BAKURETSU  
*(Explosión)*

Proyecto de TGA

Grupo 4

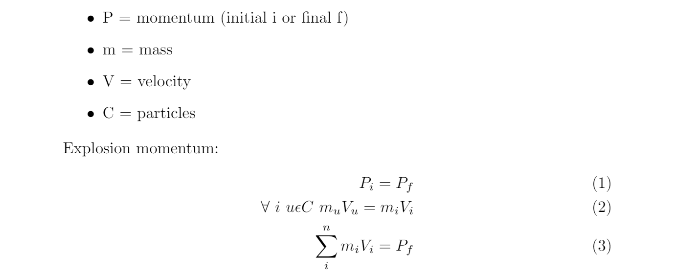
Alejandro Domínguez Besserer  
Marc Prat Masó

Descripción del Problema

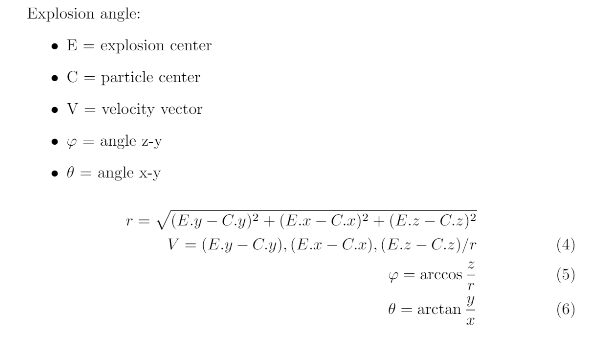
Nuestro proyecto consiste en la integración de **CUDA**, **OpenGL** y **GLSL** con el objetivo de renderizar una simplificación de explosiones reales. Reducimos el problema de simular una explosión realista a la visualización de un conjunto de partículas idénticas que son afectadas por una fuerza inicial aplicada desde un punto origen y que pueden colisionar de forma independiente con el medio o entre ellas. Debemos tener en cuenta, además, las limitaciones en el tiempo y los recursos de los que disponemos, así que prescindiremos de algunos elementos de realismo con tal de poder presentar un prototipo funcional.

Solución Propuesta

El modelo matemático que utilizamos es el siguiente:



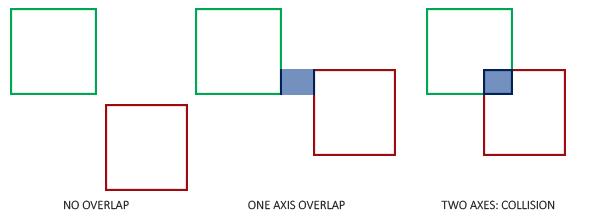
Las ecuaciones anteriores describen el instante pre-explosión, con las cuales calculamos el momento de cada partícula y sus velocidades iniciales.



Aquí podemos ver como determinamos el ánguloy la dirección iniciales del vector velocidad **V** de cada partícula.

La primera simplificación que realizamos es descartar el efecto de la gravedad sobre las partículas: supondremos que están en el espacio y contenidas dentro de una caja hueca. Por lo tanto, no consideramos ninguna variación de la dirección ni la velocidad en función del tiempo, pero sí en función de las colisiones que se producen.

Podemos detectar eficientemente cada colisión mediante el algoritmo **AABB**, que para cada pareja de partículas, las proyecta en los ejes de coordenadas y comprueba si existen puntos en común, si se da el caso en los tres ejes, tenemos dos partículas que colisionan.

  
*(El algoritmo* ***AABB*** *para el caso 2D)*

Una vez detectada una colisión, tenemos que determinar cómo se resolverá, calculando nuevos vectores de movimiento para las partículas que han chocado. Utilizamos la explicación indicada en el anexo (ii) como base para nuestra solución.

Solución Secuencial

La solución secuencial consiste en aplicar las fórmulas anteriores de manera secuencial para cada partícula, lo que implica que en el momento de detectar colisiones tenemos un coste cuadrático respecto al número total de elementos que supone un cuello de botella significativo.

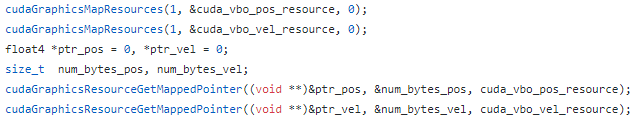
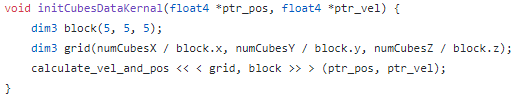
Programamos esta solución enteramente en **OpenGL** y **GLSL** (shaders), tanto la parte de cálculo como la visualización de la explosión.

Para reducir el coste en tiempo y memoria, sólo guardamos las partículas como vectores de 3 elementos que representan una de sus esquinas, después usamos un geometry shader para renderizar la forma cúbica que vemos finalmente.

Solución CUDA

Para obtener mayor rendimiento, usamos **CUDA** para paralelizar las partes del programa más costosas: todos los cálculos independientes que hacemos para cada partícula o par de partículas.

El primer reto que supone la utilización de **CUDA** es la integración con **OpenGL**. Relegaremos la tarea del cálculo intensivo a un programa **CUDA** y usaremos **OpenGL** sólo para la visualización del resultado. Esta implementación consta de un proceso de 6 pasos:

1. Crear un búfer que contiene un solo vértice para cada partícula y reservar memoria   
     
   
2. Registrar el búfer en **CUDA**
3. Mapear el búfer a **CUDA** (mientras esté mapeado a **CUDA**, OpenGL no puede acceder a él) y obtener un puntero a los datos  
     
   
4. Ejecutar el código **CUDA**

1. Desmapear el búfer de **CUDA** para que pueda ser accedido por **OpenGL**
2. Dibujar el frame y cambiar el búfer   
     
   

Análisis de los Resultados

Referencias

1. <https://www.nvidia.com/content/GTC/documents/1055_GTC09.pdf>
2. <http://www.yaldex.com/games-programming/0672323699_ch13lev1sec6.html>