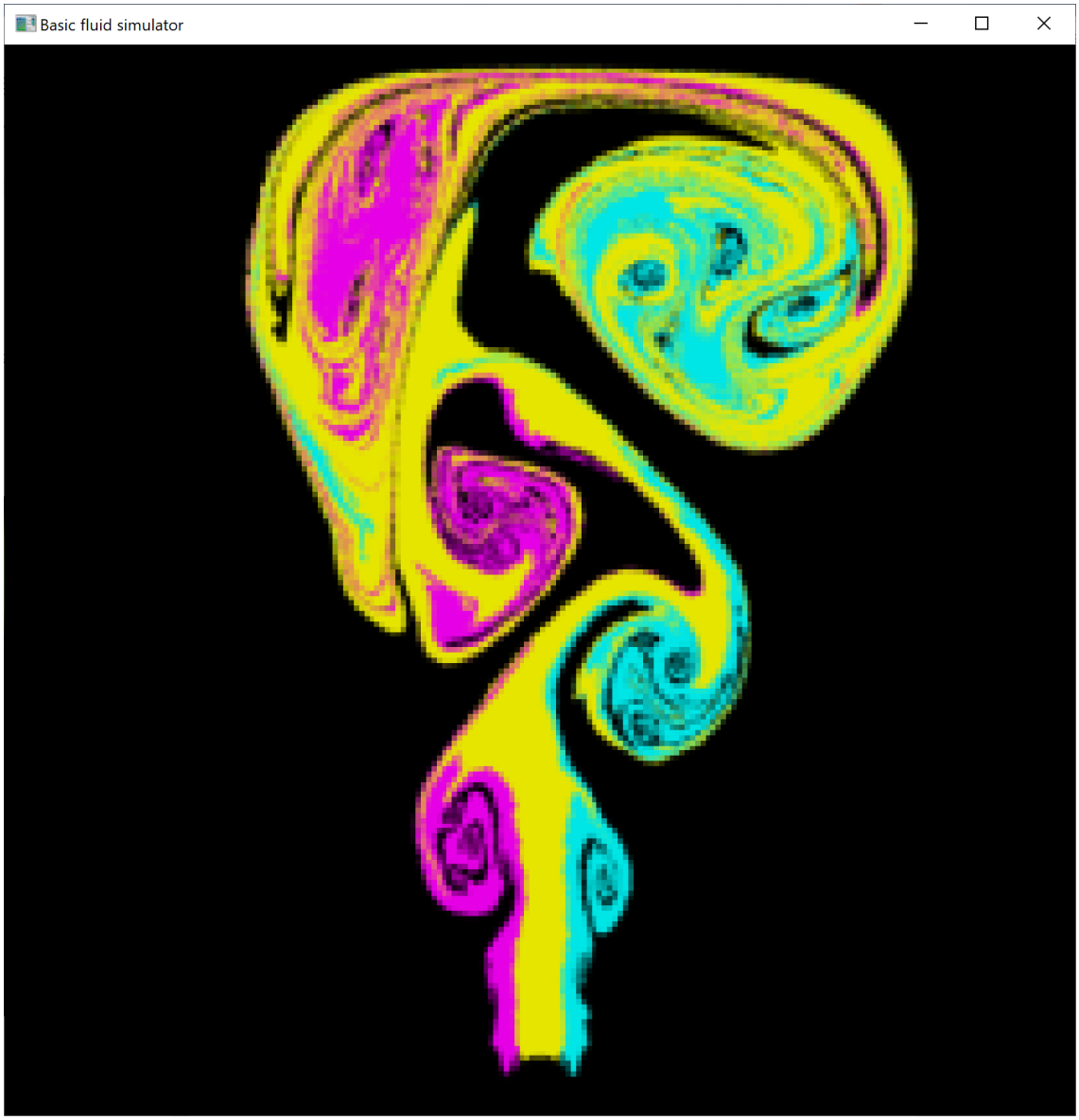
