UNIVERSIDAD NACIONAL DE SAN AGUSTIN

ESTRUCTURAS DE DATOS AVANZADAS

Práctica 03

Realizado por:

Pucho Zevallos, Kelvin Paul

2020

1. Modificaciones del Algoritmo QuadTree

1.1. Modificacion del archivo quadtree.js

1.1.1. Implementacion de la funcion *query*

El proposito de implementar esta funcion es que requiere encontrar todos los puntos dentro de un rectangulo siempre que este intersecte con las regiones del Quadtree.Tambien se debe verificar las propiedades de cada Quadtree que fue creado por ejemplo los puntos, estos deben pertenecer al Quadtree padre o a sus decendientes,al igual que ver si el Quadtree fue divido o no, esto para restringir la busqueda innecesaria.Estos procedimientos fueron realizados de manera recursiva.
Finalmente estos puntos seran registrardos en un arreglo y devueltos para se renderizados y que se muestre de manera mas notable los puntos encontrados.

```
query(range , found ){
        if(!found){
            found = [];
        }
        if(!this.boundary.intersects(range)){
            return found;
        }
        for(let i = 0; i < this.points.length; i++){
            if(range.contains(this.points[i])){
                found.push(this.points[i]);
                 cont++;
            }
        }
        if(!this.divided){
            return;
        }
        this.northeast.query(range,found);
        this.northwest.query(range,found);
        this.southeast.query(range,found);
        this.southwest.query(range,found);
        return found;
    }
```

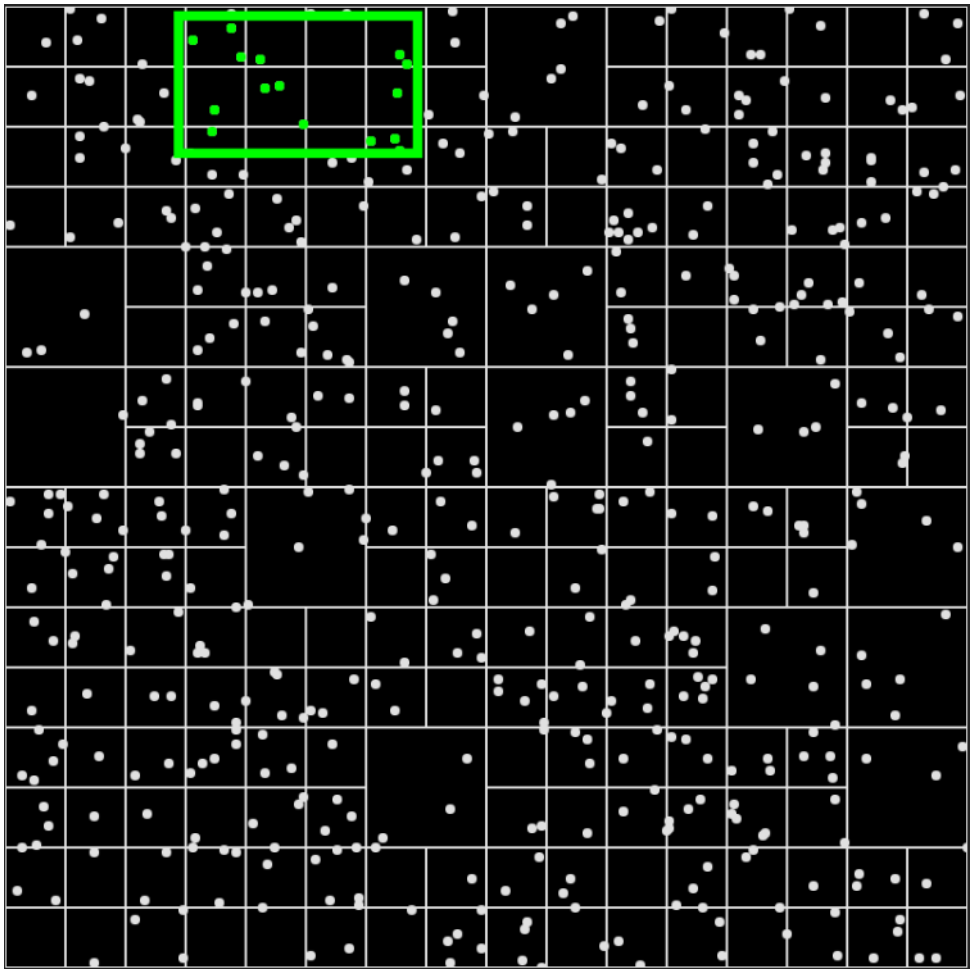
1.2. Modificacion del archivo sketch.js en la funcion setup

