Clase: Introducción a DearPyGui

(Framework gráfico rápido para Python con enfoque en aplicaciones de alto rendimiento y herramientas internas)

1. Esquema de Trabajo (Modelo Conceptual)

DearPyGui (DPG) sigue un enfoque **"inmediato"** (similar a Dear ImGui en C++), donde:

No hay bucles de eventos tradicionales (como en Tkinter/PyQt).

La interfaz se reconstruye en cada frame (cuadro), pero de manera optimizada. **Los widgets son "volátiles"**: Se destruyen y recrean automáticamente, pero el estado se gestiona mediante identificadores únicos.

2. Funcionalidades Clave

A. Widgets y Jerarquía

Contenedores: window() , group() , child() .

Controles básicos: button() , text() , input_text() , slider_float() , etc. **Avanzados**:

Gráficos (plot()), tablas (table()), menús (menu_bar()).

Ejemplo mínimo:

```
python

import dearpygui.dearpygui as dpg

dpg.create_context()
with dpg.window(label="Mi Ventana"):
    dpg.add_text("Hola, alumnos!")
    dpg.add_button(label="OK", callback=lambda: print("Click!"))
dpg.create_viewport(title="Demo", width=600, height=300)
dpg.setup_dearpygui()
dpg.show_viewport()
dpg.start_dearpygui()
dpg.destroy_context()
```

B. Callbacks y Datos

Callbacks: Funciones que se ejecutan al interactuar (ej: botón).

Data API: Acceso a valores de widgets con `get_value()` / `set_value()`.

Importante:

"Los callbacks pueden recibir el ID del widget que los triggered. DPG usa 'sender' y 'app_data' para pasar información."

C. Temas y Estilos

Temas predefinidos: `dpg.add_theme()` con colores y estilos.

Personalización: Modificar casi cualquier propiedad visual.

3. Modelo de Procesos (Flujo de Trabajo)

1. **Inicialización**:

```
python

dpg.create_context() # Crea el "universo" de DPG.
```

2. **Definición de UI**:

   Widgets se añaden dentro de bloques `with`.

3. **Viewport y Bucle Principal**:

```
python

dpg.create_viewport() # Ventana OS.
dpg.setup_dearpygui() # Prepara recursos.
dpg.show_viewport() # Muestra la ventana.
dpg.start_dearpygui() # Bucle interno (no bloqueante).
```

4. **Limpieza**:

```
python

dpg.destroy_context() # Libera memoria.
```

¡Punto clave!

"DPG maneja su propio bucle. No usamos `while True:` ni `root.mainloop()`. La UI se redibuja automáticamente, pero podemos pausar/continuar con `dpg.pause_dearpygui()`."

4. Diferencias Clave vs Otros Frameworks

Aspecto	DearPyGui	Tkinter/PyQt
Rendimiento	Alto (GPU acelerado)	Moderado/Bajo
Código	Menos verboso	Más estructurado
Uso típico	Herramientas internas	Apps generales

5. ¿Dónde se usa DearPyGui?

- **Tools para desarrollo**: Debuggers, editores de niveles.
- **Visualización rápida**: Datos científicos, prototipado.
- **Aplicaciones ligeras**: Que requieren alta frecuencia de actualización.

Posibles preguntas que tal vez nos hagamos.

Q1: ¿Puedo usar DPG para aplicaciones complejas con múltiples ventanas?

"Sí, pero requiere planear bien los IDs y el flujo de datos. DPG es ideal para herramientas compactas, no para suites tipo Office."

Q2: ¿Cómo manejo persistencia de datos (ej: guardar config)?

"Usa `get_value()` / `set_value()` o almacena datos en Python (listas, diccionarios) vinculados a callbacks."

Q3: ¿Es compatible con threads?

"No directamente. DPG corre en el hilo principal. Para tareas pesadas, usa `async` o procesos separados."