexats

3. adjacència de cares

4. winged edge



x	y	z
v1.x	v1.y	v1.z
v2.x	v2.y	v2.z
v3.x	v3.y	v3.z
v2.x	v2.y	v2.z
v4.x	v4.y	v4.z
v3.x	v3.y	v3.z
v1.x	v1.y	v1.z
v2.x	v2.y	v2.z
v4.x	v4.y	v4.z
v1.x	v1.y	v1.z
v3.x	v3.y	v3.z
v4.x	v4.y	v4.z

3. Malles o mesh

Representacions de malles poligonals:

1. explícita

2. vèrtex indexats

3. adjacència de cares

4. winged edge

Per n triangles: $3*n$ ints
+ 1 vector de vèrtexs (3 floats per vèrtex)

Connectivitat de polígon explícita

No es coneixen les cares veïnes: no topologia de mesh

<https://youtu.be/hFRInNci3Rs>

v	x	y	z
0	1.0	1.0	1.0
1	-1.0	1.0	-1.0
2	-1.0	-1.0	1.0
3	1.0	-1.0	-1.0

Conjunt de vèrtexs:

x	y	z
---	---	---

t	i	j	k
0	0	1	2
1	0	2	3
2	0	3	1
3	3	2	1

Conjunt de cares:

idx primer vèrtex	idx segon vèrtex	idx tercer vèrtex
-------------------	------------------	-------------------

3. Malles o mesh

- Representació de mesh de polígons: **adjacència de cares:**

- Llista de vèrtexs:**
 - les seves coordenades
 - Index a una de les cares a la que pertany
- Llista de cares:**
 - Indexes als punts
 - Indexes a les cares adjacents

Per n triangles:
 $3*n_vertexs$ floats + 1
 $*n_vertexs$ enter + $6*n$ enters
Connectivitat de mesh i de polígon explícites

v	x	y	z	l
0	1.0	1.0	1.0	0
1	-1.0	1.0	-1.0	0
2	-1.0	-1.0	1.0	0
3	1.0	-1.0	-1.0	1

x	y	z	cara
---	---	---	------

t	i	j	k	n_0	n_1	n_2
0	0	1	2	3	1	2
1	0	2	3	3	2	0
2	0	3	1	3	0	1
3	3	2	1	0	2	1

idx primer vèrtex	idx segon vèrtex	idx tercer vèrtex	1 ^a cara adjacent	2 ^a cara adjacent	3 ^a cara adjacent
-------------------	------------------	-------------------	------------------------------	------------------------------	------------------------------

Topologia de malla

3. Malles o mesh

- Representació de mesh de polígons: Llistes de **cares+arestes+vèrtexs**: **model Winged-edge** (optimització)

<https://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/winged-e.html>

- Llista d'arestes:**

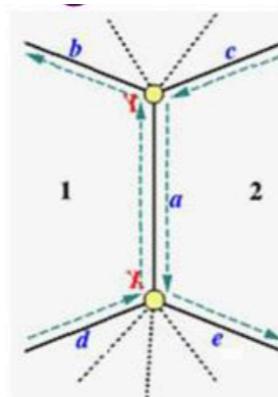
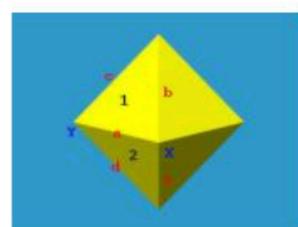
- Vèrtex inicial (VSTART) i final (VEND) - sentit horari
- Les dues cares adjacents (FCW-Face ClockWise i FCCW - Face CounterClockWise)
- Arestes anterior i posterior de les dues cares (EPCW, ENCW, EPCCW, ENCCW)

- Llista de vèrtexs:**

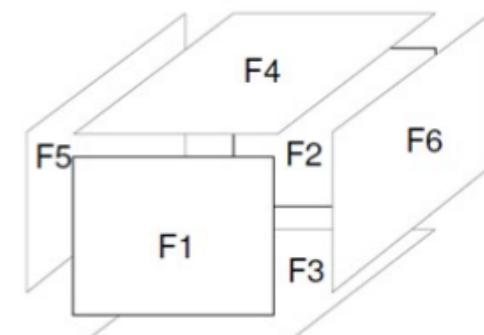
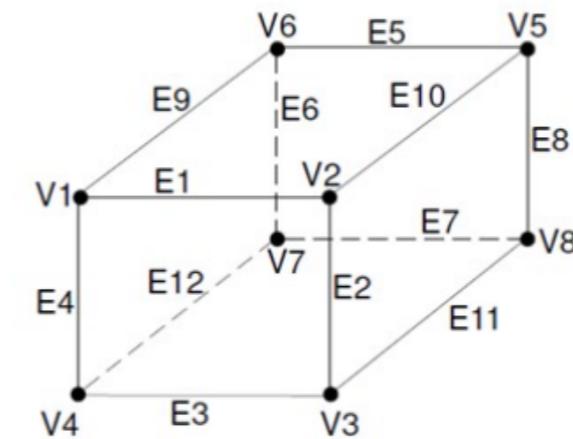
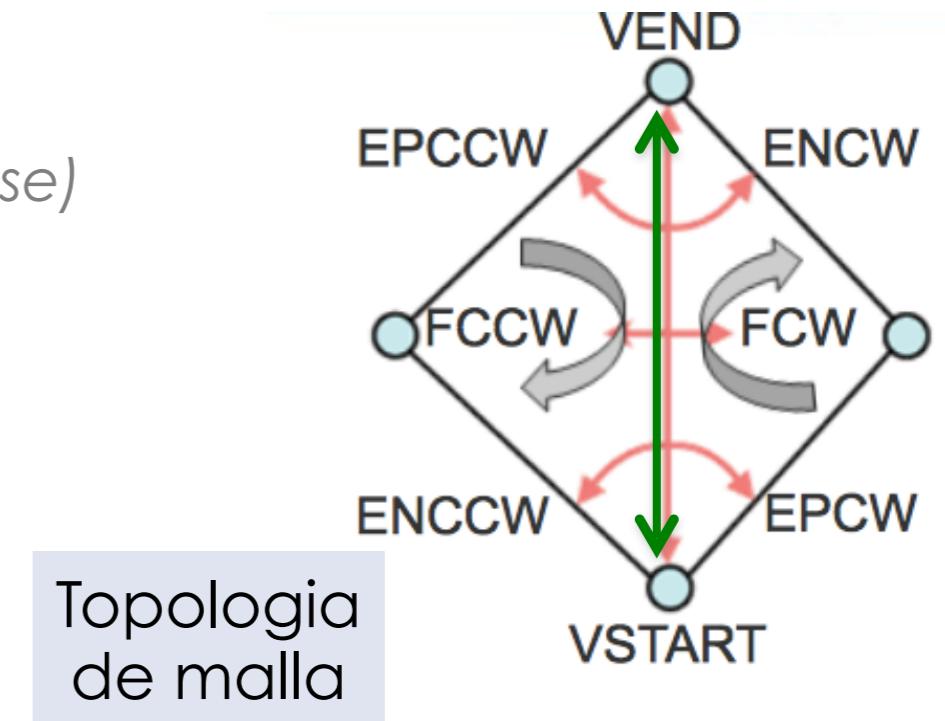
- Posició
- Una aresta adjacent

