```
// Para tener un mismo objeto con diferentes posiciones
  void MyGLWidget::paintGL () {
    //...
    glBindVertexArray (VAO_Pat);    // Activem el VAO per a pintar el Patricio

    modelTransformPatricio();      // Pintem el Patricio 1
    glDrawArrays(GL_TRIANGLES, 0, patModel.faces().size()*3);

    modelTransformPatricio2();     // Pintem el Patricio 2
    glDrawArrays(GL_TRIANGLES, 0, patModel.faces().size()*3);
  }

// Función de viewTransform()
  void ExamGLWidget::viewTransform () {
    View = glm::translate(glm::mat4(1.f), glm::vec3(0, 0, -2*radiEsc));
    View = glm::rotate(View, -angleX, glm::vec3(1, 0, 0));
    View = glm::rotate(View, angleY, glm::vec3(0, 1, 0));
    View = glm::translate(View, -centreEsc);
    glUniformMatrix4fv (viewLoc, 1, GL_FALSE, &View[0][0]);
  }

  void ExamGLWidget::viewTransform () {
    View = glm::lookAt(obs, vrp, up);
    glUniformMatrix4fv (viewLoc, 1, GL_FALSE, &View[0][0]);
  }

// Función de projectTransform()
  void ExamGLWidget::projectTransform() {
    float fov, zn, zf;
    glm::mat4 Proj;    // Matriu de projecció
    fov = float(M_PI/3.0);
    zn = radiEsc;
    zf = radiEsc*3;
    Proj = glm::perspective(fov, rav, zn, zf);
    glUniformMatrix4fv (projLoc, 1, GL_FALSE, &Proj[0][0]);
  }

  void ExamGLWidget::projectTransform() {
    glm::mat4 Proj;    // Matriu de projecció
    Proj = glm::ortho(l, r, b, t, zn, zf);
    glUniformMatrix4fv (projLoc, 1, GL_FALSE, &Proj[0][0]);
  }
```

```
// projectTransform() con control de redimensionado
  void MyGLWidget::resizeGL (int w, int h) {
    ra = float(ample)/float(alt);
    r1 = 1.0;
    r2 = 1.0;
    if (ra < 1.0) {
      fov = 2.0*atan(tan(FOVini/2.0)/ra);
      r2 = ra;
    } else if (ra > 1.0) {

      r1 = ra;
    }
    projectTransform();
  }

  void ExamGLWidget::projectTransform () {
    glm::mat4 Proj;    // Matriu de projecció
    Proj = glm::perspective(fov, ra, zn, zf);
    glUniformMatrix4fv (projLoc, 1, GL_FALSE, &Proj[0][0]);
  }

  void ExamGLWidget::projectTransform() {
    glm::mat4 Proj;    // Matriu de projecció
    Proj = glm::ortho((-radiEsc+4)*r1, (radiEsc-4)*r1, (-radiEsc+4)/r2, (radiEsc-4)/r2, zn, zf);
    glUniformMatrix4fv (projLoc, 1, GL_FALSE, &Proj[0][0]);
  }

// Calculo de luminosidad
  mat3 normalMatrix = inverse(transpose(mat3(view * TG)));
  vec3 normSCO = normalize(normalMatrix * normal);
  vec4 vertSCO = view * TG * vec4(vertex, 1.0);
  vec3 posFocusSCO = (vec4(posFocus, 1.0)).xyz;                // SCO desde el observador
  vec3 posFocusSCO = view * (vec4(posFocus, 1.0)).xyz;         // SCA fija respecto a la escena
  vec3 posFocusSCO = view *TG*(vec4(posFocus, 1.0)).xyz;   // SCM fija respecto a un objeto
  vec3 L = normalize(posFocus - vertSCO.xyz);
  fcolor = Phong(normSCO, L, vertSCO);
```