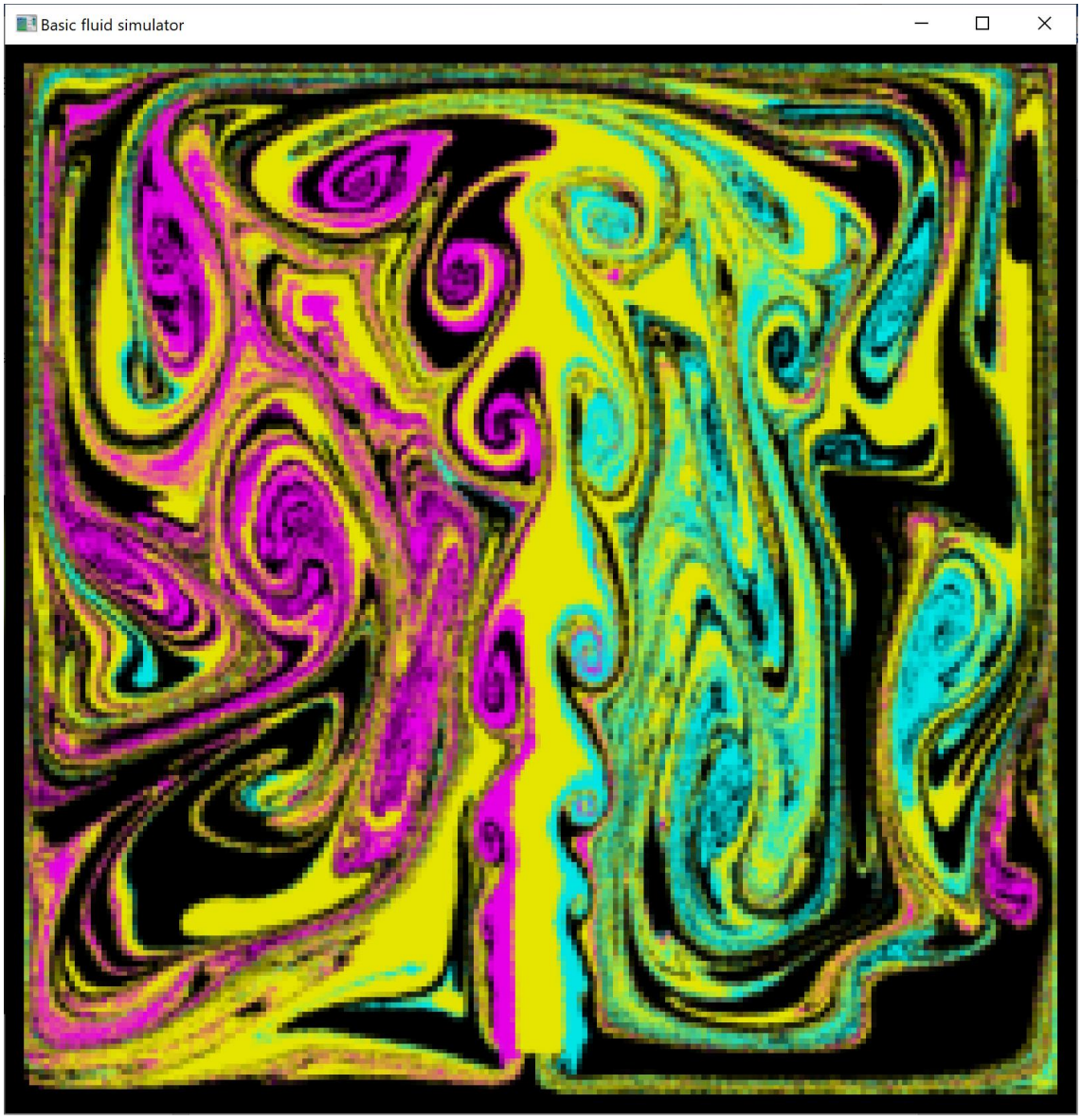
Animación Avanzada, Curso 2020-2021