Es la encargada el nuevo rectangulo y renderizar los puntos encontrados

```
let qt;
let count = 0;

function setup(){
    createCanvas(400,400);
    let boundary = new Rectangle(200,200,200,200);
    qt = new QuadTree(boundary,4);
    console.log(qt);
    for (let i = 0; i < 500; i++){
        let p = new Point(Math.random()*400,Math.random()*400);
        qt.insert(p);
    }

    background(0);
    qt.show();

    stroke (0 ,255 ,0) ;
    rectMode(CENTER);
    let range = new Rectangle(random(200),random(200),random(50),
       random(50))
    rect (range.x, range.y, range.w*2 , range.h *2) ;
    let points = [];
    qt.query(range , points );
    let numpoints = 0;
    for (let p of points ){
        strokeWeight(4) ;
        point(p.x, p.y);
        numpoints++;
    }
    console.log(count);
    console.log(numpoints);

}
```

1.2.1. Resultados

Como se muestra en la figura siguiente, hay un rectangulo creado de manera aleatoria la cual busca los puntos dentro de su rango y los cambia de color para indicar los dichos puntos fueron encontrados.

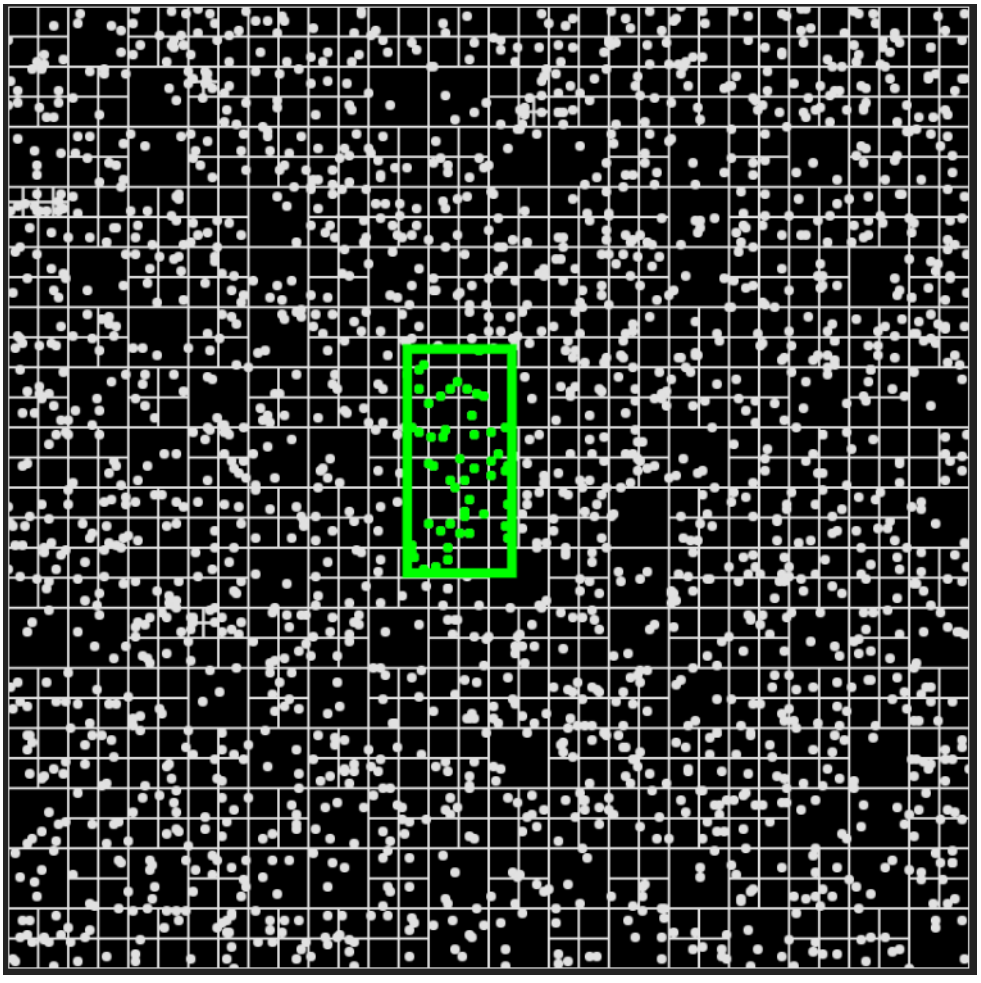


1.Visualizacion del Quadtree con 50 puntos y el rectangulo de busqueda.

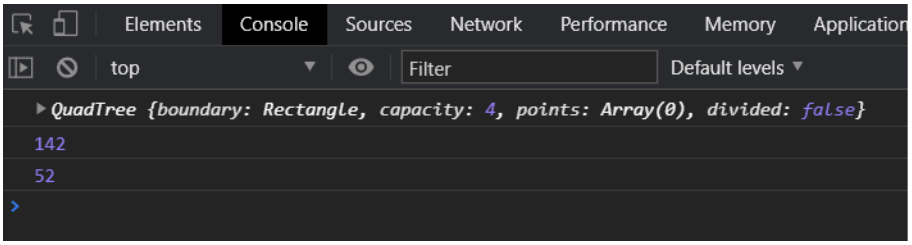
1.3. Evaluacion y verificacion de cuantas veces se consultada un punto,

Se coloco 2000 puntos, y en la figura siguiente muestra el registro de cuantas veces un punto es consultado y la cantidad de puntos que hay dentro del recuadro.

1.3.1. Resultados



2.Visualizacion del Quadtree con 2000 puntos y el rectangulo de busqueda.



3.Visualizacion de los registros de la consola

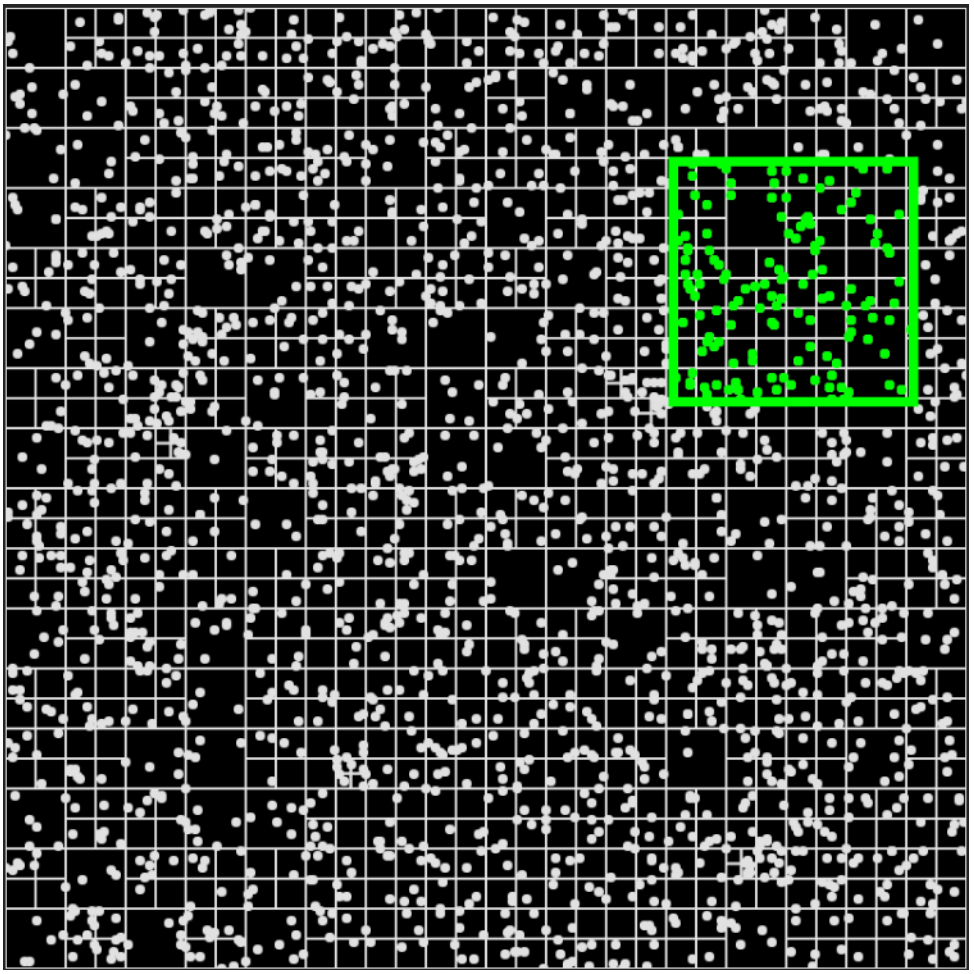
1.4. Modificacion del archivo Sketch.js creando la funcion draw

