# Laboratori OpenGL – Sessió 2.3

- Zoom (òptica perspectiva)
- Creació d'una escena completa
- Òptica ortogonal
- *Resize* també amb òptica ortogonal

# Zoom
## (exercici 1)

- Per a fer un zoom ho farem modificant l'angle d'obertura de la càmera (FOV)
  - Zoom-in → decrementar l'angle FOV (tecla 'Z')
  - Zoom-out → incrementar l'angle FOV (tecla 'X')
- Noteu que heu de controlar el valor màxim i mñinim de FOV...
- També ho podeu fer amb el ratolí i el botó dret

# Interacció directa amb Qt

- Per tal de tractar events de baix nivell en una aplicació OpenGL amb Qt cal re-implementar els mètodes virtuals corresponents (a la classe MyGLWidget):

```
virtual void mousePressEvent ( QMouseEvent * e )
virtual void mouseReleaseEvent ( QMouseEvent * e )
virtual void mouseMoveEvent ( QMouseEvent * e )
virtual void keyPressEvent ( QKeyEvent * e )
```

# Interacció directa amb Qt

- Exemple d'implementació:

```
void MyGLWidget::keyPressEvent (QKeyEvent *e) {
   makeCurrent ();
   switch ( e->key() ) {
      ...
       case  Qt::Key_Z :
             zoom= zoom - inc_zoom  //controlar que FOVa+zoom  > 20º p.e.
              projectTransform()  //aquí utilitzar FOVa+zoom
             break;
       case  Qt::Key_X :
             ....
             break;
       default: e->ignore (); // propagar al pare
   }
   update ();
}
```

# Interacció directa amb Qt

- Per tal de tractar events de baix nivell en una aplicació OpenGL amb Qt cal re-implementar els mètodes virtuals corresponents (a la classe MyGLWidget):

**virtual void mousePressEvent ( QMouseEvent * e )**

**virtual void mouseReleaseEvent ( QMouseEvent * e )**

**virtual void mouseMoveEvent ( QMouseEvent * e )**

virtual void keyPressEvent ( QKeyEvent * e )

**En MyGLWidget.h caldrà: #include <QMouseEvent>**
**i declarar el mètodes virtuals**

# Interacció directa amb Qt

- Exemple d'implementació:

```
void MyGLWidget::mousePressEvent (QMouseEvent *e)
{
   makeCurrent();
   xClick=e->x();
   yClick = e ->y();
   if ( e->buttons() == Qt::LeftButton )
       DoingInteractive=ROTATE;
   if ( e->buttons() == Qt::RightButton )
      DoingInteractive=ZOOM;
}
```

**Tenir diferents "estats" d'interacció,** caldrà haver declarat en .h

```
typedef enum {ROTATE, NONE, ZOOM} InteractiveAction;
InteractiveAction DoingInteractive;
```

# Interacció directa amb Qt

- Exemple d'implementació:

```
void MyGLWidget::mouseMoveEvent (QMouseEvent *e)
{
    makecurrent();
     if (DoingInteraction==ROTATE) {
          psi+=( e->x() – xClick)*M_PI/100.0;
          …
          xclick=e->x();
          viewTransform();
     }
       if (DoingInteraction==ZOOM)
         {…..}
      update();
}
```

# Interacció directa amb Qt

- Exemple d'implementació:

```
void MyGLWidget::mouseReleaseEvent (QMouseEvent *e)
{
    makecurrent();
    DoingInteraction=NONE;
}
```

