TRABAJO FINAL REACREACION DE GLULOOKAT

CURSO : Diseño asistido por computador
DOCENTE : Ing. Omar Latorre Vilca

Integrantes

Mireles Bernabé, Víctor Fernando

2017-119066

Ramos Vásquez, Carlos Junior

2017-119062

Tabla de contenido

FUNCIONAMIENTO

GLULOOKAT()

Permite definir de forma específica donde se va a situar la cámara, hacia dónde mirará ésta y cuál será el orden de los ejes de coordenadas.

Según sea el valor del vector de vista hacia arriba el objeto que se visualizará tendrá un aspecto diferente.En OpenGL lo tenemos fácil con:

```
gluLookAt(eyeX, eyeY, eyeZ, atX, atY, atZ, upX, upY, upZ);
```

GLULOOKAT()

gluLookAt(eyeX, eyeY, eyeZ, atX, atY, atZ, upX, upY, upZ);

En cuanto a los parámetros que demanda la función son los siguientes:

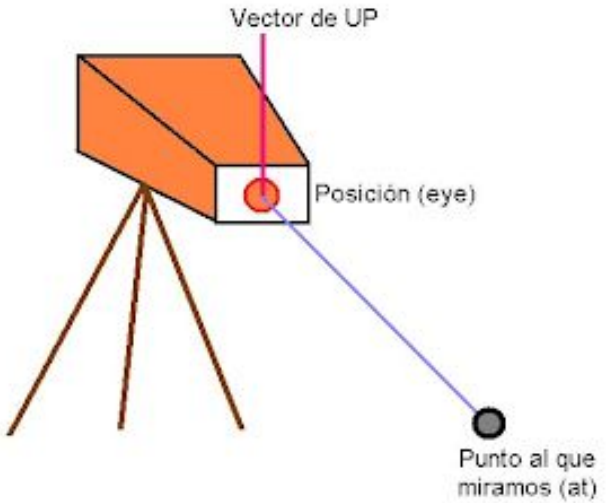
Coordenadas del "**eye**". Es literalmente la posición XYZ dónde colocar la cámara dentro del mundo.

Coordenadas del "**at**". Es el valor XYZ del punto al que queremos que mire la cámara. Un punto del mundo obviamente.

Coordenadas del vector "**up**". Con él regularemos la orientación de la cámara. Este ha de ser el vector que "mira hacia arriba" si entendemos que el vector que mira hacia adelante es el que va del "eye" hasta el "at". Variando el "up" variamos la orientación.

GLULOOKAT()

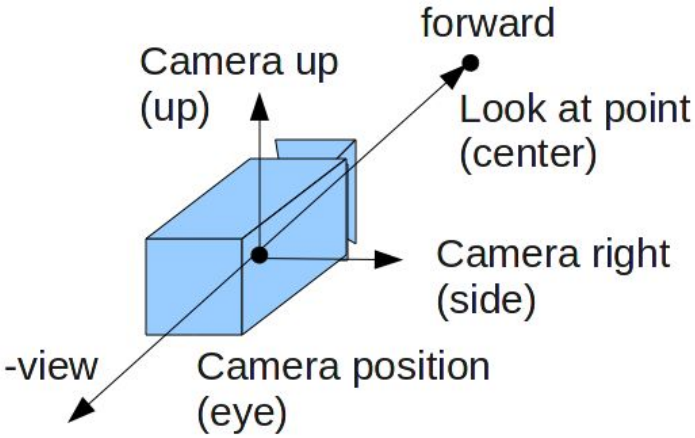
gluLookAt(eyeX, eyeY, eyeZ, atX, atY, atZ, upX, upY, upZ);

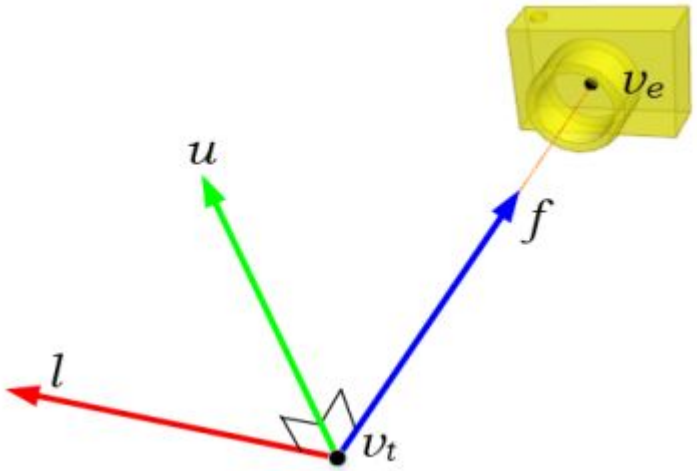


- gluLookAt altera la matriz GL_MODELVIEW y esta es la que guarda el estado de todo nuestro sistema en cuanto a transformaciones.

