Asignatura	Datos del alumno	Fecha
Informática Gráfica y Visualización	Apellidos: Bocanegra Capera	19/mayo/2024
	Nombre: Oscar David	

Actividades

Laboratorio #1: Introducción a OpenGL

Preparación para el laboratorio

Antes de acudir al laboratorio deberás haber instalado OpenGL. Para ejercitar el aprendizaje autodidacta no vamos a detallar aquí el proceso de instalación, sino que el alumno deberá buscar recursos en Internet que le ayuden a realizar la instalación. A continuación se describen algunas pautas.

Los usuarios de Windows deben instalar la librería GLUT.

Aquí se describe cómo hacerlo:
http://web.cs.wpi.edu/~gogo/courses/mingw/

En clase usaremos el comando gcc de las GCC para compilar. Si quieres pueden instalar otros editores de texto adicionales con los que te sientas cómodo Xcode, Dev-C++, NetBeans, etc, pero deberás conocerlo: el profesor usará gcc y no resuelve problemas con IDEs.

Usa el foro «Pregúntale al profesor de la asignatura» para resolver problemas de configuración antes del laboratorio.

Tiempo asignado para el laboratorio 2 horas. El trabajo realizado se evalúa al acabar el laboratorio y la entrega se realizará en la fecha indicada en el sistema de entrega de actividades.

Asignatura	Datos del alumno	Fecha
Informática Gráfica y Visualización	Apellidos: Bocanegra Capera	19/mayo/2024
	Nombre: Oscar David	

Descripción del laboratorio

Durante el laboratorio vamos a familiarizarnos con las primitivas de dibujo de OpenGL. Para ello el estudiante debe de hacer un programa OpenGL que dibuje un reloj analógico, incluidas las marcas de las horas y las manillas. El reloj debe ser estático, ya que la animación no se trata hasta el Tema 12.

Entrega del laboratorio

Una vez acabado el trabajo, adjunta:

1. Un fichero `solution.c` con la implementación de tu programa.
2. Un fichero de respuestas que contenga:
   a. Un *screenshot* del programa en ejecución.
   b. La respuesta a estas preguntas: ¿Funciona bien tu programa? ¿Qué fallos existen?

Nota: Solo se acepta ficheros editables (.doc, .docx, .odf, .rtf). No se aceptan imágenes escaneadas del ejercicio ni el formato .pdf.

Asignatura	Datos del alumno	Fecha
Informática Gráfica y Visualización	Apellidos: Bocanegra Capera	19/mayo/2024
	Nombre: Oscar David	

Solución

El primer paso para poder desarrollar este laboratorio fue importar las respectivas librerías y el ide que en este caso use visual estudio code, para el caso de esta actividad la desarrolle en Python, por ende, en la siguiente imagen se podrá observar cuales fueron las importaciones, además de esto puedo decir que

- **sys:** Permite acceder a funcionalidades del sistema, como los pueden ser los argumentos de línea de comandos.
- **math:** Proporciona funciones matemáticas como el seno, coseno y conversión de grados a radianes, esenciales para posicionar elementos circulares como el reloj.
- **OpenGL.GL:** Contiene las funciones básicas de dibujo
- **OpenGL.GLUT:** Proporciona utilidades de la *OpenGL Utility Toolkit* como ventanas, eventos, entrada de teclado y fuentes de texto.
- **OpenGL.GLU:** Ofrece funciones de utilidad como gluOrtho2D, que se usa para configurar una proyección ortográfica.

Y adicionalmente a las librerías definí una constante (FONT) la cual se encarga de representar el tipo de letra para los números que se verán en el reloj

```
import sys
import math
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import OpenGL.GLUT as GLUT

# Definir la fuente para los números
FONT = GLUT.GLUT_BITMAP_HELVETICA_18
```

Función draw_circle

```python
# Definir la fuente para los números
FONT = GLUT.GLUT_BITMAP_HELVETICA_18

def draw_circle(radius):
    """
    Dibuja un círculo usando líneas para representar el borde del reloj.
    :param radius: Radio del círculo
    """
    glBegin(GL_LINE_LOOP)
    for i in range(360):
        angle = math.radians(i)
        x = radius * math.cos(angle)
        y = radius * math.sin(angle)
        glVertex2f(x, y)
    glEnd()
```