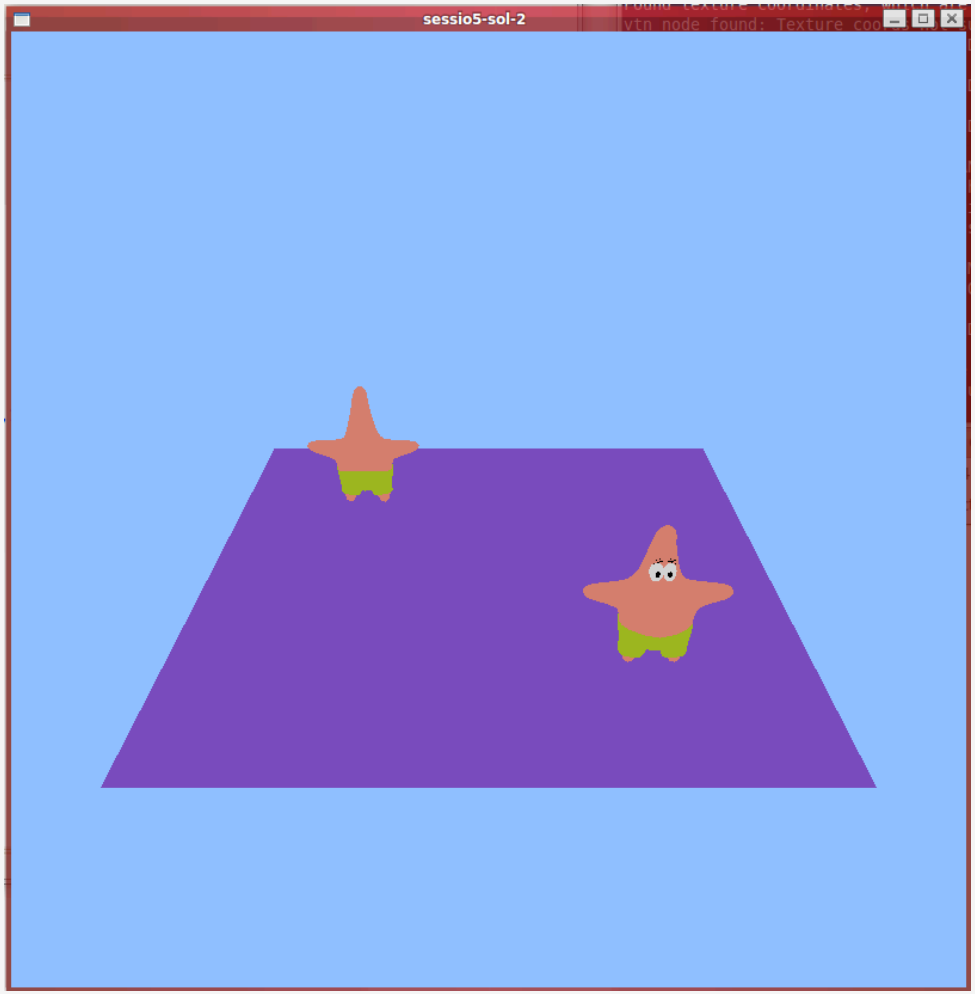
# Escena completa (exercici 2)

Modifiqueu la vostra escena per a veure el que es veu a la imatge.

La nova escena està formada per:

- Terra de 4x4 centrat al (0,0,0)
- Patricios d'alçada 1 amb centres base en (1,0,1) i (-1,0,-1). El segon rotat 180º

Calen paràmetres de càmera per a veure-ho tot (3ª persona)



# Consells d'estructuració de codi

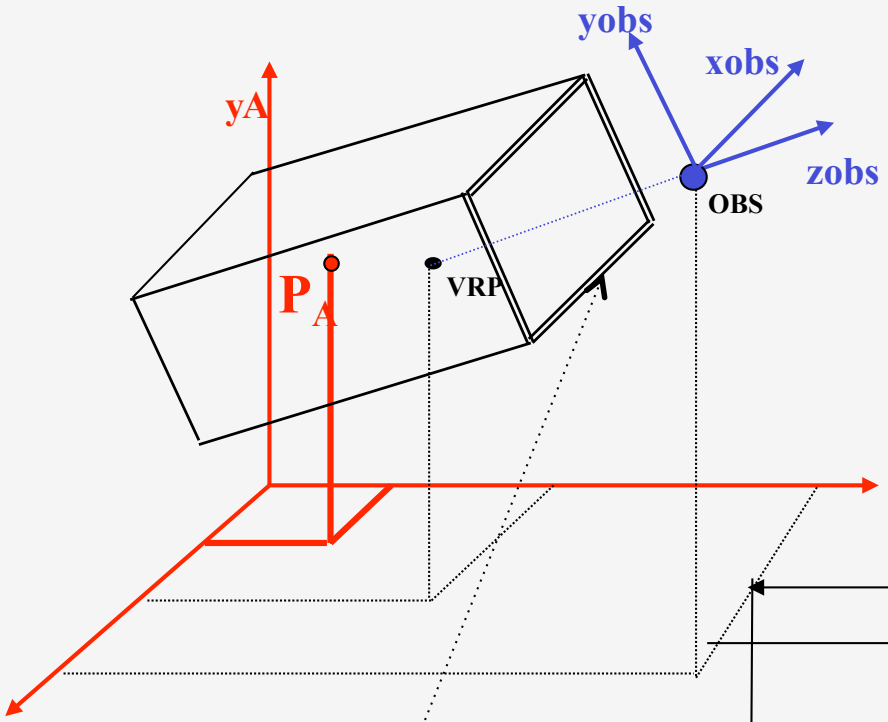
- Crear en *createBuffers()* un VAO per cada model (els dos patricios tenen el mateix VAO (caldrà carregar el model).
- Tenir mètode *CalculCapsaModel()*

