Main Render Loop

Gráficas por Computadora (CC2018) - 2026

Main Render Loop

Semestre 02, 2026

Objetivo

- Comprender qué es y cómo funciona el Main Render Loop.
- Saber cómo manejar entradas de usuario (teclado, mouse, etc.) dentro del loop.
- Aplicar buenas prácticas para mantener un framerate constante y un loop eficiente.

Definición

El Main Render Loop es el ciclo principal que ejecuta cualquier aplicación gráfica o videojuego. Permite:

- Procesar entradas de usuario.
- Actualizar el estado de la aplicación.
- Renderizar la escena en pantalla de manera continua.

Así se logra la ilusión de movimiento y la interactividad en tiempo real.

Importancia

Sin el render loop, las aplicaciones gráficas serían estáticas. Con el loop:

- Se puede responder al usuario en tiempo real.
- Se pueden crear animaciones fluidas.
- Se puede mantener actualizada la lógica de la aplicación.

Funcionamiento

En cada iteración del loop se realizan 3 pasos principales:

1. Handle Input – procesar entradas de teclado, mouse, etc.
2. Update State – actualizar posiciones, físicas, animaciones.
3. Render Frame – dibujar la nueva escena en pantalla.

Flujo

```
init_window(width, height)

while not window_should_close():
    process_input()
    update_scene()
    render_frame()
    swap_buffers()
    limit_framerate()
```

Detalle

Inicialización

- Crear la ventana.
- Cargar recursos: texturas, modelos, shaders.

Handle Input

El render loop escucha eventos de entrada: teclado, mouse, joysticks o controladores.

Eventos:

- `KEY_DOWN`: tecla presionada.
- `KEY_UP`: tecla liberada.
- `MOUSE_MOVE`: movimiento del cursor.
- `MOUSE_BUTTON`: clics.

Update State

- Calcula nueva posición de objetos.
- Aplica físicas y lógica del juego.
- Controla animaciones basadas en el tiempo transcurrido.

Render Frame

- Dibuja la escena actualizada en el framebuffer.
- Limpia la pantalla antes de dibujar para evitar artefactos.
- Muestra el resultado en la ventana.

Swap Buffers / Sincronización

Swap Buffers:

- Cambia entre el framebuffer actual y el próximo.

- Evita parpadeos (flickering).

Limitación de FPS:

- Mantiene un framerate estable con `vsync` o retrasos manuales.

Presupuesto de frame

Cada frame dispone de un tiempo fijo, que comparten process_input, update_scene y render_frame.

- 60 FPS → cada frame dispone de ~16.67 ms.
- 30 FPS → cada frame dispone de ~33.3 ms.

La cuenta es simple: tiempo por frame = 1000 ms / FPS. Si el trabajo de un frame excede su presupuesto, los FPS bajan y aparece el stutter.

Delta time

Si mueves un objeto una cantidad fija por frame (por ejemplo `x += 5`), la velocidad depende de los FPS: a 120 FPS avanza el doble que a 60. El juego correría distinto en cada máquina.

La solución es el delta time (dt): el tiempo transcurrido desde el frame anterior. Escalamos el movimiento por dt para que la velocidad sea constante en tiempo real, sin importar los FPS.

```
last_time = now()

while not window_should_close():
    current_time = now()
    dt = current_time - last_time
```

```
    last_time = current_time

    x += velocidad * dt
    render_frame()
```

- `last_time = now()`: guardamos el tiempo inicial antes del loop.
- `current_time = now()`: al inicio de cada frame tomamos el tiempo actual.
- `dt = current_time - last_time`: cuánto tiempo pasó desde el frame anterior.
- `last_time = current_time`: guardamos para el siguiente frame.
- `x += velocidad * dt`: al escalar por dt, la velocidad es constante sin importar los FPS.

vsync y límite de FPS

`limit_framerate` evita que el loop consuma el 100% de CPU y GPU sin control. Hay dos formas de lograrlo:

- vsync: sincroniza el swap con el refresco del monitor (por ejemplo 60 Hz). Elimina el tearing, pero limita los FPS al refresco.
- Límite manual: duerme (sleep) el tiempo sobrante del presupuesto de frame. Da control fino del ritmo, pero no elimina el tearing por sí solo.

Buenas prácticas

- Limpia el framebuffer al inicio de cada frame.
- Usa delta time para animaciones independientes del FPS.
- No cargues archivos dentro del loop → hazlo en inicialización.
- Mantén un framerate constante para evitar “stutter”.

En el proyecto de gráficas

El render loop es lo que envuelve el raycaster o raytracer:

- process_input: mueve la cámara o el jugador.
- update_scene: actualiza posiciones y ángulos con delta time.
- render_frame: lanza los rayos y escribe el color de cada píxel en el framebuffer.
- swap_buffers y limit_framerate: muestran el frame y mantienen el ritmo estable.