Con el uso de esta función lo que vamos a hacer es la creación del circulo del reloj, en donde el parámetro radius lo que hace es definir el radio del circulo y el bucle lo que hace es indicar que se va a dibujar una línea cerrada y este acaba en donde indicamos el glEnd()

Función draw_marks

```python
def draw_marks():
    """
    Dibuja las 12 marcas de las horas alrededor del reloj y los números de las horas.
    El 12 debe estar arriba, 3 a la derecha, 6 abajo, 9 a la izquierda.
    """
    for i in range(12):
        # Mueve el 12 dos posiciones a la izquierda
        angle = math.radians((i + 1) * -30 + 90)
        x1 = 0.9 * math.cos(angle)
        y1 = 0.9 * math.sin(angle)
        x2 = math.cos(angle)
        y2 = math.sin(angle)
        glLineWidth(3)
        glBegin(GL_LINES)
        glVertex2f(x1, y1)
        glVertex2f(x2, y2)
        glEnd()
        # Dibujar el número de la hora, centrado
        glColor3f(0, 0, 0)
        num_radius = 0.78
        num_x = num_radius * math.cos(angle)
        num_y = num_radius * math.sin(angle)
        offset_x = -0.03 * len(str(i+1))
        offset_y = -0.04
        glRasterPos2f(num_x + offset_x, num_y + offset_y)
        hour = str(i+1)
        for ch in hour:
            glutBitmapCharacter(FONT, ord(ch))
```

Básicamente, lo que hace esta función es colocar las **marcas horarias** y los **números del 1 al 12** alrededor del reloj, o mejor dicho, alrededor del **círculo** que representa la base del reloj. Esto es fundamental para darle al dibujo el **aspecto de reloj analógico**, que es precisamente el objetivo de esta actividad de visualización gráfica.

La función inicia con un bucle for que recorre las 12 posiciones del reloj lo cual haría referencia a una por cada hora y puedo decir que en cada iteración lo que se hace es:

1. Se calcula el ángulo correcto para ubicar la marca correspondiente, asegurando que el número **12 aparezca en la parte superior**, el **3 a la derecha**, el **6 abajo** y el **9 a la izquierda**, exactamente como en un reloj tradicional.
2. Los ángulos se ajustan para que el recorrido sea en **sentido horario**, lo que refuerza el realismo del diseño.
3. Se dibuja una **línea corta que representa la marca de la hora**, con un grosor de 3 píxeles, que va desde un punto cercano al borde del círculo hacia afuera.
4. Luego, se calcula la **posición exacta para ubicar el número** que representa la hora

Función draw_hand

```python
def draw_hand(length, angle_deg, width=2):
    """
    Dibuja una manecilla del reloj.
    :param length: Longitud de la manecilla (relativa al radio)
    :param angle_deg: Ángulo en grados desde las 12 en sentido horario
    :param width: Grosor de la línea
    """
    angle = math.radians(angle_deg)
    glLineWidth(width)
    glBegin(GL_LINES)
    glVertex2f(0, 0)
    glVertex2f(length * math.cos(angle), length * math.sin(angle))
    glEnd()
```

Esta función es mucho más corta pero lo que hace esta es trazar una línea que inicia desde el centro del reloj, esto quiere decir, el punto (0,0) y se extiende hacia afuera en una determinada dirección para lograr que esta línea represente una manecilla del reloj.

Así que gracias a esta función, el reloj puede mostrar gráficamente la hora, el minuto y el segundo de forma diferenciada.

Función display

```python
def display():
    """
    Función de dibujo principal. Limpia la pantalla y dibuja el reloj con sus manecillas.
    """
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(0, 0, 0)
    draw_circle(1.0)
    draw_marks()
    # Manecillas (estáticas, ejemplo: 10:10:30)
    glColor3f(0.2, 0.2, 0.8)  # Hora
    draw_hand(0.5, 120, 6)
    glColor3f(0.2, 0.8, 0.2)  # Minuto
    draw_hand(0.8, 60, 4)
    glColor3f(0.8, 0.2, 0.2)  # Segundo
    draw_hand(0.9, 180, 2)
    glutSwapBuffers()
```

Acá como podemos observar en el esta es la función encargada de dibujar absolutamente todo lo que se ve en la pantalla emergente, que en este caso va a ser la imagen de un reloj. Por ende, puedo decir que esta función arma el reloj paso a paso cada vez que la ventana necesita actualizarse.

