Simulación FX

Grado en Ingeniería del Software

U-tad — Curso 24/25

Laboratorio 3 (Stable Fluids I)

§ Instrucciones §

- El código correspondiente al laboratorio debe organizarse en un script de Python (`.py`) para cada método debidamente documentado.
- No está permitido el uso de código procedente de referencias o modelos de lenguaje (LLMs) como ChatGPT o Github Copilot.
- Todos los ficheros deben entregarse comprimidos en un fichero `.zip`.
- El laboratorio se puede hacer de forma individual o por parejas; el vídeo debe ser individual.

Difusión – Jacobi

El objetivo de este apartado es sustituir el método de difusión actual (*forward*) inestable implementado en `stable_fluid_starter.py`, por el método *backward* utilizando **Jacobi** para resolver el sistema de ecuaciones.

Los pasos sugeridos para resolver este apartado son los siguientes:

(1) Crear un parámetro `iters` que indique el número de iteraciones a realizar con el método de Jacobi.

(2) Sustituir el taichi kernel `diffuse` por una función (no `@ti.kernel`) `diffuse` que contenga el bucle de Jacobi. Para cada iteración de Jacobi, se debe llamar a un taichi kernel (`@ti.kernel`) `iter` que realice la iteración correspondiente a cada celda.

(3) Modificar la función `step` para que llame a la función `diffuse` en el lugar correspondiente.

Nota sobre la interfaz gráfica (GUI): Si experimentas problemas al utilizar la nueva interfaz de usuario GGUI de Taichi Lang (basada en Vulkan) puedes utilizar la versión anterior de la GUI. Para facilitar el uso de la versión anterior, se proporciona el código de inicio alternativo `stable_fluid_starter_no_vk.py`.

Video Explicativo

Grabar un video de **máximo 7 minutos** de duración donde se expliquen los siguientes aspectos de la implementación del método de Jacobi para la difusión:

- **Fundamentos Teóricos del Método de Jacobi:** Explicar brevemente en qué consiste el método de Jacobi para resolver sistemas de ecuaciones lineales, especialmente en el contexto de la difusión. Se puede hacer referencia a la notación vista en clase.
- **Implementación en Taichi y Python:** Describir cómo se ha implementado el método de Jacobi en el código, mostrando la estructura del código en Python y Taichi. Explicar la función `diffuse`, el kernel `iter`, y cómo se gestionan las iteraciones. **Mostrar claramente el paralelismo entre la notación matemática del método de Jacobi y la estructura del código implementado.**
- **Demostración Visual del Resultado:** Mostrar la simulación de fluidos funcionando con el método de difusión de Jacobi implementado. Explicar cómo se observa visualmente el efecto de la difusión en el campo de densidad.