- Llista de cares:**

- Una aresta adjacent



Edge	Vertices		Faces		Left Traverse		Right Traverse	
Name	Start	End	Left	Right	Pred	Succ	Pred	Succ
a	x	y	1	2	d	b	c	e



3. Malles o mesh

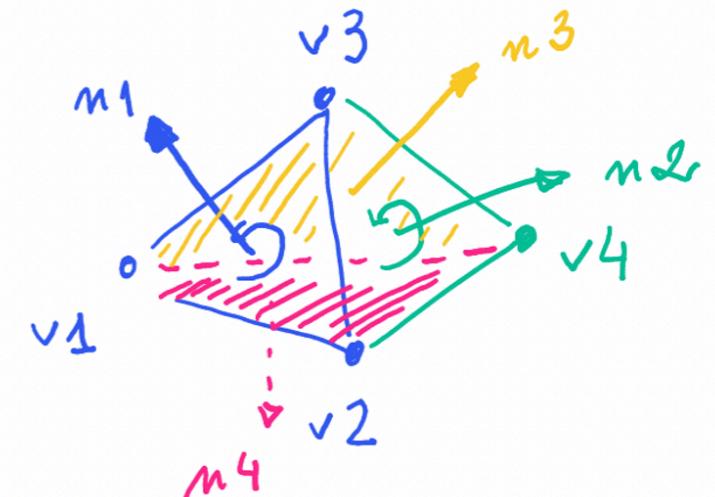
Representacions de malles poligonals:

1. explícita

2. vèrtex indexats

3. adjacència de cares

4. winged edge



x	y	z
v1.x	v1.y	v1.z
v2.x	v2.y	v2.z
v3.x	v3.y	v3.z
v2.x	v2.y	v2.z
v4.x	v4.y	v4.z
v3.x	v3.y	v3.z
v1	v1.y	v1.z
v2.x	v2.y	v2.z
v4.x	v4.y	v4.z
v1	v1.y	v1.z
v3.x	v3.y	v3.z
v4.x	v4.y	v4.z

3. Malles o mesh

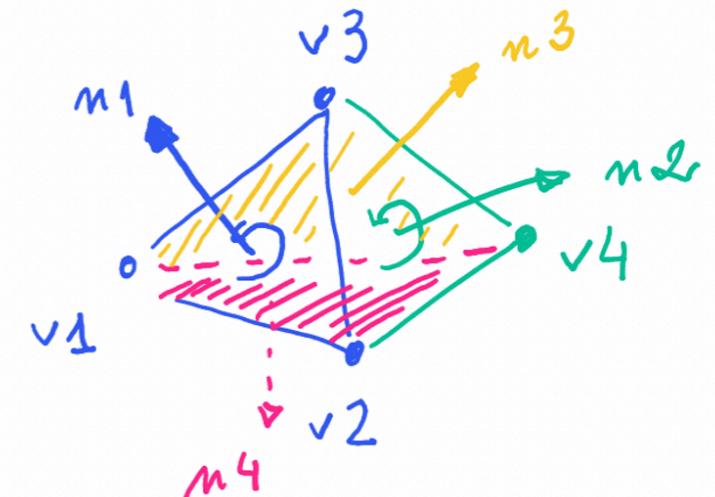
Representacions de malles poligonals:

1. explícita

2. vèrtex indexats

3. adjacència de cares

4. winged edge



x	y	z
v1.x	v1.y	v1.z
v2.x	v2.y	v2.z
v3.x	v3.y	v3.z
v2.x	v2.y	v2.z
v4.x	v4.y	v4.z
v3.x	v3.y	v3.z
v1	v1.y	v1.z
v2.x	v2.y	v2.z
v4.x	v4.y	v4.z
v1.x	v1.y	v1.z
v3.x	v3.y	v3.z
v4.x	v4.y	v4.z