# Práctica 3B: Simulador de Fluidos PIC/FLIP

(Asignación 13 de Abril; Entrega XX de Mayo a las 23:59)




Basic fluid simulator

**¿Cómo entregar la práctica?**

Enviar una copia del fichero *Fluid2Exercise.cpp* antes del XX de Mayo a las 23:59. Se habilitará el proceso de entrega a través del Aula Virtual de la asignatura durante la semana previa a la entrega. En caso de que surja algún problema con el campus virtual se podrá realizar la entrega enviando el resultado de la práctica a través de email <*ivan.alduan@urjc.es*>.

**Especificaciones Generales:**

Se proporciona un proyecto sobre *Microsoft Visual Studio 2017/2019* preconfigurado con todos los ficheros de código que se encargan de inicializar y configurar la simulación, ejecutar el bucle de simulación, y visualizar los resultados utilizando la librería *freeglut*. La práctica también se puede realizar en otra plataforma (Linux, Mac) y compiladores, utilizando el fichero *CMakeLists* proporcionado y suponiendo que dicha plataforma disponga de una implementación de *Glut*.

El ejercicio consiste en la programación de cada paso necesario para la simulación de un fluido en 2D Híbrido siguiendo la teoría de las clases. La implementación de cada uno de los pasos necesarios para el funcionamiento de la simulación se encuentra en *Fluid2Exercise.cpp*. Todo nuevo código ha de ser añadido en el fichero F*luid2Exercise.cpp*. La práctica está preparada para que no se necesiten librerías adicionales. Todo el código adicional que se precise ha de estar incluido en el fichero *Fluid2Exercise.cpp*.

La práctica se evaluará intercambiando el fichero F*luid2Exercise.cpp*, compilando y ejecutando, con lo cual no compilará si se precisan otros ficheros.

De forma excepcional, en caso de que se realicen modificaciones adicionales necesarias para la evaluación de la práctica, se deberán incluir en la entrega todos los ficheros afectados y una breve explicación de los cambios realizados. La práctica está preparada para que se pueda obtener la máxima calificación modificando únicamente el fichero de entrega F*luid2Exercise.cpp.*

No se requiere memoria explicativa, la evaluación se realizará a partir del código fuente.

**Explicación de la Demo:**

Este ejercicio toma como punto de partida la solución de la primera parte de la práctica con todas las *boundaries* del dominio tratadas como sólido. La demo contiene todo el framework necesario para realizar una simulación de fluidos Híbrida como la de la imagen, a falta de que el alumno integre su solución de la Práctica 3A y proporcione las rutinas adicionales para la implementación de un método PIC/FLIP.

Este proyecto incorpora varios cambios adicionales respecto a la base de código anterior. estos cambios pretenden facilitar al alumno la implementación del método híbrido objetivo de esta parte. La clase Fluid2 ahora incorpora una variable *flipEnabled* (por defecto activa) que permite ejecutar el simulador resultado de la parte anterior de la práctica si esta variable se encuentra inactiva. Cuando la variable *flipEnabled* se encuentre activa, varias porciones de código adicionales dentro de *Fluid2Exercise.cpp* serán ejecutadas, convirtiendo el simulador en un simulador de fluido PIC/FLIP. La implementación de estas partes adicionales es el objetivo de esta práctica. Para no complicar el código y desviar el objetivo de la práctica no se permite cambiar dinámicamente el método de simulación, sino que será necesario parar la ejecución y recompilar el ejercicio con el nuevo valor de *flipEnabled*.

Al iniciar la demo esta se encuentra detenida.

La tecla *'s'* permite pausar/continuar la simulación.

La tecla *'g'* permite visualizar la discretización subyacente a la simulación.

La tecla *'ESC'* permite salir del programa.

**Clases Particle2, Fluid2, FluidVisualizer2:**

La clase Particle2 es la única clase nueva respecto a la base anterior de la práctica. Esta clase permite manejar un array de partículas con todas las propiedades que se van a necesitar.

Varias clases previamente presentes como la clase Fluid2 o la clase FluidVisualizer2 se han modificado para proporcionar al alumno el esqueleto necesario para un simulador Híbrido.

