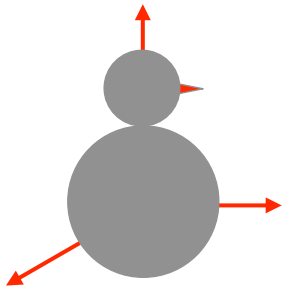
Bloc_2: Transformacions Geomètriques i Models

Sessions i Objectius

- Sessió 1 –seccions 1 a 3- : Transformacions Geomètriques
  - Objectes glut
  - Entendre el funcionament de les transformacions geomètriques per: posicionar i animar objectes.
  - Utilització en OpenGL.
  - Exercici: creació ninot
- Sessió 2 –seccions 4 i 5-:
  - Utilització de les transformacions geomètriques per a inspeccionar objectes.
  - Models geomètrics (OBJ) i visualització en OpenGL.
- Sessió 3 –secció 6- :
  - Aplicació resum de conceptes: sistema solar amb astronauta.

Què heu de fer en 1ra sessió Bloc 2?

- Pintar algun objecte glut (secció 1)
- Utilitzar OpenGL per aplicar TG a un objecte (secció 2)
  - Entendre els paràmetres de les crides i composició d'operacions
  - Recordeu queOpenGL aplica la matriu del top de la pila MODELVIEW als vèrtexs i que les operacions amb matrius afecten a la matriu del top de la pila activa.
  - Utilitzeu callbacks de teclat i ratolí per modificar TG
- Utilitzar OpenGL per a aplicar diferentes TG als diferentes objectes de l'escena
  - Caldrà Push/Pop Matrius
  - Gir dels dos triangles
- Crear una escena utilitzant objectes glut (secció 3)



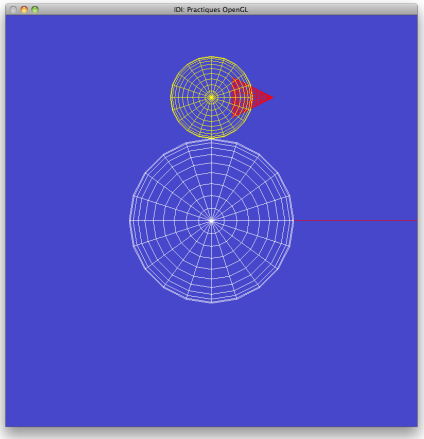
Què heu de fer en 2a sessió Bloc 2?

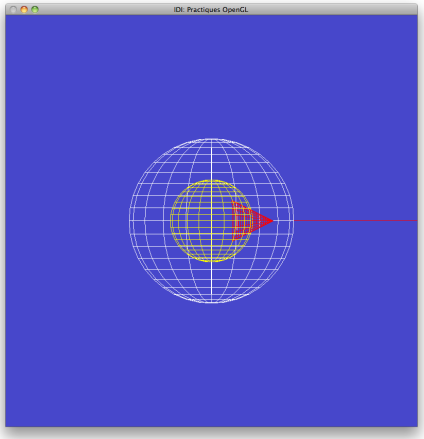
- Inspecció d'objectes (secció 4)
  - TG es poden utilitzar per a veure els objectes des de diferents punts de vista
  - Veurem més en el Bloc 3

- Càrrega de models OBJ (secció 5)
  - Estructura de dades i classe model
  - Exemple 1: pintat d'un model correctament ubicat en volum de visió: HomerProves.obj.
  - Exemple 2: pintat d'un model que cal ubicar centrat en origen coordenades i sense retallar dins volum de visió => càlcul capsa objecte i TG a aplicar al model .

Inspecció d'objectes (1)

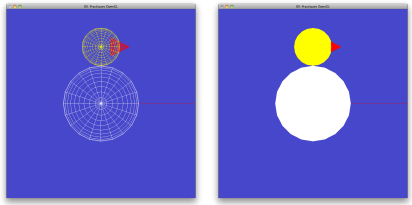
- Inspecció d'objectes: Una primera opció
  - Recordeu que càmera per defecte està en origen de coordenades, mirant en direcció de l'eix Z negatiu i és ortogonal i volum de visió (-1,-1,-1) a (1,1,1).
  - Vista en planta, alçat i perfil i TG= G_x * G_y : prement "v" obtenir diferents vistes.

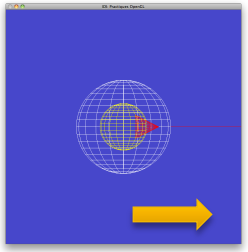




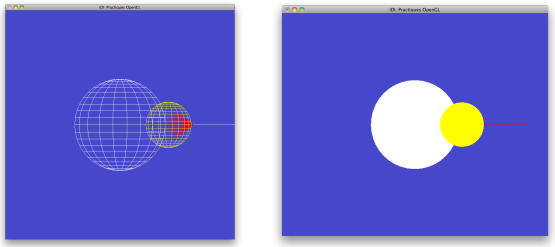
Inspecció d'objectes (2)

- Inspecció interactiva d'objectes:



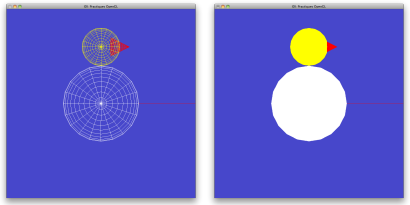


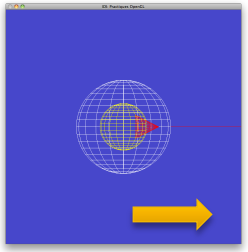
+ Desplaçament ratolí



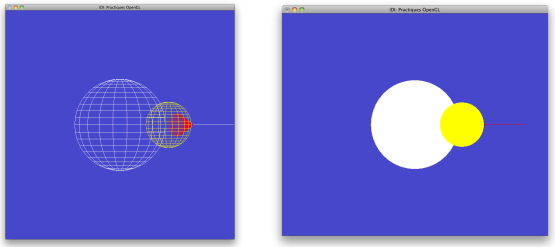
Inspecció d'objectes (3)

- Inspecció interactiva d'objectes:

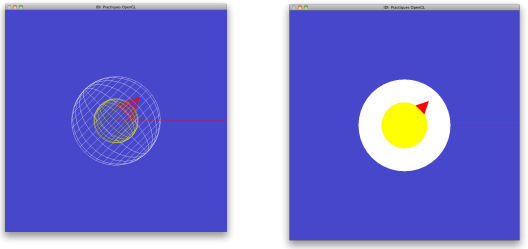




+ Desplaçament ratolí



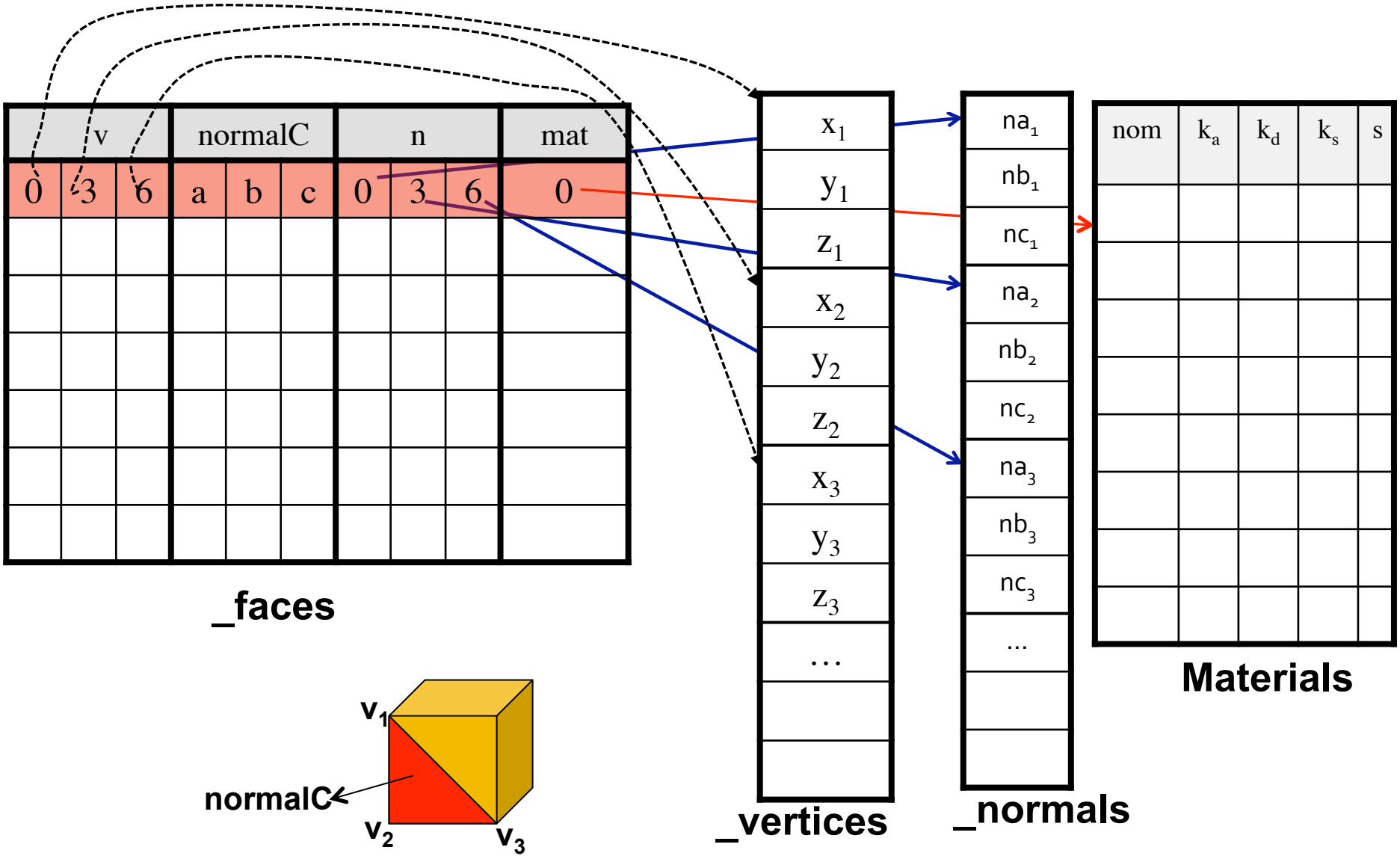
- Modificant G_y en $TG=G_x * G_y$ ➔



- Cal modificar TG actual per esquerra ➔ $TG^* = G_y * TG_{Actual}$ ➔ Codi Secció 4