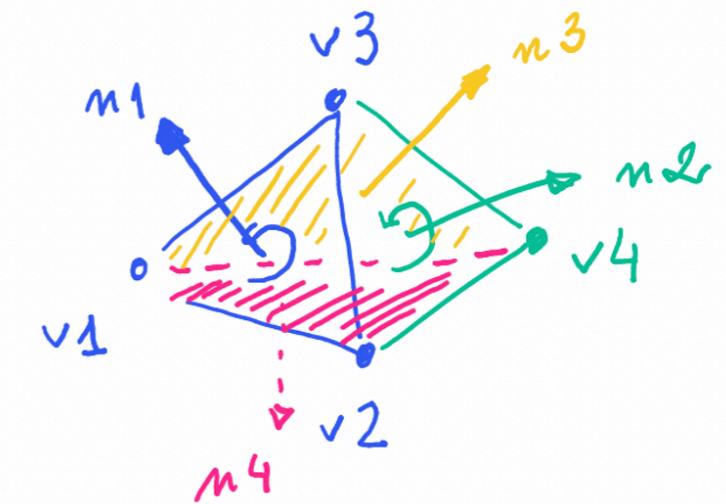
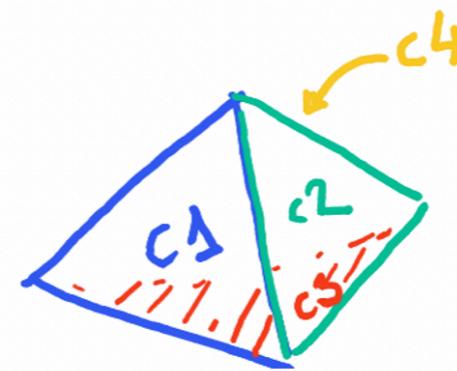
3. Malles o mesh

x	y	z
v1	v1.x	v1.y
v2	v2.x	v2.y
v3	v3.x	v3.y
v2	v2.x	v2.y
v4	v4.x	v4.y
v3	v3.x	v3.y
v1	v1.x	v1.y
v2	v2.x	v2.y
v4	v4.x	v4.y
v1	v1.x	v1.y
v3	v3.x	v3.y
v4	v4.x	v4.y

Definicions de malles poligonals:

íctica

2. vèrtex indexats



Conjunt de cares:

	idx primer vèrtex	idx segon vèrtex	idx tercer vèrtex
c1	0	1	2
c2	1	3	2
c3	0	1	3
c4	0	2	4

Conjunt de vèrtexs:

x	y	z
v1	v1.x	v1.y
v2	v2.x	v2.y
v3	v3.x	v3.y
v4	v4.x	v4.y

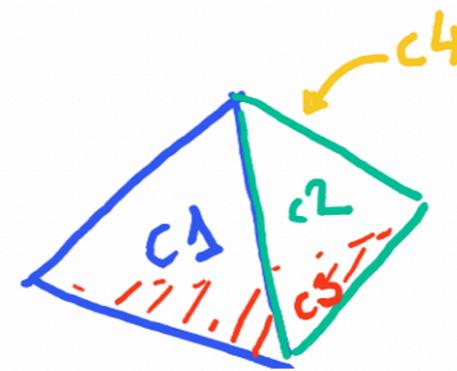


on podria guardar les normals?

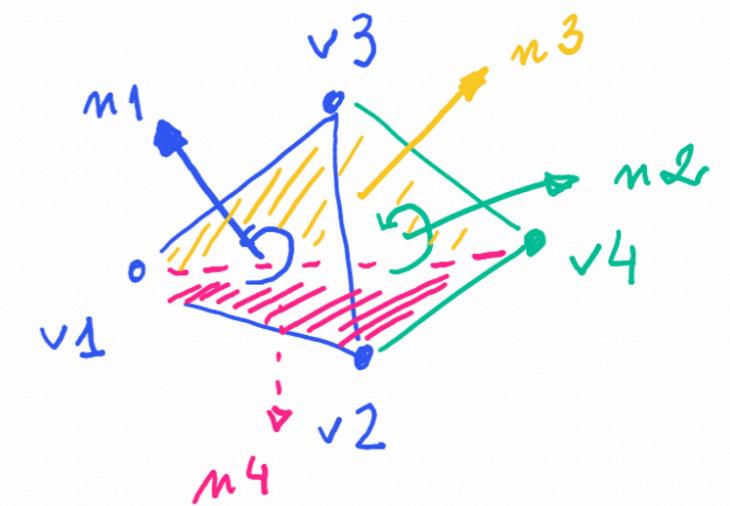
3. Malles o mesh

Representacions de malles poligonals:

1. explícita



2. vèrtex indexats



3. adjacència de cares

4. winged edge

Conjunt de cares:

idx primer	idx segon	idx tercer	1 ^a cara adjacent	2 ^a cara adjacent	3 ^a cara adjacent
------------	-----------	------------	------------------------------	------------------------------	------------------------------

c1?



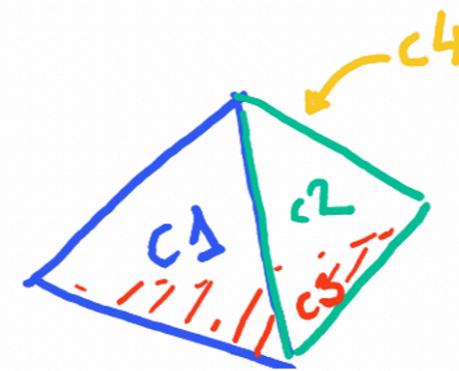
Conjunt de vèrtexs:

x	y	z	cara
---	---	---	------

3. Malles o mesh

Representacions de malles poligonals:

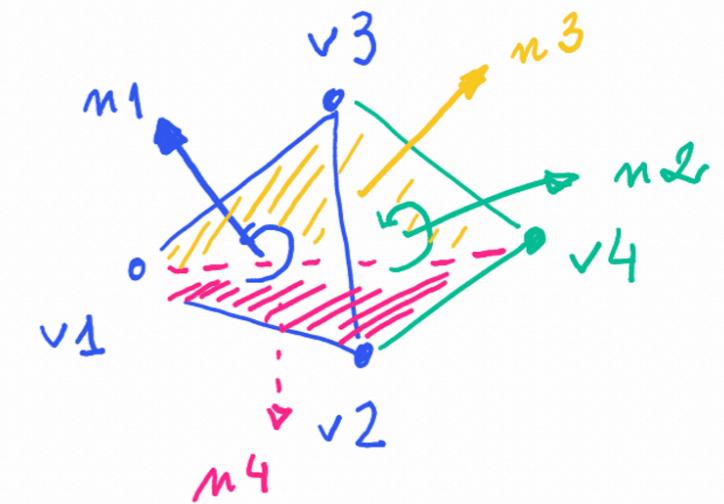
1. explícita



2. vèrtex indexats

3. adjacència de cares

4. winged edge



Conjunt de cares:

idx primer vèrtex	idx segon vèrtex	idx tercer vèrtex	1 ^a cara adjacent	2 ^a cara adjacent	3 ^a cara adjacent
c1 0	1	2	1	2	3
c2 1	3	2	2	3	0
c3 0	1	3	3	0	1
c4 0	2	3	0	1	2

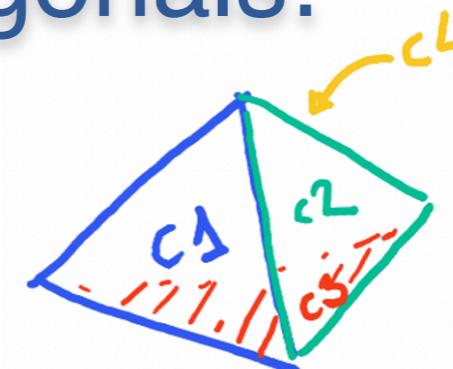
Conjunt de vèrtexs:

v1	x	y	z	cara
v1	v1.x	v1.y	v1.z	0
v2	v2.x	v2.y	v2.z	0
v3	v3.x	v3.y	v3.z	0
v4	v4.x	v4.y	v4.z	1

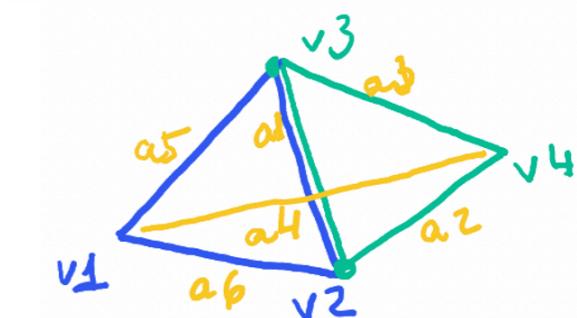
3. Malles o mesh

Representacions de malles poligonals:

1. explícita



2. àrees indexades



3. adjacències de cares



a1?

Conjunt arestes:

vstart vend FCW FCCW EPCW ENCW EPCCW ENCCW

4. winged edge

Conjunt de vèrtexs:

	x	y	z	aresta adjacent
v1	v1.x	v1.y	v1.z	0
v2	v2.x	v2.y	v2.z	0
v3	v3.x	v3.y	v3.z	0
v4	v4.x	v4.y	v4.z	1

Conjunt de cares:

aresta adjacent
c1
c2
c3
c4

3. Malles o mesh

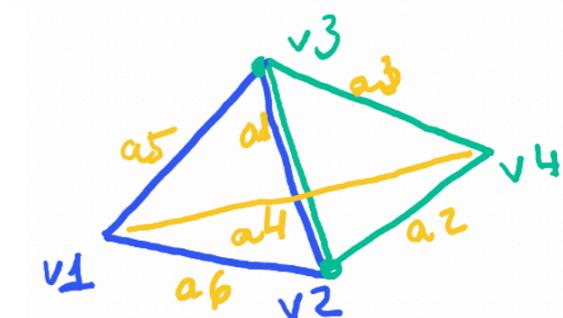
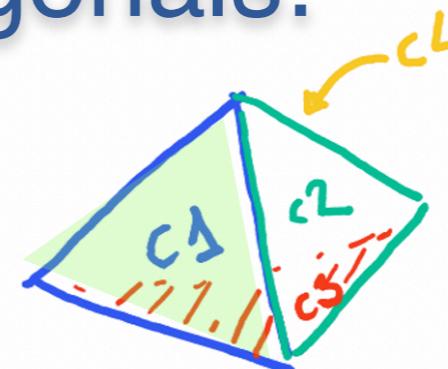
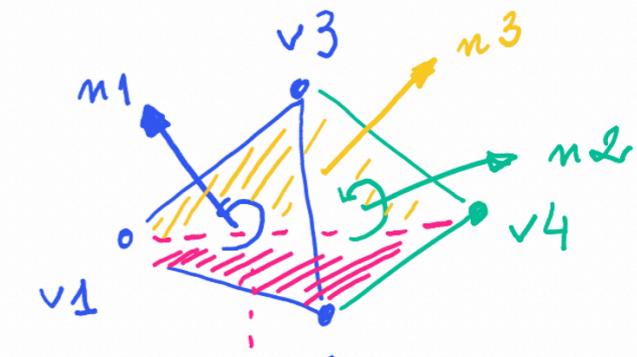
Representacions de malles poligonals:

1. explícita

2. vèrtex indexats

3. adjacència de cares

4. winged edge



Conjunt arestes:

	vstart	vend	FCW	FCCW	EPCW	ENCW	EPCCW	ENCCW
a1	1	2	1	0	2	1	5	4
a2	1	3	2	1	3	1	0	2
a3	3	2	3	1	4	3	1	0
a4	3	0	3	2	2	4	1	5
a5	2	0	3	0	3	2	0	5
a6	0	1	2	2	3	1	2	3

Conjunt de vèrtexs:

	x	y	z	aresta adjacent
v1	v1.x	v1.y	v1.z	0
v2	v2.x	v2.y	v2.z	0
v3	v3.x	v3.y	v3.z	0
v4	v4.x	v4.y	v4.z	1

Conjunt de cares:

aresta adjacent
c1
c2
c3
c4

ordre de recorregut CCW sempre!