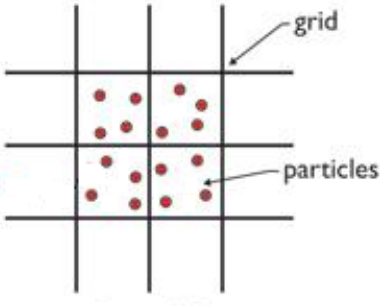
La clase Fluid2 contiene algunos campos adicionales como un sistema de partículas y algunas propiedades de rejilla adicionales que será necesario utilizar de forma correcta para implementar el método PIC/FLIP.

La clase FluidVisualizer2 se ha modificado y ahora es capaz de pintar en el visor de escena todas las partículas contenidas en nuestro fluido.

**Notas adicionales de la práctica:**

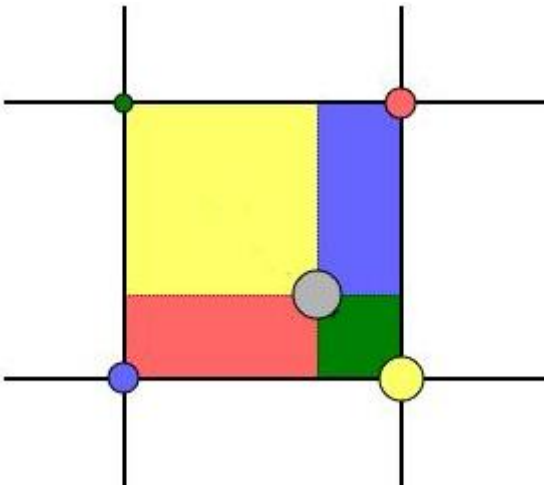
Dentro del fichero *Fluid2Exercise.cpp* se encuentran todas las porciones de código adicionales que se deben incorporar. A continuación, se detalla la funcionalidad de cada una de las porciones de código a implementar:

I. Implementación de la función **initParticles**: Esta función se ejecuta únicamente al inicio de la simulación si la variable *flipEnabled* está activa. El objetivo de esta función es realizar el sampling de partículas en todo el dominio de nuestra rejilla, para contener este array de partículas la clase Fluid2 tiene una nueva variable llamada *particles*. Por cada celda se recomienda introducir 4 partículas debidamente espaciadas, adicionalmente podemos aplicar un pequeño desplazamiento random para que el sampling no sea totalmente regular. Una vez tengamos realizado el sampleado el visualizador nos mostrará la posición de todas nuestras partículas de fluido.




II. El método **fluidAdvection** debe de ser reimplementado ya que ahora utilizaremos las partículas para realizar este paso del simulador. En concreto las tareas que se deben implementar son en el siguiente orden:
(1) Integración de la posición de las partículas mediante un método del Punto Medio (Runge-Kutta 2) utilizando las velocidades actuales del grid.
(2) Asegurarse de que todas las partículas tras moverse están contenidas dentro del dominio de la simulación, en caso contrario deberemos proyectar las partículas a posiciones dentro del grid.
(3) Crear las rejillas de ink, velocityX, velocityY a partir de las propiedad de las partículas, para la realización de este paso la contribución de cada partícula debe de repartirse entre los cuatro puntos de la rejilla más cercanos, con unos pesos que dependerán de lo cerca o lejos que se encuentra la partícula de cada punto, tras sumar la contribución de todas las partículas tendremos en un array la suma de contribuciones ponderadas y en otro la suma de todos los pesos aplicados en cada punto, utilizaremos estos pesos para normalizar la suma de cantidades en la rejilla.

(4) Nos guardaremos el estado actual de velocidades, la clase Fluid2 tiene nuevas variables para ello.



III. El método **fluidEmission** debe de ser reimplementado, en el caso de que FLIP esté activo este método deberá de modificar las propiedades de todas las partículas contenidas dentro del source.

IV. El método **fluidPressureProjection** en caso de que FLIP esté activo realiza una serie de pasos adicionales al final, tendremos que calcular el delta o diferencia entre las velocidades en rejilla ahora y las que nos guardamos anteriormente, tras ello en cada partícula aplicaremos una actualización de su velocidad utilizando una mezcla entre el método PIC y el método FLIP:

PARTICLE_VELOCITY = (0.95 * FLIP_VELOC + 0.05 * PIC_VELOC)