Máster en Programación de Videojuegos
Fundamentos Matemáticos
Profesor José María Benito

Práctica 2: Transformaciones

Introducción

El objetivo de esta práctica es familiarizarse con los aspectos más importantes de las operaciones de transformación en 3D.

Notas Generales

Esta práctica está diseñada para ser realizada en clase y no va a ser corregida o calificada. Sin embargo debes intentar completar todos los puntos y aprovechar el tiempo en clase. En caso contrario será motivo para suspender la asignatura.

Creación del proyecto en Visual Studio

1. En Visual Studio:
   1.1. Crea un proyecto de consola Win32 vacío, dale un nombre estilo “Transformations”.
   1.2. Añádele los ficheros entregados con la práctica en el directorio del proyecto.
      - main.cpp, GLInclude.h,
   -
   1.3.

Ejercicios

1. escribe una función en OpenGL para dibujar linea a partir de sus puntos:

```
void drawLine(LINE line, COLOUR color = grey, bool doDrawDots = 
false) {


    glColor3f(color.r,color.g,color.b);
```

```
    // usa GL_LINE_STRIP en modo inmediato (glBegin/glEnd)
    // enviar puntos a OpenGL usando glVertex3f
}
```

2. función dibujar axis: usando la función anterior dibuja tres líneas de color rojo (X) verde (Y) y azul(Z)
3. vectores3D: escribe las siguiente funciones:

```
VECTOR3D Add(VECTOR3D a, VECTOR3D b);

VECTOR3D Substract(VECTOR3D a, VECTOR3D b);

VECTOR3D Multiply(VECTOR3D a, VECTOR3D b);

VECTOR3D MultiplyWithScalar(float scalar, VECTOR3D a);

double Magnitude(VECTOR3D a);

VECTOR3D Normalize(VECTOR3D a);

VECTOR3D CrossProduct(VECTOR3D a, VECTOR3D b);

double DotProduct(VECTOR3D a, VECTOR3D b);
```

4. Matrices3D: escribe las siguientes funciones:

```
MATRIX3 Transpose(MATRIX3 m);

VECTOR3D Transform (MATRIX3 m, VECTOR3D a);
```

puedes utilizar la función anterior y DotProduct para escribir esta función más rápidamente.

```
MATRIX4 InverseOrthogonalMatrix(MATRIX3 A, VECTOR3D t);
```

recuerda que puedes realizar esta operación con la siguiente fórmula:

$A^{-1} = A^{T}b - A^{T}t$ (sii A ortogonal)

y que en una matrix4 puedes rotar y trasladar a la vez si utilizas la cuarta columna para la traslación. Utiliza las funciones anteriores.

5. lookAt: vamos a sustituir gluLookAt. Escribe la siguiente función:

```
MATRIX4 lookAt( VECTOR3D eyePosition, VECTOR3D target, VECTOR3D
upVector )
```

utiliza la siguiente fórmula:

5.1. Construye el vector Z (forward) utilizando target y eyePosition.

5.2. Construye el vector X (side) utilizando el producto vectorial (cross) de forward con upVector.

5.3. Construye el vector Y (up) utilizando el producto vectorial (cross) de side y forward.

5.4. Genera una matriz de rotación con los vectores anteriores como columnas. (recuerda que OpenGL apunta a -Z)

5.5. Invierte la matriz ortogonal que has construido mediante InverseOrthogonalMatrix con la posición de cámara (eyePosition) como desplazamiento. Devuelve la matriz obtenida.

5.6. En main.cpp sustituye gluLookAt() por lo siguiente:

```
    MATRIX4 lookAtMatrix = lookAt(camera.position, target,
camera.up);
    glLoadMatrixf(lookAtMatrix.m);
```