3. Malles o mesh

En el model de representació de mesh de polígons, Winged-Edge, quina de les següents afirmacions és certa?

Trieu-ne una:

- a. Es guarda una llista de vèrtexs, una llista de cares i una llista de arestes. En la llista de vèrtexs és guarda la referència es guarda una de les cares en què es troba
- b. Es guarda una llista d'arestes, una llista de vèrtexs i una llista de cares. En la llista de cares és guarden les altres cares adjacents.
- c. Es guarda una llista de cares i una de vèrtexs. La llista de vèrtexs inclou a cada vèrtex la referència a una de les cares que la conté, i a la llista de cares s'emmagatzemen tots els índexs dels punts que representen els triangles i referències a les cares adjacents.
- d. Cap de les anteriors.

3. Malles o mesh

En el model de representació de mesh de polígons, Winged-Edge, quina de les següents afirmacions és certa?

Trieu-ne una:

- a. Es guarda una llista de vèrtexs, una llista de cares i una llista de arestes. En la llista de vèrtexs és guarda la referència es guarda una de les cares en què es troba
- b. Es guarda una llista d'arestes, una llista de vèrtexs i una llista de cares. En la llista de cares és guarden les altres cares adjacents.
- c. Es guarda una llista de cares i una de vèrtexs. La llista de vèrtexs inclou a cada vèrtex la referència a una de les cares que la conté, i a la llista de cares s'emmagatzemen tots els índexs dels punts que representen els triangles i referències a les cares adjacents.
- d. Cap de les anteriors.

[Esborra la meva selecció](#)

3. Malles o mesh

Quan es defineix una malla poligonal (o mesh de polígons) d'un objecte...

- a. per a calcular quina cara és adjacent a una altra, el model més eficient en memòria és la representació per adjacència de cares.
- b. la representació explícita és la més utilitzada ja que és eficient en memòria.
- c. si es modifiquen les coordenades d'un vèrtex, la representació winged-edge és la més eficient.
- d. si s'utilitza la representació per vèrtexs indexats s'eviten repeticions de punts i fa que la cerca de les cares a les que pertany un vèrtex sigui molt eficient.

3. Malles o mesh

Quan es defineix una malla poligonal (o mesh de polígons) d'un objecte...

- a. per a calcular quina cara és adjacent a una altra, el model més eficient en memòria és la representació per adjacència de cares.
- b. la representació explícita és la més utilitzada ja que és eficient en memòria.
- c. si es modifiquen les coordenades d'un vèrtex, la representació winged-edge és la més eficient.
- d. si s'utilitza la representació per vèrtexs indexats s'eviten repetitions de punts i fa que la cerca de les cares a les que pertany un vèrtex sigui molt eficient.

Conjunt de cares:

	idx primer vèrtex	idx segon vèrtex	idx tercer vèrtex	1 ^a cara adjacent	2 ^a cara adjacent	3 ^a cara adjacent
c1	0	1	2	1	2	3
c2	1	3	2	2	3	0
c3	0	1	3	3	0	1
c4	0	2	3	0	1	2

Conjunt de vèrtexs:

	x	y	z	cara
v1	v1.x	v1.y	v1.z	0
v2	v2.x	v2.y	v2.z	0
v3	v3.x	v3.y	v3.z	0
v4	v4.x	v4.y	v4.z	1

3. Malles o mesh



Si es vol fer un RayTracing per visualitzar objectes representats amb malles poligonals i es té un mètode que calcula el test d'intersecció que entre un raig i una malla poligonal tancada (o boundary object), quina és la millor representació de la malla tenint en compte l'eficiència en memòria i en temps?

- a. La representació explícita.
- b. La representació per vèrtexs indexats.
- c. La representació de cares adjacents.
- d. La representació winged-edge

3. Malles o mesh

Si es vol fer un RayTracing per visualitzar objectes representats amb malles poligonals i es té un mètode que calcula el test d'intersecció que entre un raig i una malla poligonal tancada (o boundary object), quina és la millor representació de la malla tenint en compte l'eficiència en memòria i en temps?

- a. La representació explícita.
- b. La representació per vèrtexs indexats.
- c. La representació de cares adjacents.
- d. La representació winged-edge

Conjunt de cares:

	idx primer vèrtex	idx segon vèrtex	idx tercer vèrtex
c1	0	1	2
c2	1	3	2
c3	0	1	3
c4	0	2	4

Conjunt de vèrtexs:

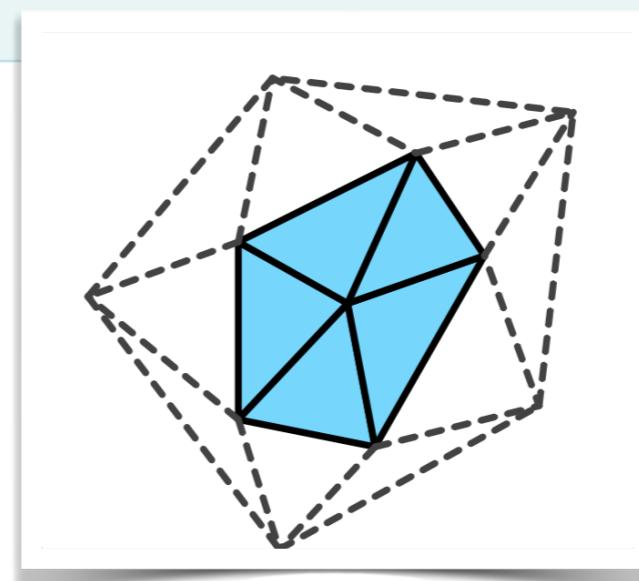
	x	y	z
v1	v1.x	v1.y	v1.z
v2	v2.x	v2.y	v2.z
v3	v3.x	v3.y	v3.z
v4	v4.x	v4.y	v4.z

