usepackageurl

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CIENCIAS

Práctica 1

Visualización

Ayala Flores Brenda

Fecha de entrega: 12 de septiembre de 2025

1. **¿Qué se conoce como Open Graphics Library?**

   Open Graphics Library (OpenGL) una API multilenguaje y multiplataforma para renderizar gráficos vectoriales 2D y 3D (utilizando polígonos para representar imágenes). La API de OpenGL está diseñada principalmente en hardware.

2. **¿Para qué se utiliza la biblioteca GLFW?**

   GLFW es una biblioteca multiplataforma de código abierto para el desarrollo de OpenGL, OpenGL ES y Vulkan en el escritorio. Nos ayuda para crear ventanas, contextos y superficies, y recibir entradas y eventos. **GLFW** proporciona una ventana y un contexto OpenGL con solo dos llamadas de función, compatibilidad con múltiples ventanas, múltiples monitores, rampas de gamma y DPI alto, también admite entrada de eventos mediante teclado, mouse, gamepad, tiempo y ventana, a través de sondeo o devoluciones de llamadas.

3. **¿Para qué se utiliza la biblioteca GLEW?**

   Es una biblioteca multiplataforma de código abierto para la carga de extensiones C/C++. GLEW proporciona mecanismos eficientes en tiempo de ejecución para determinar qué extensiones OpenGL son compatibles con la plataforma de destino. El núcleo de OpenGL y la funcionalidad de las extensiones se exponen en un único archivo de encabezado. GLEW se ha probado en diversos sistemas operativos, como Windows, Linux, Mac OS X, FreeBSD, Irix y Solaris.

4. **Describe brevemente las diferencias entre OpenGL, GLFW y GLEW.**

   OpenGL es una API para renderizar gráficos, GLFW es una biblioteca que crea ventanas y maneja eventos y GLEW es una biblioteca que carga los extensores de OpenGL en tiempo de ejecución.

5. **En el pipeline de OpenGL, ¿qué ocurre en el vertex shader?**

   El vertex shader es la primera etapa programable del pipeline gráfico de OpenGL y también la única obligatoria: Su función principal es transformar las posiciones de los vértices (de coordenadas del mundo a coordenadas de pantalla) y generar información que será utilizada por las siguientes etapas del pipeline.

6. **En el pipeline de OpenGL, ¿qué ocurre en el fragment shader?**

   Su tarea principal es calcular el color de cada fragmento individual, y potencialmente otras propiedades como la profundidad (depth), transparencia o coordenadas para mapas de texturas. A diferencia del vertex shader, que opera por cada vértice, el fragment shader se ejecuta una vez por fragmento, lo que puede traducirse en millones de ejecuciones por cuadro en una escena compleja. Es aquí donde se definen los detalles visuales más importantes como: iluminación, texturas, reflejos, sombreado, etc.

7. **¿Para que se utiliza la clase std::vector y cuales son los métodos más utilizados?**

   Podemos acceder a los elementos del contenedor vector usando un índice y el operador de indexación []. Los elementos ocupan posiciones contiguas en memoria. Ofrece un tamaño dinámico, es decir, puede crecer o reducirse automáticamente al añadir o eliminar elementos. Un vector es el contenedor más apropiado para una secuencia cuando el rendimiento de acceso aleatorio es importante. Uno de los métodos más utilizados son:

   - operator[](index): Devuelve una referencia al elemento en la posición index
   - capacity(): Devuelve el número de elementos que el vector puede almacenar sin necesitar reasignar memoria.
   - clear(): Elimina todos los elementos del vector, dejándolo vacío.
   - size(): Devuelve el número de elementos actuales en el vector.
   - empty(): Devuelve true si el vector está vacío, y false en caso contrario.

Referencias

[1] An OpenGL library. (2024, 23 de febrero). *GLFW*. https://www.glfw.org/

[2] El contenedor vector — Fundamentos de Programación en C++. (s.f.). https://www2.eii.uva.es/fund$_i nf/cpp/temas/9_v ectores/vectores_s tl.html : : text = en$

[3] GeeksforGeeks. (2025, 23 de julio). Getting started with OpenGL. *GeeksforGeeks*. https://www.geeksforgeeks.org/computer-graphics/getting-started-with-opengl/

[4] La biblioteca Wrangler de extensiones OpenGL. (s.f.). Recuperado 12 de septiembre de 2025, de https://glew.sourceforge.net/

[5] TylerMSFT. (s.f.). *vector (clase)*. Microsoft Learn. https://learn.microsoft.com/es-es/cpp/standard-library/vector-class?view=msvc-170

[6] Zavala, F. (2025, 20 de junio). *Cómo funciona el pipeline gráfico de OpenGL*. Código En Llamas. https://codigoenllamas.com/graphics-pipeline-en-openglheading-vertex-shader