Función Render

En LibGDX, el método `render()` constituye la fase de actualización y dibujo del ciclo principal de ejecución (game loop). LibGDX invoca este método de forma automática en cada fotograma mediante su bucle interno, proporcionado por las implementaciones de `Application` (como `LwjglApplication` o `AndroidApplication`).

Responsabilidad principal

`render()` es responsable de:

1. **Actualizar el estado del juego**
    - Procesar entradas del usuario (`Gdx.input`)
    - Actualizar físicas, animaciones y posiciones
    - Ejecutar lógica dependiente del tiempo (normalmente usando el parámetro `delta`)
2. **Renderizar la escena**
    - Limpiar buffers gráficos (`Gdx.gl.glClear`)
    - Dibujar sprites, textos y entidades
    - Gestionar cámaras, viewports y lotes de dibujo (`SpriteBatch`)

Frecuencia de llamada

El motor llama a `render()` tantas veces como sea posible, normalmente a 60 FPS si la plataforma lo permite. La ejecución está controlada por:

- El subsistema gráfico subyacente (OpenGL)
- El sistema operativo
- Los límites definidos por el usuario (`setForegroundFPS()` en LWJGL3)

Variantes del método

1. `ApplicationAdapter`

```
public void render();
```

Este método **no recibe parámetros**, pero puede obtener el tiempo entre fotogramas mediante:

```
float delta = Gdx.graphics.getDeltaTime();
```

2. `Screen` y `Game`

```
public void render(float delta);
```

En este caso, LibGDX **calcula y pasa automáticamente** el tiempo transcurrido desde el último frame (`delta`), lo que permite implementar **lógica independiente del rendimiento**.

Secuencia típica dentro de render()

Para ApplicationAdapter

```
@Override
public void render() {
    float delta = Gdx.graphics.getDeltaTime();  // tiempo entre frames

    update(delta);  // lógica del juego
    draw();         // renderizado
}
```

Para Screen

```
@Override

public void render(float delta) {

    update(delta);

    draw();

}
```

Consideraciones técnicas

- LibGDX usa OpenGL ES (móviles) o OpenGL (desktop) para ejecutar el contenido de render().
- Este método debe ser rápido y no bloqueante: operaciones que tarden demasiado provocarán pérdida de FPS.
- No debe usarse para inicializar recursos; esto corresponde a create() o show().
- El orden correcto es:
    1. Limpiar buffers
    2. Configurar matrices/cámaras
    3. Comenzar lote de dibujo (batch.begin())
    4. Dibujar
    5. Finalizar lote (batch.end())