3. Malles o mesh



Es defineix un model superficial d'un objecte poligonal format per m triangles i n vèrtexs, amb $m > 10000$. Es volen trobar per un vèrtex $v = (vx, vy, vz)$ de l'objecte, els triangles als que pertany v , suposant que un vèrtex com a molt pertany a 5 triangles, quina de les següents afirmacions és CERTA?

- a. amb la representació de vèrtexs indexats s'eviten problemes de robustesa de comparacions entre floats ja que guarda els índexs dels vèrtexs i no cal comparar floats a l'hora de cercar el vèrtex v .
- b. el model d'adjacències de cares és el més eficient en temps ja que només cal buscar el vèrtex v en el conjunt de vèrtexs per a obtenir el seu índex i l'índex a un dels triangles al que pertany. A partir d'aquest triangle ja es poden trobar la resta amb un màxim de 9 accessos a la taula de triangles.
- c. el model winged-edge és el més eficient en temps ja que només cal buscar el vèrtex v en el conjunt de vèrtexs per a obtenir el seu índex i l'índex a una de les seves arestes adjacents. A partir de l'aresta adjacent es poden trobar la resta d'arestes adjacents de forma directa, sense haver d'accendir a més posicions de la taula d'arestes.
- d. la representació explícita de vèrtexs, tot i que guarda $m * 3$ vèrtexs en memòria, és el més eficient en temps ja que codifica la connectivitat explícita de mesh permetent trobar directament tots els triangles als que pertany el vèrtex fent una cerca en la taula de vèrtexs de forma que quan se'n troben 5, ja es pot aturar la cerca.

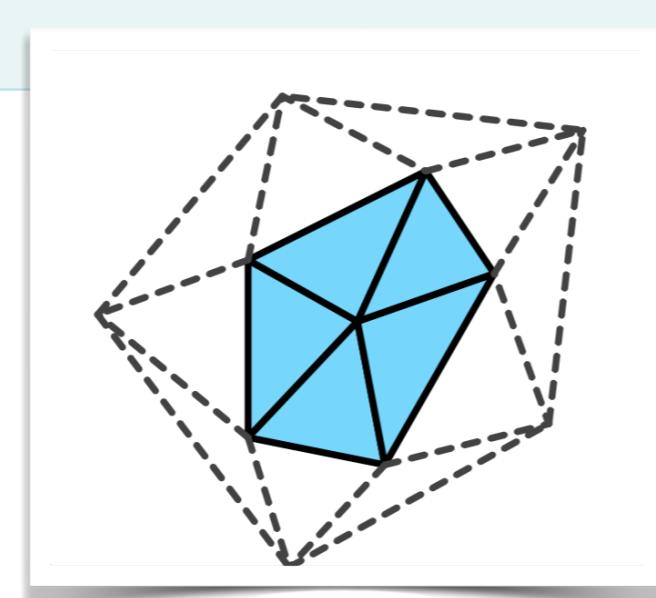


3. Malles o mesh



Es defineix un model superficial d'un objecte poligonal format per m triangles i n vèrtexs, amb $m > 10000$. Es volen trobar per un vèrtex $v = (vx, vy, vz)$ de l'objecte, els triangles als que pertany v , suposant que un vèrtex com a molt pertany a 5 triangles, quina de les següents afirmacions és CERTA?

- a. amb la representació de vèrtexs indexats s'eviten problemes de robustesa de comparacions entre floats ja que guarda els índexs dels vèrtexs i no cal comparar floats a l'hora de cercar el vèrtex v .
- b. el model d'adjacències de cares és el més eficient en temps ja que només cal buscar el vèrtex v en el conjunt de vèrtexs per a obtenir el seu índex i l'índex a un dels triangles al que pertany. A partir d'aquest triangle ja es poden trobar la resta amb un màxim de 9 accessos a la taula de triangles.
- c. el model winged-edge és el més eficient en temps ja que només cal buscar el vèrtex v en el conjunt de vèrtexs per a obtenir el seu índex i l'índex a una de les seves arestes adjacents. A partir de l'aresta adjacent es poden trobar la resta d'arestes adjacents de forma directa, sense haver d'accendir a més posicions de la taula d'arestes.
- d. la representació explícita de vèrtexs, tot i que guarda $m * 3$ vèrtexs en memòria, és el més eficient en temps ja que codifica la connectivitat explícita de mesh permetent trobar directament tots els triangles als que pertany el vèrtex fent una cerca en la taula de vèrtexs de forma que quan se'n troben 5, ja es pot aturar la cerca.



Conjunt de cares:

idx primer vèrtex	idx segon vèrtex	idx tercer vèrtex	1 ^a cara adjacent	2 ^a cara adjacent	3 ^a cara adjacent
c1 0	1	2	1	2	3
c2 1	3	2	2	3	0
c3 0	1	3	3	0	1
c4 0	2	3	0	1	2

Conjunt de vèrtexs:

	x	y	z	cara
v1	v1.x	v1.y	v1.z	0
v2	v2.x	v2.y	v2.z	0
v3	v3.x	v3.y	v3.z	0
v4	v4.x	v4.y	v4.z	1

3. Malles o mesh



Volem carregar un objecte de tipus mesh/malla poligonal. En el fitxer d'on llegim l'objecte hi tenim les coordenades dels vèrtexs, suposant l'objecte centrat en l'origen de coordenades, i a part ens donen el centre de l'objecte. Volem carregar l'objecte amb la intenció que un cop creat l'objecte poder-li aplicar diverses transformacions geomètriques de tipus translacions i escalat i que siguin el més eficients possibles. Quina de les següents estratègies és millor?

