Programación concurrente: Trabajo Final

Alumno: Gregorio Iniesta Ovejero

Descripción del proyecto

Consiste en implementar una compartición de datos entre las tecnologías OpenGL y OpenCL.

El escenario de pruebas va a consistir en una escena con cubos dispuestos a su vez en una matriz cúbica y al pulsar la tecla “E”, los cubos van a salir despedidos en dirección contraria a un punto en el espacio y a una velocidad que es proporcional al inverso del cuadrado de la distancia a ese punto. Además los cubos deceleran un 20% por segundo.

Detalles técnicos

OpenGL utiliza la técnica de renderizado “geometry instancing” que consiste en enviar una única malla y muchos puntos en los que dibujar esas geometrías.
Por lo tanto en nuestro escenario OpenGL necesita un cubo y un buffer de posiciones.

OpenCL se encargará en cada frame de mover los objetos cuando sea necesario en función de los requisitos anteriormente mencionados.
Por lo tanto OpenCL necesita poder acceder al array de posiciones de OpenGL y un array adicional para controlar las velocidades inaccesible por OpenGL.

El Kernel

Necesita acceder a 2 arrays globales y a 3 posiciones de cada array. Además a cada array se debe acceder tanto en lectura como en escritura para actualizar los datos.

Para optimizar los accesos a los buffer se guarda en una variable el contenido que se va a usar y no se vuelve a acceder al buffer hasta el final.

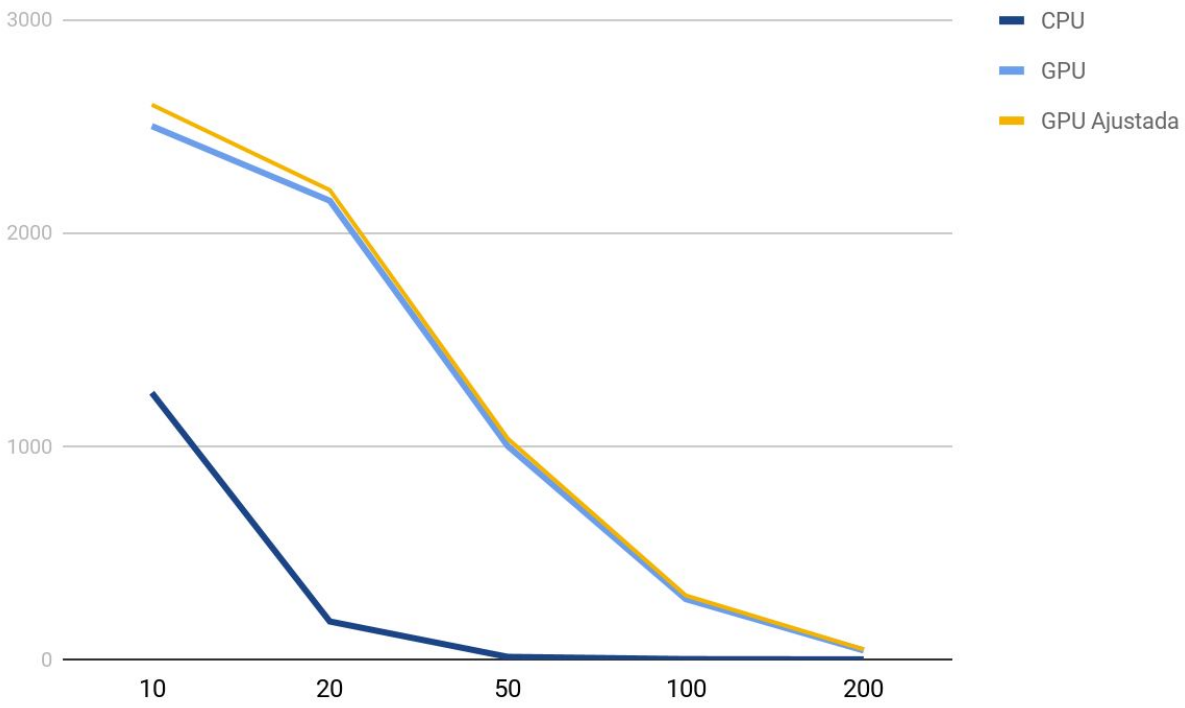
Aparte del if para evitar accesos a zonas de memoria prohibidas, solo hay un if que bifurca el comportamiento dependiendo si se ha añadido una nueva fuerza o no. Este if es homogéneo en toda la GPU, por lo tanto no habrá hilos inactivos en ningún momento.

