“Año del Bicentenario del Perú: 200 años de Independencia”

UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN

- UNJBG -

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA EN INFORMÁTICA Y SISTEMAS

ASIGNATURA : DISEÑO ASISTIDO POR COMPUTADOR

TEMA : **TRABAJO FINAL**

DOCENTE : Dr. Omar Latorre Vilca

AÑO : CUARTO AÑO

INTEGRANTES DE GRUPO:

QUISPE OCHOA, JOHAN DAVID 2018-119016

YANAPA CHICALLA, ALEJANDRO DANIEL 2018-119003

TACNA – PERÚ

2020

Función gluLookAt():

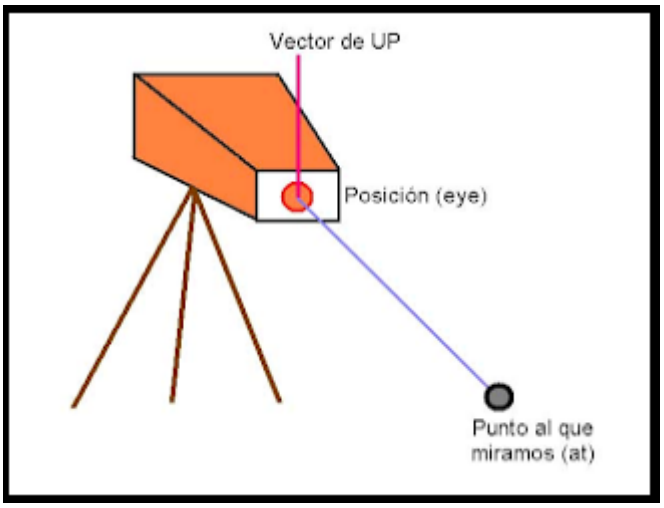
Esta función propia de la librería GL/glut , la cual nos permite seleccionar dónde se colocará la cámara y hacia dónde apuntará.

Esta función lleva la siguiente sintaxis:

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

Los parámetros eye son las coordenadas donde se sitúa la cámara y los parámetros at son las coordenadas hacia donde apuntará ésta.

Adicionalmente se colocan los parámetros up de tipo vector que significan la orientación de la cámara.



PROGRAMA

Nuestro programa se encarga de aplicar nuestra propia función LookAt y también la función gluLookAt para poder compararlas.

Cuenta con una pequeña interfaz para que el usuario pueda cambiar las coordenadas de los diferentes puntos “Ve” ,”Vt” y “Up”; también muestra las matrices de traslación y rotación en pantalla así como el punto Vt.

1. Librerías:

```
#include <iostream>
#include <GL/glut.h>
#include<GL/freeglut.h>
#include <cmath>
```

<iostream> Lo usamos para poder convertir las coordenadas de “Ve”, ”Vt” , “Up” y los valores de las matrices de traslación y rotación en char para poder mostrarlos en pantalla.

<GL/glut.h> <GL/freeglut.h> son la API de OpenGL.

<cmath> Para poder usar raíz cuadrada que se usará al calcular el módulo de vectores.

2. Variables globales:

a. Coordenadas de puntos “Ve” ,”Vt” y “Up”:

```
GLfloat vex=0.0,vey=1.0,vez=5.0,vtx=0.0,vty=0.0,vtz=0.0,upx=0.0,upy=1.0,upz=0.0;
```

Estas son definidas como GLfloat para luego poder cargar la matriz de rotación.

b. Array matriz_R:

```
GLfloat matriz_R[16];
```

Ya que la función para cargar matrices “glLoadMatrixf()” acepta array de 16 con valores tipo GLfloat creamos esta matriz para aplicar la rotación.

c. Variables del menú e interacción del usuario:

```
bool cambio=true,menuuso=false,matriz_v=true,p_vista=true;
int menux=1,menuy=1;
```

Estas variables nos ayudarán a que las opciones de la interfaz funcionen correctamente.