Trieu-ne una:

- a. És indiferent el tipus de representació i la manera com guardem els vèrtexs i el centre ja que el nombre d'operacions en la translació i l'escalat és el mateix.
- b. Utilitzem una representació per vèrtexs indexats i guardem els vèrtexs ja traslladats pel vector "centre de l'objecte".
- c. Utilitzem una representació per vèrtexs indexats i guardem els vèrtexs tal qual ens els donen i centre de l'objecte com un atribut a part.
- d. Utilitzem una representació explícita i guardem els vèrtexs ja traslladats pel vector "centre de l'objecte".

3. Malles o mesh



Volem carregar un objecte de tipus mesh/malla poligonal. En el fitxer d'on llegim l'objecte hi tenim les coordenades dels vèrtexs, suposant l'objecte centrat en l'origen de coordenades, i a part ens donen el centre de l'objecte. Volem carregar l'objecte amb la intenció que un cop creat l'objecte poder-li aplicar diverses transformacions geomètriques de tipus translacions i escalat i que siguin el més eficients possibles. Quina de les següents estratègies és millor?

Trieu-ne una:

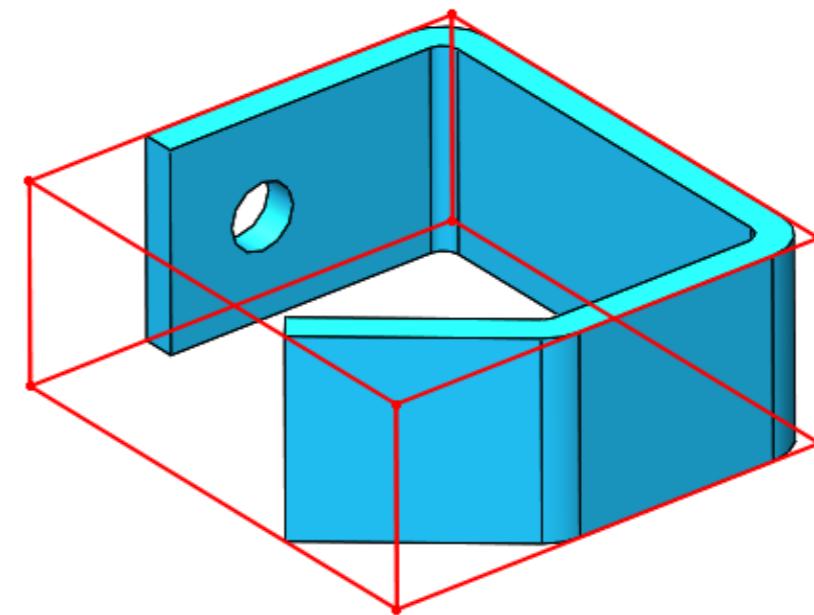
- a. És indiferent el tipus de representació i la manera com guardem els vèrtexs i el centre ja que el nombre d'operacions en la translació i l'escalat és el mateix.
- b. Utilitzem una representació per vèrtexs indexats i guardem els vèrtexs ja traslladats pel vector "centre de l'objecte".
- c. Utilitzem una representació per vèrtexs indexats i guardem els vèrtexs tal qual ens els donen i centre de l'objecte com un atribut a part.
- d. Utilitzem una representació explícita i guardem els vèrtexs ja traslladats pel vector "centre de l'objecte".

[Esborra la meva selecció](#)

3. Malles o mesh

Si es vol calcular la dimensió d'objectes representats amb malles poligonals i es té un mètode que calcula la capsula contenidora 3D d'una malla, quina és la millor representació tenint en compte l'eficiència en memòria i en temps?

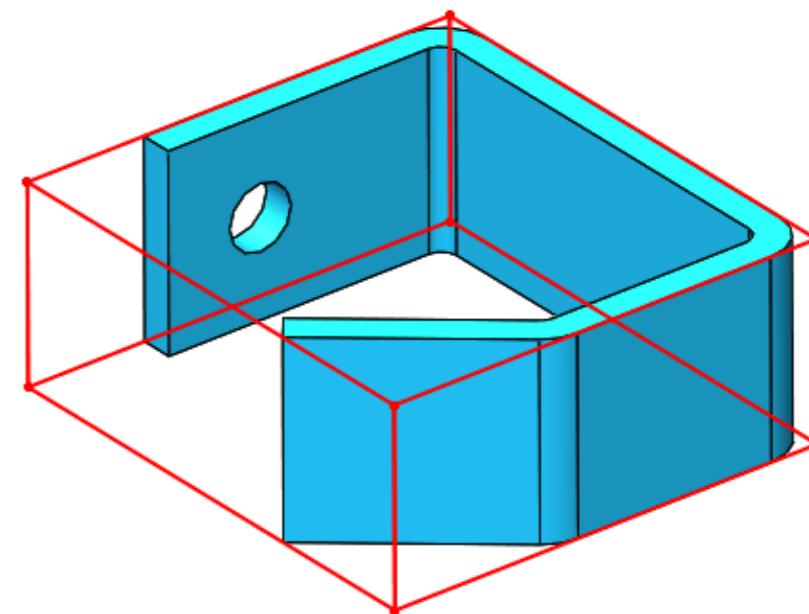
- a.La representació explícita.
- b.La representació per vèrtexs indexats.
- c.La representació per adjacència de cares.
- d.El model winged-edge.