Càrrega de Models (1)

- Classe Model: permet carregar models .obj
  - /assig/idi/Model
- Analitzeu l'arxiu model.h
- `Model::load(std::string filename)`
  - Inicialitza les estructures de dades a partir d'un model en format OBJ-Wavefront en disc
- /assig/vig/models
  - Si els copieu, per cada .obj copieu també (si existeix) el .mtl → definició dels materials corresponents
- Més a la xarxa


v
normalC
n
mat
0
3
6
a
b
c
0
3
6
0
_faces
x1
y1
z1
x2
y2
z2
x3
y3
z3
…
_vertices
na1
nb1
nc1
na2
nb2
nc2
na3
nb3
nc3
...
_normals
nom
ka
kd
ks
s
Materials
v1
normalC
v2
v3

Classe Model

- Model::load( )
  - Totes les cares són triangles
  - Les cares es triangulen en el moment de llegir-se
  - Tots els vectors de vèrtexs del model resultant són de tres components
- El vector de normals pot ser buit (si el model original no en té per vèrtex)
- Sempre podeu fer servir el vector Face::normalC (normal per cara)
  - Model::load() l'haurà inicialitzat amb un vector unitari perpendicular al triangle

Classe Model

- Tres std::vector<T> de la stl:
  - Un de coordenades: _vertices; un de components de normals: _normals i un de cares: _faces
- Mètodes consultors que retornen const
  - El codi en què les feu servir haurà de ser "const-correcte"

```
const std::vector< Face>&  faces() const {
    return _faces; }

conts Face &f = m.faces()[12];
glvertex3dv(&m.vertices() [f.v[0])];
```

Càrrega de models

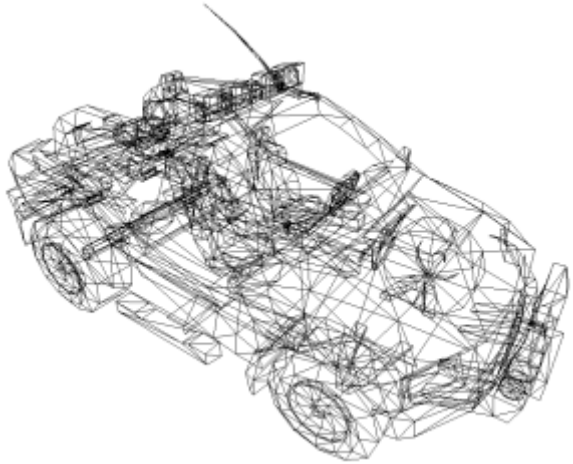
- Mètode que permet carregar OBJ (ini_escena)
  - HomerProves.OBJ
- refresh(): ha de recòrrer totes les cares de tots els models i enviar a pintar

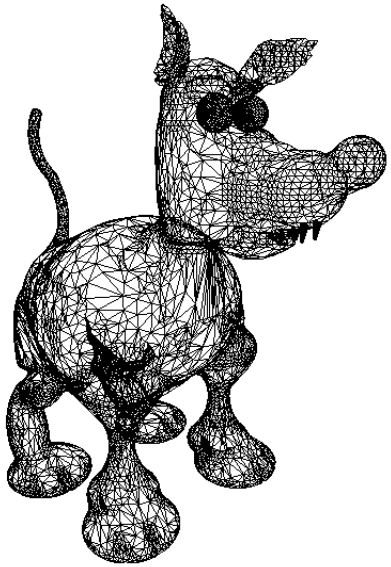
- Completar mètode de càrrega per:
  - Càlcul capsa mínima contenidora
  - Càlcul escalat, translació
- refresh (): ha de calcular TG abans d'enviar a pintar.
  - Homer.OBJ

glPolygonMode

- void glPolygonMode(GlEnum face, GlEnum mode)
  - face: les cares a les que ens referim
    - GL_FRONT_AND_BACK → aquest
    - GL_FRONT
    - GL_BACK
  - mode: mode de dibuix
    - GL_POINT
    - GL_LINE
    - GL_FILL

glPolygonMode





Depth test

- Algoritme de z-buffer
  - *glEnable(GL_DEPTH_TEST);*
  - Esborrar el buffer de profunditat:
    glClear(…. |GL_DEPTH_BUFFER_BIT);
  - En glutInitDisplayMode afegir:
    |GLUT_DEPTH
  - Per no tenir problemes amb volum de visió de defecte en Init_GL:

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho (-1.,1.,-1.,1.,-1.,1.);
glMatrixMode (GL_MODELVIEW);
```