3. Estructuras del programa:
   a. struct Vector:
   Esta estructura se crea para poder manipular fácilmente las coordenadas de los puntos “Ve” ,”Vt” y “Up” así como también los vectores “f”,”l” y “up”.
   Tiene como atributo x,y,z que son variables tipo GLfloat para que encajen con los datos del array matriz_R.

```
struct Vector
{
    GLfloat x,y,z;
    Vector()
    {
        x=y=z=0;
    }
    Vector(GLfloat x2, GLfloat y2, GLfloat z2)
    {
        x=x2;
        y=y2;
        z=z2;
    }
};
```

Vector tiene dos constructores:

- Vector(): pondrá los valores de x,y,z en 0 si no se le agrega ningún parámetro.
- Vector(GLfloat x2, GLfloat y2, GLfloat z2) : pondrá a los valores de x,y,z los valores que le pasamos como parámetros.

4. Funciones del programa:
   a. Funciones asociadas a la función LookAt:
      - vectorial:

```
Vector vectorial(Vector a, Vector b)
{
    return Vector(a.y*b.z-a.z*b.y,a.z*b.x-a.x*b.z,a.x*b.y-a.y*b.x);
}
```

Tendrá como parámetros dos vectores, y devolverá un vector que será el producto de los vectores recibidos como parámetros. Nos servirá para obtener los vectores de "l" y "up".

$$\vec{l} = \frac{\vec{up} \times \vec{f}}{\left\| \vec{up} \times \vec{f} \right\|} \qquad \vec{u} = \vec{f} \times \vec{l}$$

- mod:

```
void mod(Vector &vector)
{
    GLfloat div=sqrt(vector.x*vector.x+vector.y*vector.y+vector.z*vector.z);
    vector.x=vector.x/div;
    vector.y=vector.y/div;
    vector.z=vector.z/div;
}
```

Modificar los atributos del vector que demos como parámetro aplicando el módulo y luego dividiéndolo entre los valores de x,y,z actuales. Nos servirá para obtener los vectores de "f" y "l".

$$\vec{f} = \frac{\vec{v_e} - \vec{v_t}}{\left\| \vec{v_e} - \vec{v_t} \right\|} \qquad \vec{l} = \frac{\vec{up} \times \vec{f}}{\left\| \vec{up} \times \vec{f} \right\|}$$

b. Funciones asociadas a la interfaz:

- void tiempo(int param):
  Nos ayudará a generar el bucle del programa.
- void letra(float x, float y,float z, char *string,int a):
  Nos ayudará a dibujar caracteres en pantalla.
- int menu_color(int x,int y):
  Nos ayudará a decidir qué color debe llevar los caracteres que representan las coordenadas de los puntos "Ve" ,"Vt" y "Up" al estar modificandolos.
- void L_menu(int xm, int ym, float m,float X,float Y,bool T):
  Nos ayudará a convertir los Glfloat a tipo char también a darles una posición para mostrarlos en pantalla.
- void ControlRaton(int button, int state, int x, int y):
  Permitirá al usuario el uso del ratón:
  Click izquierdo para cambiar entre el uso de la función gluLookAt y la función LookAt.
  Click derecho para mostrar o no mostrar las matrices de traslación y rotación en pantalla.

- void ControlTeclado(unsigned char key,int x,int y):
  Permitirá al usuario el uso del teclado:
  Tecla "e" para mostrar o no una esfera que representa el punto de vista "Vt".
  Tecla "space" en el menú de modificación de coordenadas para seleccionar un eje o salir de este.
  Teclas "a,s,d,w" para moverse en el menú de modificación de coordenadas y también para cambiar los valores de un eje.
- void menu():
  Permitirá dibujar el menú de modificación de coordenadas, mostrar las matrices de transformación y rotación, también nos mostrará un texto si usamos gluLookAt o LookAt.
- void dibujar():
  Permitirá dibujar unos objetos en pantalla que serán una tetera, un icosaedro y un cubo para visualizar mejor el uso de las funciones LookAt y gluLookAt.