EXPLICACIÓN MATEMÁTICA

GRAFICA


forward
Camera up
(up)
Look at point
(center)
Camera right
(side)
-view
Camera position
(eye)


v_e
u
f
l
v_t

Vectores - f

En primer lugar, se necesita hallar el vector " f " , segun la documentacion de opengl es necesario restar el vector donde apunta la cámara y el vector eye, entonces quedaría de la siguiente manera :

$$F = \begin{matrix} centerX - eyeX \\ centerY - eyeY \\ centerZ - eyeZ \end{matrix}$$

Después es necesario normalizar :

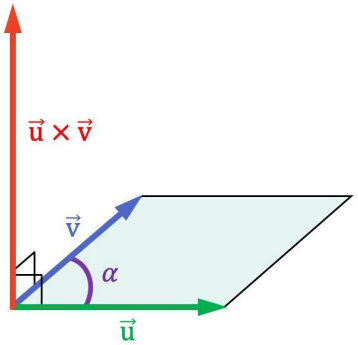
$$|F|$$

Por último el vector se obtiene de la siguiente manera:

$$f = \frac{F}{|F|}$$

Vectores - l

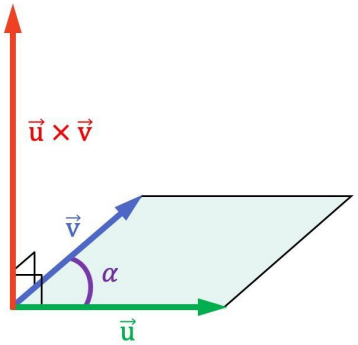
Para poder hallar el vector " l ", se utiliza el producto vectorial entre los vectores up y f debido a que ambos vectores son ortogonales, de esta manera se encuentra el nuevo vector.



$$l_x = f_y * up_z - f_z * up_y$$
$$l_y = f_z * up_x - f_x * up_z$$
$$l_x = f_x * up_y - f_y * up_x$$

Vectores - u

Para poder hallar el vector " u ", se utiliza el producto vectorial entre los vectores l y f debido a que ambos vectores son ortogonales, de esta manera se encuentra el nuevo vector.



$$u_x = l_y * f_z - l_z * f_y$$

$$u_y = l_z * f_x - l_x * f_z$$

$$u_x = l_x * f_y - l_y * f_x$$

Por último se coloca todos los valores de los vectores hallados en un matriz :

$$M = \begin{bmatrix} l_x & l_y & l_z & 0 \\ u_x & u_y & u_z & 0 \\ -f_x & -f_y & -f_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

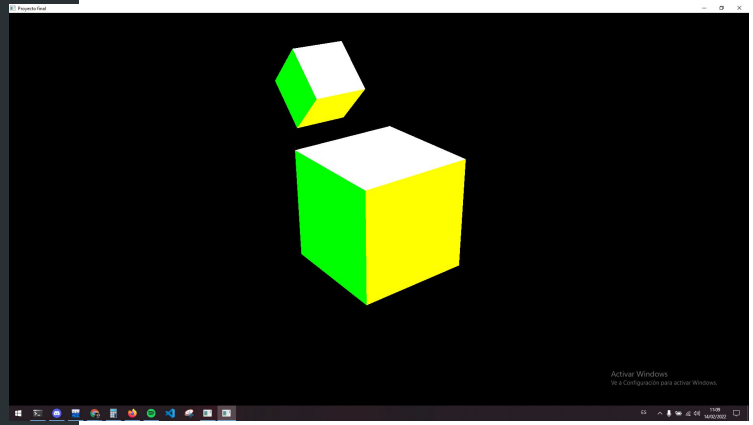
Por último se multiplica la matriz con la matriz de opengl “MODELVIEW” y también se aplica una traslación :

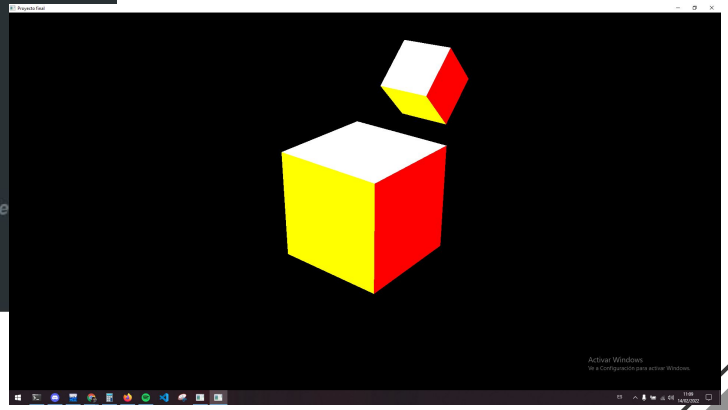
```
glMultMatrixf(M);
glTranslated(-eyex, -eyey, -eyez);
```

IMPLEMENTACIÓN EN OPENGL

main.cpp

```cpp
#include <GL/freeglut.h>
#define _USE_MATH_DEFINES
#include <math.h>

class Renderer {

public:
  float t;

public:
  Renderer() : t(0.0) {}

public:

  void resize(int w, int h) {
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective (30.0, (float)w/(float)h, 2.0, 20.0);
  }

  void display() {
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    double rad = M_PI / 180.0f * t;

    //Funcion de la libreria

//    gluLookAt(10.0*cos(rad), 5.0 , 10.0*sin(rad), // eye
//    gluLookAt(10.0, 10.0 , 10.0, // eye
//              0.0, 0.0, 0.0, // look at
//              0.0, 1.0, 0.0); // up

```





GRACIAS