Y cuando se abre la ventana del programa, o cuando se necesita redibujar lo que hay (por ejemplo, al redimensionar o refrescar), OpenGL llama automáticamente a esta función display() para que vuelva a dibujar todo el contenido visual.

Función reshape

```
def reshape(w, h):
    """
    Ajusta la vista cuando la ventana cambia de tamaño.
    """
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(-1.2, 1.2, -1.2, 1.2)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
```

De esta función puedo decir que ajusta la forma en que se ve el reloj en la pantalla cuando el tamaño de la ventana cambia (por ejemplo, si se agranda o se hace más pequeña).

Así que, sin esta función, al redimensionar la ventana, el reloj podría deformarse o simplemente podría salir del área visible. Entonces, reshape() se asegura de que el reloj siga viéndose correctamente, bien centrado y con las proporciones correctas, sin importar el nuevo tamaño de la ventana.

Asignatura	Datos del alumno	Fecha
Informática Gráfica y Visualización	Apellidos: Bocanegra Capera	19/mayo/2024
	Nombre: Oscar David	

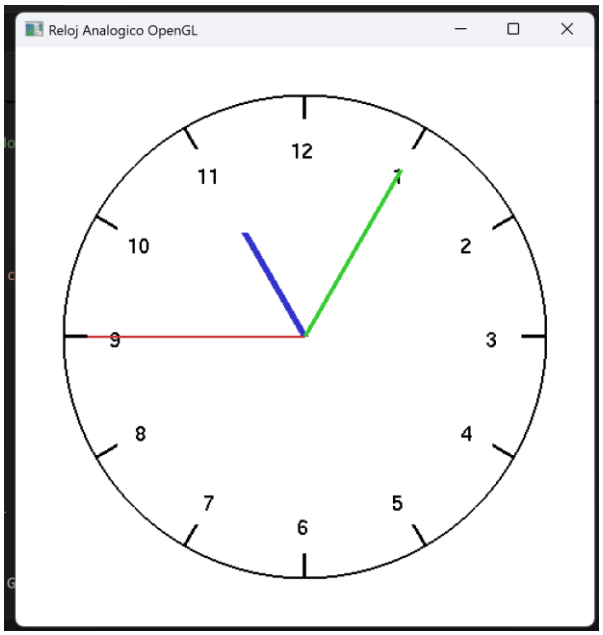
Función main

```python
def main():
    """
    Función principal: inicializa GLUT y ejecuta el bucle principal.
    """
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)
    glutInitWindowSize(500, 500)
    glutCreateWindow(b'Reloj Analogico OpenGL')
    glClearColor(1, 1, 1, 1)
    glutDisplayFunc(display)
    glutReshapeFunc(reshape)
    glutMainLoop()

if __name__ == '__main__':
    main()
```

Para etsa funcion es la encargada de inicializar todo el entorno gráfico de OpenGL con GLUT, ademsa de esto crear la ventana, configurar cómo se va a ver el reloj, y poner el programa a funcionar en un bucle infinito que mantiene la ventana abierta y actualizada.

***Screenshot* del programa en ejecución.**



Asignatura	Datos del alumno	Fecha
Informática Gráfica y Visualización	Apellidos: Bocanegra Capera	19/mayo/2024
	Nombre: Oscar David	

La respuesta a estas preguntas: ¿Funciona bien tu programa? ¿Qué fallos existen?

Para el caso de esta actividad se evidencia que el programa funciona adecuadamente a lo solicitado, esto lo logre usando Python, y ejecutándolo con vscode, lo cual a nivel de ejecución no me dio ningún problema, el único inconveniente que tuve fue el poder ajustar el numero de las horas en su respectivo lugar, pero a la final se pudo solucionar, así que a mi parecer el programa funciona perfectamente con lo solicitado.