3. Malles o mesh

Si es vol calcular la dimensió d'objectes representats amb malles poligonals i es té un mètode que calcula la capsula contenidora 3D d'una malla, quina és la millor representació tenint en compte l'eficiència en memòria i en temps?

- a. La representació explícita.
- b. La representació per vèrtexs indexats.**
- c. La representació per adjacència de cares.
- d. El model winged-edge.



v	x	y	z
0	1.0	1.0	1.0
1	-1.0	1.0	-1.0
2	-1.0	-1.0	1.0
3	1.0	-1.0	-1.0

t	i	j	k
0	0	1	2
1	0	2	3
2	0	3	1
3	3	2	1

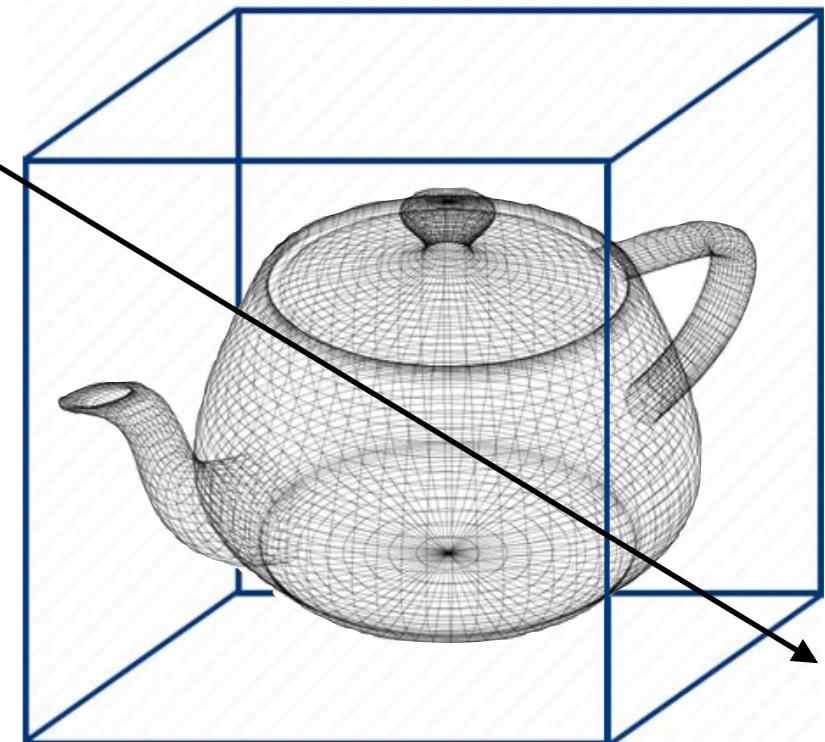
3. Malles o mesh a la pràctica

En quina representació es llegeix?

```
class Mesh : public Object  
{
```

```
    QString nom;  
    vector<Face> cares; // facees o cares de l'objecte  
    vector<vec4> vertexs; // vertexs de l'objecte sense repetits  
  
    void load(QString filename);  
    void makeTriangles();
```

```
class Face  
{  
public:  
    Face();  
  
    // constructor a partir de 3 o 4 indexs a vertex  
    Face(int i1, int i2, int i3, int i4=-1);  
  
    vector<int> idxVertices; // vector amb els indexs dels vertexs de la cara  
};
```



Com es fa el hit?

Com es calcula la capsula contenidora?

Com el transformem en unitari i centralitzat al (0, 0, 0)?



Observacions de la pràctica

```
shared_ptr<Scene> SceneFactoryVirtual::createScene(Serializable::SaveFormat saveFormat,
                                                 QString filename) {

    scene= make_shared<Scene>();
    load(saveFormat, filename);
    print(0);

    return scene;
}

void SceneFactoryVirtual::read(const QJsonObject &json)
{
    if (json.contains("scene") && json["scene"].isString())
        scene->name = json["scene"].toString();
    if (json.contains("typeScene") && json["typeScene"].isString())
        currentType = getSceneFactoryType(json["typeScene"].toString());

    if (json.contains("objects") && json["objects"].isArray()) {
        QJsonArray objectsArray = json["objects"].toArray();

        for (int objectIndex = 0; objectIndex < objectsArray.size(); objectIndex++) {
            QJsonObject objectObject = objectsArray[objectIndex].toObject();
            shared_ptr<Object> o;
            if (objectObject.contains("type") && objectObject["type"].isString()) {
                QString objStr = objectObject["type"].toString().toUpper();
                o = ObjectFactory::getInstance().createObject(ObjectFactory::getInstance().getObjectType(objStr));
                o->read(objectObject);
                scene->objects.push_back(o);
            }
        }
    }
}
```

Observacions de la pràctica

```
void Mesh::read (const QJsonObject &json)
{
    Object::read(json);
    if (json.contains("objFileName") && json["objFileName"].isString()) {
        nom = json["objFileName"].toString();
        load(nom);
    }
}

void Mesh::load (QString fileName) {
    QFile file(fileName);
    if(file.exists()) {
        if(file.open(QFile::ReadOnly | QFile::Text)) {
            while(!file.atEnd()) {
                QString line = file.readLine().trimmed();
                QStringList lineParts = line.split(QRegularExpression("\s+"));
                if(lineParts.count() > 0) {

```

```
{
    "name": "Br Objecte",
    "type": "MESH",
    "objFileName":(":/resources/cube.obj",
    "material": {
        "type": "lambertian",
        "ka": [0.2, 0.2, 0.2],
        "kd": [0.7, 0.6, 0.5],
        "ks": [0.7, 0.7, 0.7],
        "shininess": 10.0,
        "kt": [0.0, 0.0, 0.0],
        "nut": 1
    }
}
```

ha d'afegir-se a les resources del projecte!

