INFO188 - 2021

Tarea 1 (grupal, max 3 personas)
Detección de colisiones con QuadTree

Profesor: Cristóbal A. Navarro**, Ayudante:** Enzo Meneses

Introducción

En esta tarea, usted implementará un detector de colisión de partículas en un espacio 2D mediante el uso de la estructura de datos Quadtree. Dada una lista de partículas (x,y), se deberá entregar la lista de colisiones. La siguiente Figura ilustra a la izquierda un ejemplo de como un espacio puede particionarse para ayudar a agrupar las partículas en base a la localidad. En el lado derecho de la misma Figura, se puede ver una ilustración de cómo es el QuadTree.

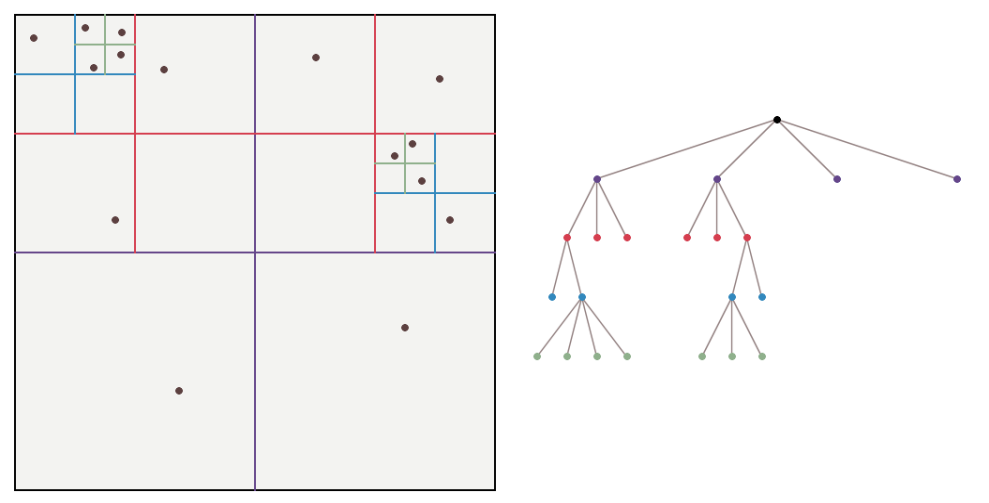

Figura 1: representación de un Quadtree

Descripción:

Se asume que cada partícula tiene un **radio r**, y se posicionan en un espacio bidimensional **[0,L] x [0,L]**. Una forma básica de implementar el proceso recién descrito es pasar por cada partícula y en cada una revisar todas las otras n-1 partículas para verificar si existe colisión o no con alguna, es decir revisar todos-con-todos. Esta técnica se considera el **algoritmo de fuerza bruta**, con

costo O(n^2), y es ineficiente ya que compara elementos que pueden estar evidentemente muy lejos para los que no era necesario compararlos.

Una técnica mucho más eficiente es utilizar un **QuadTree**, que subdivide el espacio y permita agrupar las partículas por cercanía, teniendo un costo **O(nlog(n))** en caso promedio. El **QuadTree** es un árbol de 4 hijos por nodo que va recursivamente subdividiendo el espacio en zonas de 2 x 2 (como en la Figura 1). Para cada nodo se necesita información auxiliar relacionada a la región que representa. También, la regla de subdivisión se debe definir a priori y se realiza en base a la cantidad de partículas permitidas por región. Por ejemplo, si la cantidad permitida es un **máximo de Q partículas por región**, y alguna región tiene **más que Q** partículas, entonces esta debe subdividirse en 2 x 2. El proceso de construcción del QuadTree consiste en ir subdividiendo las regiones (creando nodos hijos) hasta que estas satisfacen la condición de tener **solo hasta Q** partículas dentro. Las hojas del árbol tendrán una lista de máximo **Q** elementos.

Actividad de Programación

El objetivo de esta tarea es implementar tal proceso de detección de colisiones usando un Quadtree. El programa debe ejecutarse así:

```
./prog <m> <r> <L> <Q> <n>
```

```
<m> → Método a usar,  FuerzaBruta=0,  QuadTree=1
<r> → Radio de particulas
<L> → Tamaño del espacio 2D
<Q> → Cantidad máxima de partículas por región.
<n> → cantidad de partículas a generar.
```

Ejemplo de ejecución:

```
$> ./prog 1 1 100 8 25
**Detección de colisiones**
Metodo:     1 -> QuadTree
Parametros: r=1, L=100, Q=8, n=25
Generando 25 partículas aleatorias……OK  0.0002 secs
Calculando colisiones……OK: 0.0024 secs
Listado de colisiones:
[p4, p12] → (0.1, 0.1), (0.12, 0.13)
… (listado completo de colisiones)
```

Se recomienda al grupo trabajar de la siguiente forma:

1. **Programar la generación de partículas de forma aleatoria.**
    - Revisar el final de la clase IO para generar números aleatorios.
2. **Programar manejo de colisiones colFB usando fuerza bruta.**
    - Se utilizará para también comparar rendimiento.
3. **Implementar el tipo de dato QuadTree en haskell.**
    - Implementar la función **buildQT** que recibe una lista de partículas (x,y) y construye el Quadtree.
    - Implementar la función **checkColision** que recibe una partícula y su radio, y retorna una tupla de colisión (se puede detener a la primera colisión).
4. **Implementar el manejo de colisiones colQT usando el QuadTree.**
5. **Medir el tiempo (s) generación de partículas y detección (separados).**
6. **Paralelizar la detección de colisiones con cada método.**

Entrega plazo 24 de Noviembre del 2021, incluir:

- Informe de máximo 3 páginas + portada, el informe debe tener:
    - Introducción al problema y los desafíos detectados.
    - Solución propuesta
        - Cómo se generaron los puntos aleatorios.
        - Cómo se implemento el QuadTree.
        - Cómo se paralelizó cada método.
    - Gráficos de rendimiento:
        - Tiempo vs n (manteniendo una configuración r,L,Q)
            - En el mismo gráfico, ver ambos métodos.
        - Tiempo vs threads:
            - En el mismo gráfico, ver ambos métodos.
    - Conclusiones (comentar limitaciones y mejoras futuras).
- Comprimir su programa + informe en zip, e indicar instrucciones de compilación/ejecución/input/output.