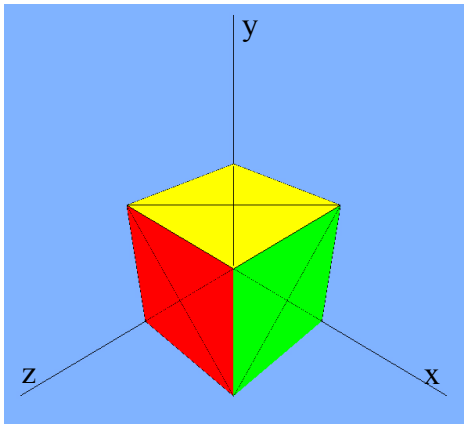
Interacció i Disseny d'Interfícies. Control parcial

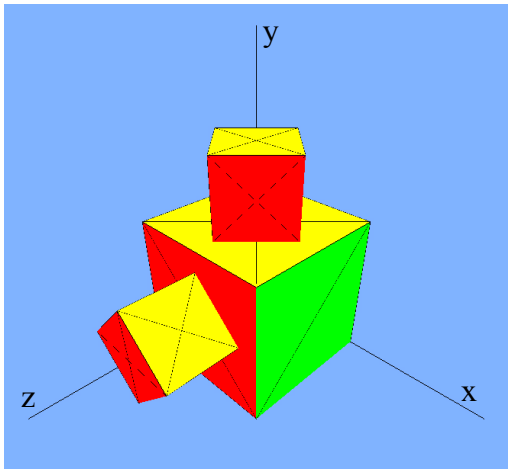
Dept. de Ciències de la Computació

E.P.S.E.V.G., 29 de març de 2019, 17:00-18:00

1) (5 punts) Disposem d'un VAO que defineix un cub de costat 2 centrat a la posició (1,1,1) amb totes les cares blaves excepte la superior que és groga, la del davant que és vermella i la de la dreta que és verda:



Volem visualitzar aquesta escena formada per tres cubs, el primer exactament igual que l'original, el segon de costat 1 girat 45º i situat sobre la cara superior del primer (centrat en el punt (1, 2.5, 1)), i el tercer també de costat 1 i girat 45º però situat davant la cara davantera del primer (centrat en el punt (1, 1, 2.5)):



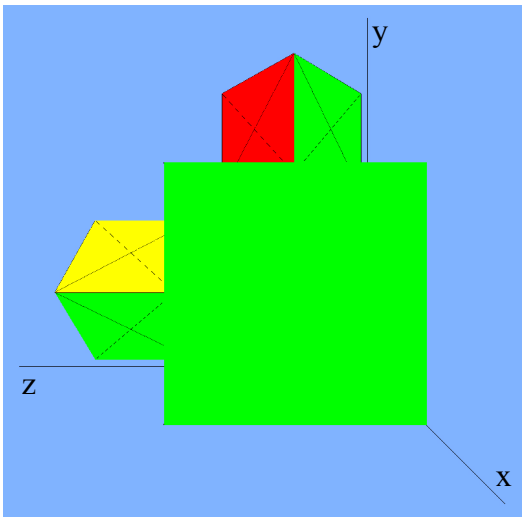
Escriu les transformacions geomètriques necessàries per passar del cub original als tres cubs finals usant les funcions de la llibrerira glm: Fes-ho programant tres cops aquesta funció `modelTransformCubX()`, on `X` és `1` o `2` o `3`:

```
void modelTransformCubX() {
  glm::mat4 TG(1.0f);

  ...

  glUniformMatrix4fv (transLoc, 1, GL_FALSE, &TG[0][0]);
}
```

2) (5 punts) Al visualitzar l'escena anterior amb una càmera perspectiva obtenim aquesta imatge:



a) Amb quins paràmetres s'ha creat la matriu de transformació de visió? Suposa que la distància entre OBS i VRP és de 3.

```
glm::mat4 View = glm::lookAt (OBS, VRP, UP);
```

b) Amb quins paràmetres s'ha creat la matriu de transformació de projecció? Suposa que el viewport té 600 píxels d'amplada i 400 píxels d'alçada.

```
glm::mat4 Proj = glm::perspective(FOV, ra, znear, zfar);
```

NOTA: No cal fer servir calculadora, pots deixar els càlculs indicats.

Proposta de solució

1)

```
void modelTransformCub1()
{
  glm::mat4 transform (1.0f);
  glUniformMatrix4fv(transLoc, 1, GL_FALSE, &transform[0][0]);
}


void modelTransformCub2()
{
  glm::mat4 transform (1.0f);
  transform = glm::translate(transform, glm::vec3(1, 2.5, 1));
  transform = glm::scale(transform, glm::vec3(0.5));
  transform = glm::rotate(transform, float(glm::radians(45.)), glm::vec3(0, 1, 0));
  transform = glm::translate(transform, glm::vec3(-1, -1, -1));
  glUniformMatrix4fv(transLoc, 1, GL_FALSE, &transform[0][0]);
}

void modelTransformCub3()
{
  glm::mat4 transform (1.0f);
  transform = glm::translate(transform, glm::vec3(1, 1, 2.5));
  transform = glm::scale(transform, glm::vec3(0.5));
  transform = glm::rotate(transform, float(glm::radians(-45.)), glm::vec3(0, 0, 1));
  transform = glm::translate(transform, glm::vec3(-1, -1, -1));
  glUniformMatrix4fv(transLoc, 1, GL_FALSE, &transform[0][0]);
}
```

2)

El VRP el situarem en el centre de la caixa mínima contenidora que té com a vèrtexs extrems pmin=(0,0,0) i pmax=(2,3,3).

El OBS està situat a la dreta del VRP seguint eix X a una distància de 3.

VRP=(1, 1.5, 1.5)
OBS=(4, 1.5, 1.5)
UP=(0, 1, 0)

La ra del window ha de ser la mateixa que la del viewport. Com que ra=600/400=1.5 és més gran que 1 no caldrà modificar el FOV inicial.

znear=2
zfar=4
ra=600/400=3/2=1.5
FOV >= 2*arctan(1.5/2)