Rendimiento

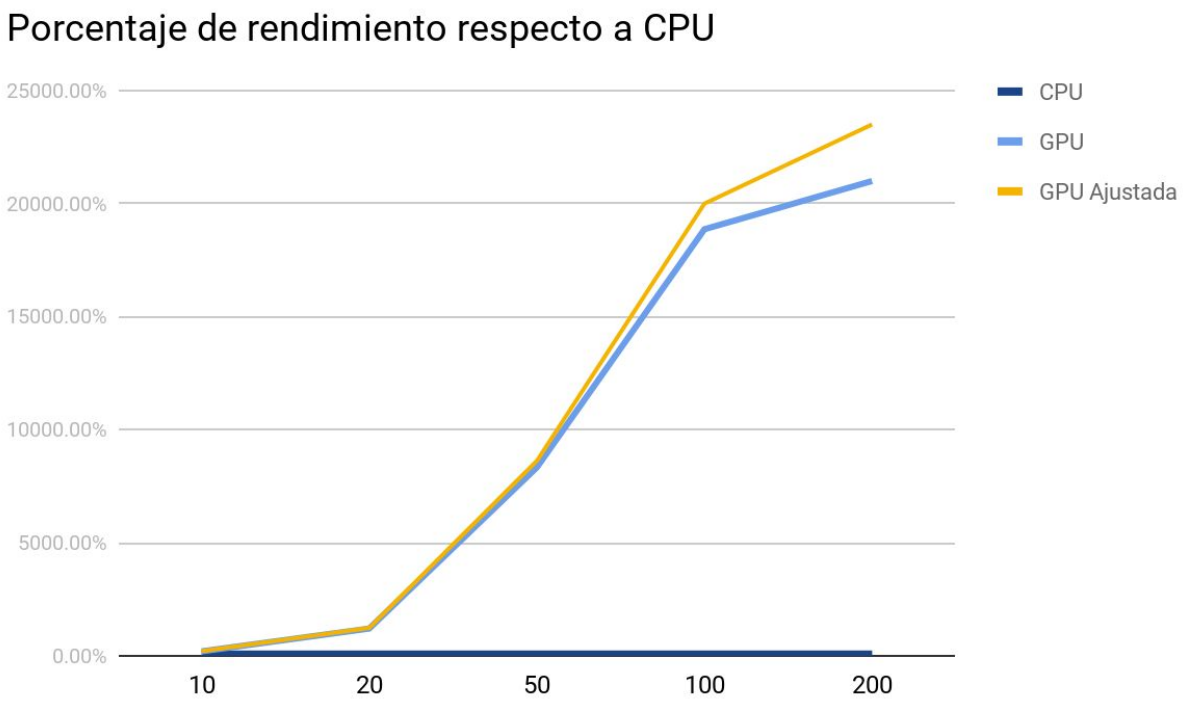
Ajustando el tamaño de los item group es posible mejorar ligeramente el rendimiento.
Ya que se trata de una aplicación gráfica, voy a expresar el rendimiento en FPS.
El equipo con el que se han hecho las mediciones tiene la siguientes características:
I7-6700 3.40GHz, Geforce GTX 1070, 16GB RAM

Dimensión de la matriz	10^3=1k	20^3=8k	50^3=125k	100^3=1M	200^3=8M
CPU	1250 fps	178 fps	12 fps	1.5 fps	0.2 fps
GPU	2500 fps	2150 fps	1000 fps	283 fps	42 fps
GPU ajustada	2600 fps	2200 fps	1035 fps	300 fps	47 fps

En líneas generales se puede ver como la GPU tarda mucho más en reducir el rendimiento que la CPU



Con respecto a la mejora relativa, se puede ver que la GPU tiene una mejora de rendimiento muy grande con respecto a la CPU. Además Se puede ver que ajustar los tamaños de grupo de la GPU tiene mayor impacto cuanto mayor es el problema a tratar.



Conclusiones

Es obvio que modificar elementos de la GPU de manera dinámica es muy costoso debido a las transferencias de grandes buffer de información desde la CPU a la GPU.
Además, el cómputo de miles o millones de elementos vía CPU uno a uno también se vuelve inviable para conseguir datos en tiempo real.
OpenCL soluciona estos dos problemas: Evita la transferencia de buffers modificados (se modifican directamente en GPU) y se tratan los numerosos elementos simultáneamente.
Obviamente, debido a las restricciones de computación en GPU, estos cómputos deben ser lo suficientemente generalistas como para poder actuar sobre todos los elementos del buffer.