Animación Avanzada, Curso 2014-2015

Práctica 3 (b): Simulador de Fluidos Híbrido PIC/FLIP

(Asignación 20 de Abril; Entrega 18 de Mayo a las 23:59)

¿Cómo entregar la práctica?

Enviar una copia del fichero *fluid2Exercise.cpp* por email *ivan.alduan@urjc.es*, antes del 21 de Mayo a las 23:59.

Especificaciones Generales:

Se proporciona un proyecto sobre *Microsoft Visual Studio 2010* 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 otras plataformas (Linux, Mac…), suponiendo que dicha plataforma disponga de una implementación de *Glut*.

Partiendo del ejercicio anterior, este ejercicio consiste en la programación de todos los elementos adicionales para convertir nuestro simulador en un simulador híbrido que utilice el método PIC/FLIP. 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 *fluid2Exercise.cpp*. No se han de enviar más ficheros, sólo *fluid2Exercise.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 *fluid2Exercise.cpp*, compilando y ejecutando, con lo cual no compilará si se precisan otros ficheros.

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. En el proyecto adjunto se proporciona al alumno una implementación del fluido básico objeto de la primera parte de la práctica, pero cada alumno es libre de decidir si prefiere continuar con su solución de esa parte de la práctica o partir de la solución proporcionada.

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 *'ESC'* permite salir del programa.

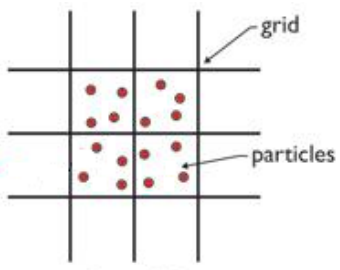
Clase Particle2, Fluid2, FluidVisualizer2:

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. En concreto 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.

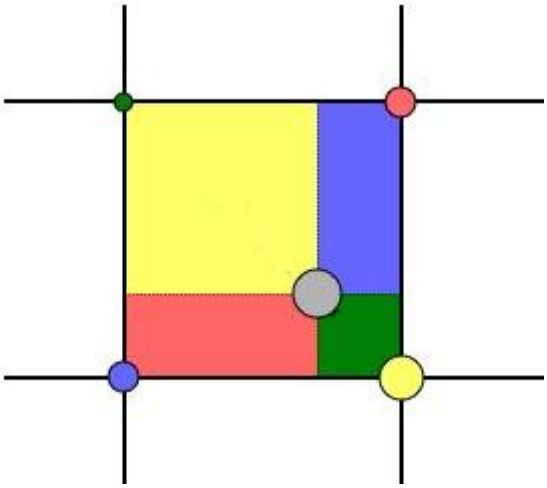
Partes obligatorias de la práctica:

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(void): 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 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 las partículas del 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 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 una 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.vel = (0.95 * FLIP_VELOCITY + 0.05 * PIC_VELOCITY);
```

Partes optativas de la práctica:

Hasta aquí se considera la parte obligatoria de la práctica, en caso de que el alumno desee profundizar más en ella se recomienda leer el artículo "*Visual Simulation of Smoke*" de Ronald Fedkiw en 2001, y tratar de aplicar algunas de las modificaciones que en dicho artículo se proponen como son una fuerza de buoyancy en lugar de una gravedad uniforme, y fuerzas de vorticity confinement. También podemos implementar una evolución del ink de nuestras partículas en el mismo estilo que el PIC/FLIP sobre velocidades, en este caso en rejilla se aplicaría un proceso de difusión sobre el ink (Laplacian) para modelar cómo las moleculas de humo tienden a dispersarse. En caso de que se decida implementar algunas de estas modificaciones es posible que se requiera modificar las clases Particles2 o Fluid2 para incorporar nueva funcionalidad. En este caso adjuntar también los ficheros modificados en la entrega.