5. Función LookAt:

```
void LookAt(GLfloat veX, GLfloat veY, GLfloat v
GLfloat vtY, GLfloat vtZ, GLfloat upX, GLfloat
{
    memset(matriz_R, 0, sizeof(matriz_R));
    matriz_R[15] = 1;
    Vector f(vtX - veX, vtY - veY, vtZ - veZ);
    mod(f);
    Vector up(upX, upY, upZ);
    Vector l = vectorial(f, up);
    mod(l);
    up = vectorial(l, f);
    matriz_R[0] = l.x;
    matriz_R[4] = l.y;
    matriz_R[8] = l.z;
    matriz_R[1] = up.x;
    matriz_R[5] = up.y;
    matriz_R[9] = up.z;
    matriz_R[2] = -f.x;
    matriz_R[6] = -f.y;
    matriz_R[10] = -f.z;
    glLoadMatrixf(matriz_R);
    glTranslated(-veX, -veY, -veZ);
}
```

```
void LookAt(GLfloat veX, GLfloat veY, GLfloat veZ, GLfloat vtX,
GLfloat vtY, GLfloat vtZ, GLfloat upX, GLfloat upY, GLfloat upZ)
{
```

Nuestra función LookAt tiene como parámetros las coordenadas de la posición de la cámara (ve), las coordenadas a donde apunta la cámara (vt), y el vector de orientación de la cámara (up).
Estos valores son de tipo GLfloat para poder cargar la matriz con la función glLoadMatrixf();

```
memset(matriz_R, 0, sizeof(matriz_R));
matriz_R[15] = 1;
```

Esta parte de la función con memset() reseteamos todos los valores del array matriz_R a 0 y damos valor a la posición 15 que sea 1 como en la matriz de rotación que se nos entregó en la actividad:

$$R = \begin{bmatrix} x_l & y_l & z_l & 0 \\ x_u & y_u & z_u & 0 \\ x_f & y_f & z_f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
Vector f(vtX - veX, vtY - veY, vtZ - veZ);
mod(f);
```

Luego creamos el vector f con el constructor de Vector y con la ayuda de los vectores “ve” y “vt” para aplicarle la función mod que se explicó anteriormente también con las indicaciones que se encuentran en la actividad:

$$\vec{f} = \frac{\vec{v_e} - \vec{v_t}}{\|\vec{v_e} - \vec{v_t}\|}$$

```
Vector up(upX, upY, upZ);
```

Se crea el vector “up” para luego poder operarlo con el vector “f” y obtener el vector “l”.

$$\vec{l} = \frac{\vec{up} \times \vec{f}}{\left\|\vec{up} \times \vec{f}\right\|}$$

```
Vector l = vectorial(f, up);
mod(l);
```

Para obtener el vector “l” sacamos el producto de “f” y “up” con la función vectorial explicada anteriormente y luego aplicando la función mod.

$$\vec{l} = \frac{\vec{up} \times \vec{f}}{\left\|\vec{up} \times \vec{f}\right\|}$$

```
up = vectorial(l, f);
```

Luego nuestro vector “up” lo reemplazamos con el producto de “l” y “f” como esta en la actividad:

$$\vec{u} = \vec{f} \times \vec{l}$$

```
matriz_R[0] = l.x;
matriz_R[4] = l.y;
matriz_R[8] = l.z;
matriz_R[1] = up.x;
matriz_R[5] = up.y;
matriz_R[9] = up.z;
matriz_R[2] = -f.x;
matriz_R[6] = -f.y;
matriz_R[10] = -f.z;
```

Definimos los diferentes valores en el array matriz_R teniendo en cuenta que es la matriz de rotación:

$$R = \begin{bmatrix} x_l & y_l & z_l & 0 \\ x_u & y_u & z_u & 0 \\ x_f & y_f & z_f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
glLoadMatrixf(matriz_R);
```

Cargamos la matriz de rotacion con la funcion glLoadMatrixf

```
glTranslated(-veX, -veY, -veZ);
```

Finalmente se aplica la traslación:

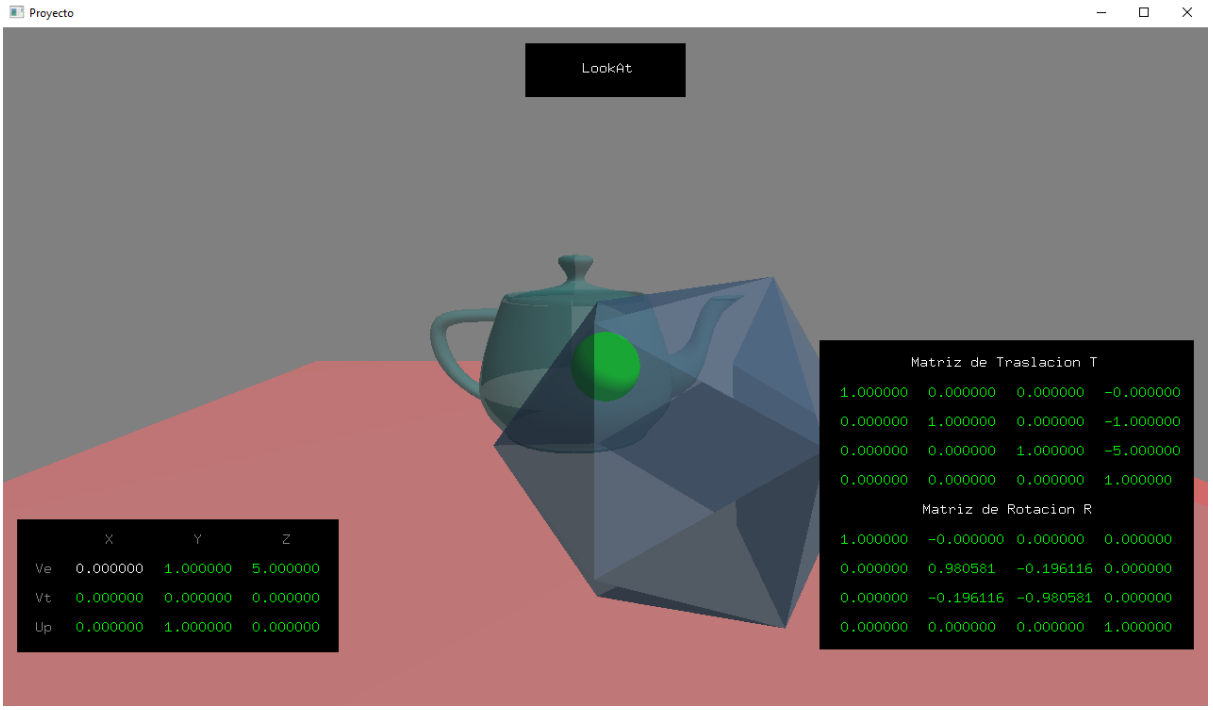
$$T = \begin{bmatrix} 1 & 0 & 0 & -x_{ve} \\ 0 & 1 & 0 & -y_{ve} \\ 0 & 0 & 1 & -z_{ve} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Llamado de la función LookAt y gluLookAt en la función display():