  calcula la capsa d'un model i els paràmetres d'escalat, rotació i posició requerits per a calcular la TG que el posa a la posició requerida de l'escena
- Mètode *ModelTransform?()* per crear la TG de cada model.
- Mètode *IniEsfera()* en el que en funció de les dades de l'escena que tenim, inicialitzem la seva capsa i es calcula centre i radi esfera.
- Mètode *IniCàmera()* que inicialitza paràmetres càmera inicial.
- En *inicializeGL()* cridar a iniEsfera, iniCamera i viewTransform (sempre tot després d'haver creat VAOs i compilat els shaders)
- Modificar *PainGL()*

# Càmera ortogonal (exercici 3)

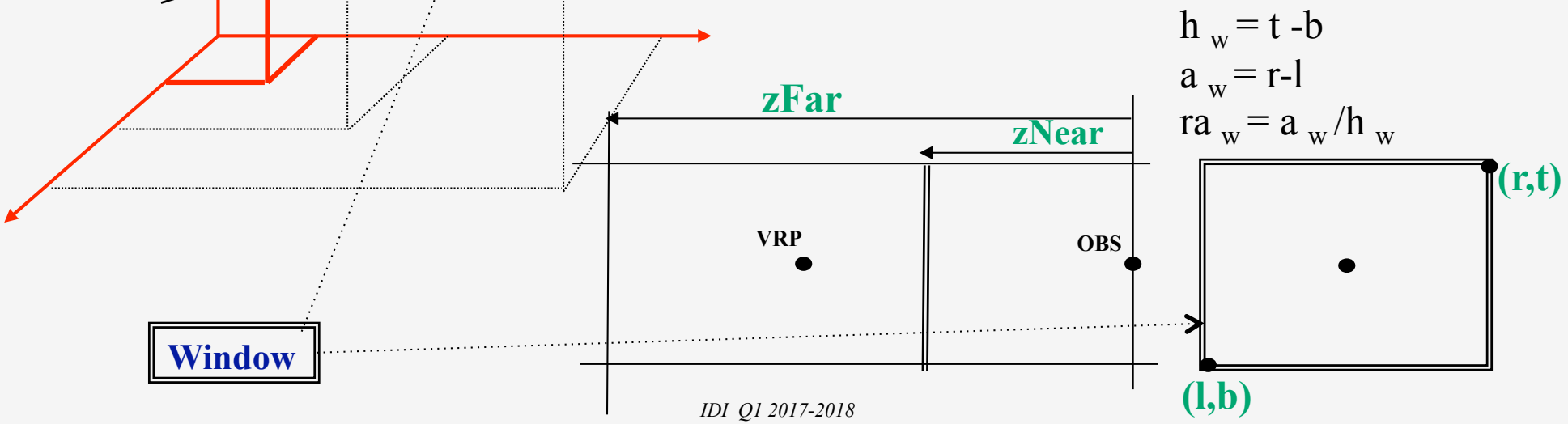
Fer òptica de càmera ortogonal *(inicialitzada per tercera persona) :*

glm::mat4 Proj = glm::ortho (left, right, bottom, top, ZNear, ZFar)




**Tipus de càmera: Axonomètrica:**

l,r,b,t, -window- , zNear, zFar



# *Resize* per a càmera ortogonal (exercici 4)

- Afegir/modificar al mètode resizeGL el necessari per a que no deformi ni retalli tampoc amb aquesta òptica *(el window)*

  En un exemple on R és el radi de l'esfera tenim:

- Window mínim requerit (centrat)= (-R,R,-R,R) => una $ra_W = 1$
- Si $ra_w \neq ra_v$ ==> deformació

  **- Si $ra_v > 1$** => cal incrementar la $ra_w$ => *modificar window*

  com $ra_w = a_w/h_w$ => podem incrementar $a_w$ o decrementar $h_w$ (és retallaria esfera!!)

  Per tant:

  $a^*_w = ra_v * h_w = ra_v*2R$ => $inc\_a = a^*_w - a_w$

  **window =** (- (R+inc_a/2), R+inc_a/2, -R, R)**= (-R $ra_v$, R $ra_v$, -R, R)**

  - raonament similar per recalcular window quan $ra_v < 1$

- Incorporar el zoom en òptica ortogonal

# Completeu l'aplicació fent:

- Amb una tecla escollir el tipus d'òptica
- Amb una tecla escollir el tipus de càmera