En esta parte se procedio a modificar la forma de como se hace presente el recuadro de busqueda de puntos, para eso se creo la funcion draw la cual nos permite dibujar un recuadro utilizando el mouse y que ademas esta implementado para que el usuario pueda interactuar de manera que cada ves que presione el click izquierdo el recuadro se muestra y encontrado los puntos que se desea,Ademas los registros de estos puntos se muestran en la consola del navegador

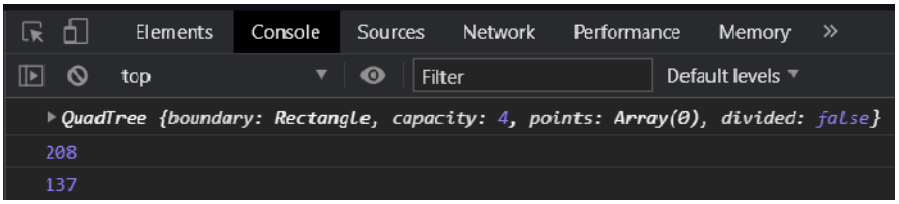
```
let qt;
let count = 0;

function setup(){
    createCanvas(400,400);
    let boundary = new Rectangle(200,200,200,200);
    qt = new QuadTree(boundary,4);
    console.log(qt);
    for (let i = 0; i < 2000; i++){
        let p = new Point(Math.random()*400,Math.random()*400);
        qt.insert(p);
    }

    background(0);
    qt.show();
}
function draw () {
    background (0);
    qt. show () ;
    if(mouseIsPressed){
        stroke (0 ,255 ,0);
        rectMode (CENTER);
```

```
23          let range = new Rectangle(mouseX,mouseY,50,50);
24          rect (range.x, range.y, range.w*2 , range.h *2) ;
25          let points = [];
26          qt.query(range , points );
27          let numpoints = 0;
28          for (let p of points ){
29              strokeWeight(4) ;
30              point(p.x, p.y);
31              numpoints++;
32          }
33          console.log(count);
34          console.log(numpoints);
35      }
36  }
```

1.4.1. Resultados



4.Visualizacion del Quadtree con 2000 puntos y el rectangulo de busqueda presionando el click izquierdo.



5.Visualizacion de los registros de la consola

2. Repositorio

https://github.com/kpzaolod6000/Cursos_2020_II/tree/master/EDA/semana_2/quadtree1.1