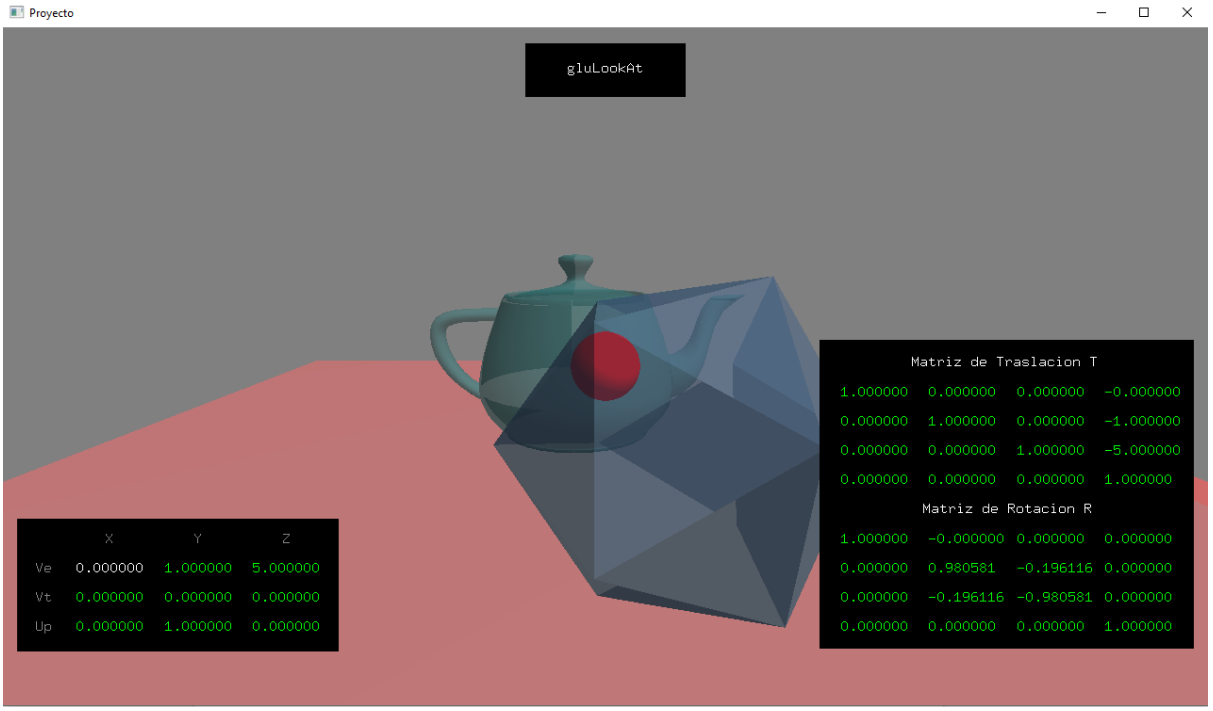
```
if(cambio){
    LookAt (vex, vey, vez, vtx, vty, vtz, upx, upy, upz);
    if(p_vista){
        glPushMatrix();
        glTranslatef(vtx,vty,vtz);
        glScalef(1,1,1);
        glColor4f(0,1,0,1);
        glutSolidSphere(0.3,50,50);
        glPopMatrix();
    }
}
else{
    gluLookAt (vex, vey, vez, vtx, vty, vtz, upx, upy, upz);
    if(p_vista){
        glPushMatrix();
        glTranslatef(vtx,vty,vtz);
        glScalef(1,1,1);
        glColor4f(1,0,0,1);
        glutSolidSphere(0.3,50,50);
        glPopMatrix();
    }
}
```

En donde la variable booleana “cambio” nos permite pasar del uso de la función LookAt a gluLookAt y viceversa.

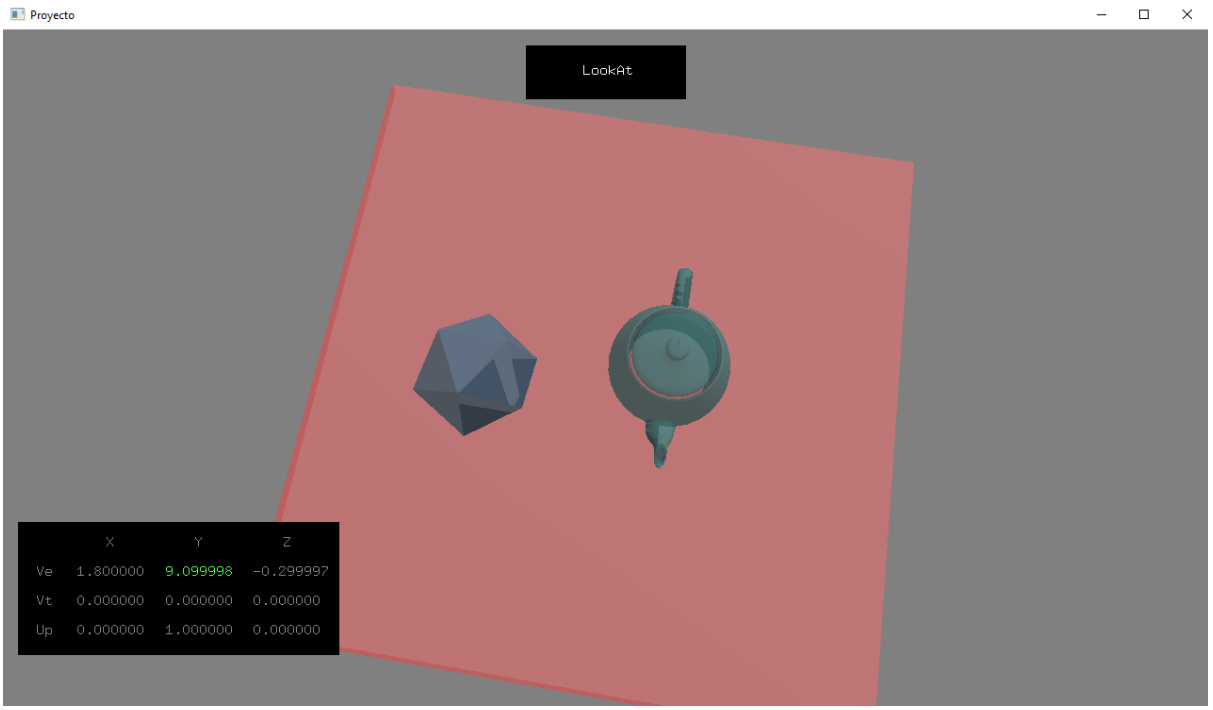
6. Programa corriendo:



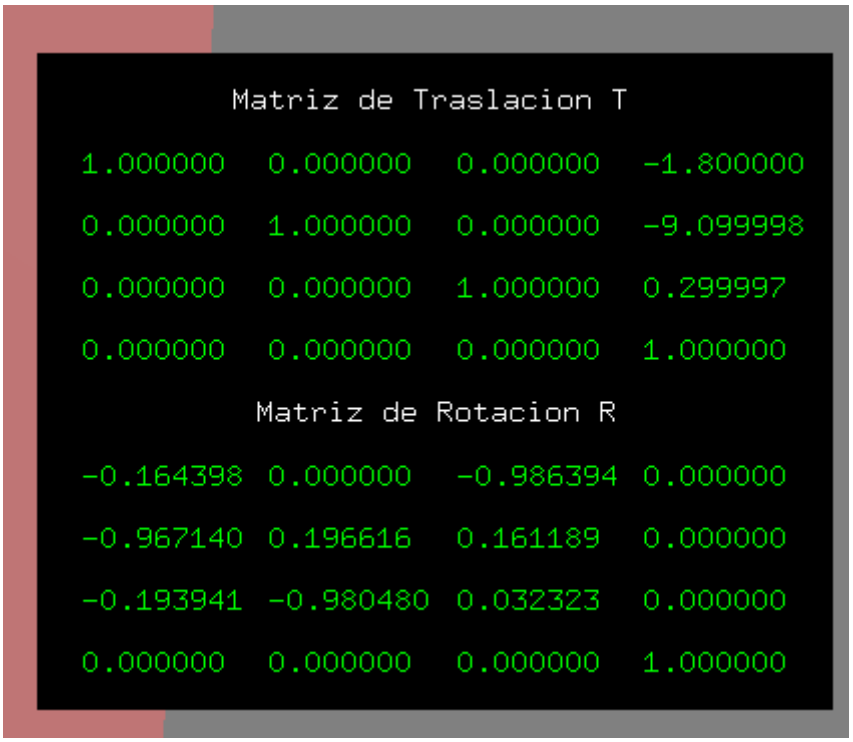
Vista usando LookAt



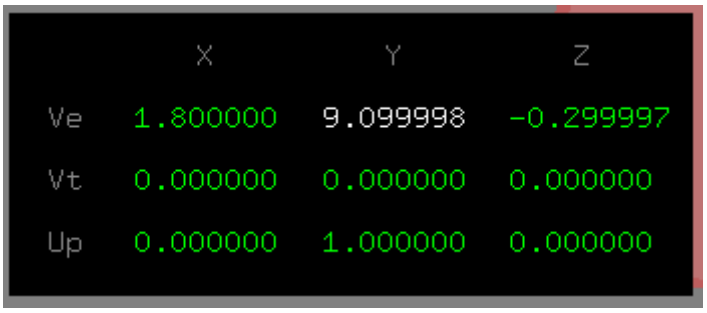
Vista usando gluLookAt



Vista usando LookAt (coordenadas de Ve modificadas)



Matrices de Traslacion y Rotacion mostradas en pantalla



Menú para modificar coordenadas de “Ve”, “Vt” y ”Up”