МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(МОСКОВСКИЙ ПОЛИТЕХ)**

09.03.01 Информатика и вычислительная техника

Образовательная программа (профиль)

«Интеграция и программирование в САПР»

Кафедра «СМАРТ технологии»

ОТЧЕТ

по дисциплине:

**Проектная деятельность**

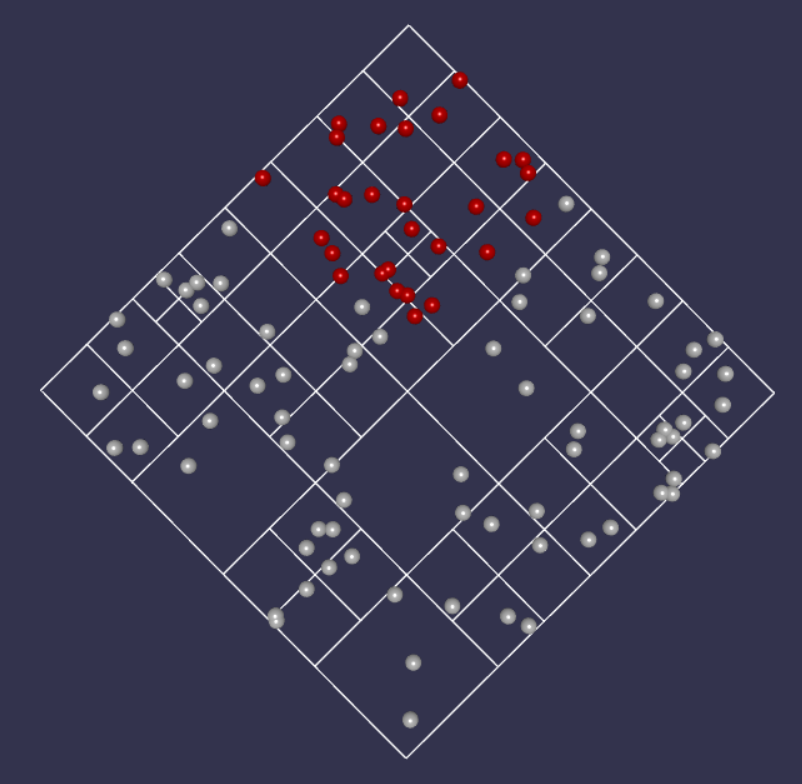
Преподаватель: / Толстиков А.В., к.т.н. /

*подпись ФИО, уч. звание и степень*

*Подготовили*

Марков Д.С. Хвалева А.Г Фильченков Д.П Шаехова Л.Р Владыкин Н.Н Телекаев Н.С

Москва, 2023 г.

****

|  |
| --- |
| class grid {      constructor(n, c) {          this.m = n; // число ячеек по стороне          this.cellSize = c;//  }          this.grid = Array.from({ length: this.m }, () => Array.from({ length: this.m }, () => []));      }      // Добавление элемента в ячейку с заданными индексами      addItem(row, column, item) {          if (this.grid[row] && this.grid[row][column]) {              this.grid[row][column].push(item);          } else {              console.error('Invalid row or column index.');          }      }      getItems(row, column) {          if (this.grid[row] && this.grid[row][column]) {              return this.grid[row][column];          } else {              console.error('Invalid row or column index.');              return [];          }      }  } |

Для более быстрого создания дерева нам необходимо разбить плоскость на ячейки и определить, какая точка к какой ячейке относится.

|  |
| --- |
| class QuadTree {        constructor(gr, M, D) { //  заполняемость и коэффициент глубины          let domain=[new BABYLON.Vector3(0, 0, 0), new BABYLON.Vector3(gr.m\*gr.cellSize, 0, gr.m\*gr.cellSize)]; //левый и правый угол четырех угольника          var points = [];            this.root=this.buildQuadratTree(gr,M,D,0,0,gr.m,0,gr.m);        } |

QuadTree – это сама структура данных.

Domain создает стартовый квадрат, который в дальнейшем мы будем делить.

Points – массив, хранящий точки.

Root – объект, хранящий корневой элемент дерева, с него начинается построение дерева.

|  |
| --- |
| buildQuadratTree(G, M, D, level, imin, imax, jmin, jmax) {  if (level >= D + 1) {  let array = []  const leftX = Math.floor(imin);  const leftY = Math.floor(jmin);  const rightX = Math.ceil(imax - 1);  const rightY = Math.ceil(jmax - 1);//получили углы  for (let i = leftX; i <= rightX; i++) {//можно убрать равно  for (let j = leftY; j <= rightY; j++) {  let getSphere = G.getItems(i, j)//получили массив точек  let arrayPoint = [];  G.grid[i][j] = arrayPoint;  for (let g = 0; g < getSphere.length; g++) {  array.push(getSphere[g])  }  }  }  let q = new QuadTreeNode(array)  return q  }  else {  let p = new QuadTreeNode();  p.rectangle.push(imin);  p.rectangle.push(imax);  p.rectangle.push(jmin);  p.rectangle.push(jmax);  let imid = (imin + imax) / 2;  let jmid = (jmin + jmax) / 2;  //записываем все по массиву  p.child[0] = this.buildQuadratTree(G, M, D, level + 1, imid, imax, jmid, jmax)//северо восток  p.child[1] = this.buildQuadratTree(G, M, D, level + 1, imid, imax, jmin, jmid)//юговосток  p.child[2] = this.buildQuadratTree(G, M, D, level + 1, imin, imid, jmin, jmid)//югозапад  p.child[3] = this.buildQuadratTree(G, M, D, level + 1, imin, imid, jmid, jmax)//северо запад  for (let i = 0; i < 4; i++) {  p.size += p.child[i].size;  }  if (p.size <= M || level >= D) {//слияем  p.points = []  for (let i = 0; i < 4; i++) {  for (let g = 0; g < p.child[i].points.length; g++) {  p.points.push(p.child[i].points[g])  }  delete p.child[i];  p.child[i] = [];  }  }  return p;  }  } |

buildQuadratTree - рекурсивный метод создания дерева.

if (level >= D + 1) - проверка на глубину рекурсии.

Если true: получаем точки из пространства, которое занимает ячейка, и возвращаем узловой элемент, в противном случае - создаем четыре дочерних узла.

Когда мы опустились на максимально доступную глубину рекурсии, мы начинаем выходить из нее (снизу вверх) и каждому родительскому узлу задаем точки, которые входят в его дочерние узлы

Если так получилось, что глубина рекурсии больше заданной (у автора это может произойти мы это убрали) или количество точек в узле меньше минимального, то происходит слияние

|  |
| --- |
| class QuadTreeNode {  constructor(array) {  this.rectangle = [];  this.child = [0, 0, 0, 0];  this.points = [];//точки если узел внешний  this.size // число точек накрываемых узлом  if (!array) {//пустой конструктор  //console.log(" ПУСТОЙ")  this.points = [];  this.size = 0;  for (let i = 0; i < 4; i++) {  this.child[i] = []  }  }  else {//Не пустой  this.points = array;  this.size = this.points.length;  for (let i = 0; i < 4; i++) {  this.child[i] = []  }  }  }  destructor() {  // Освобождение ресурсов или выполнение других действий  console.log("Destructor called");  }  } |

QuadTreeNode - класс узла.

Rectangle – зона.

Child – дети, если есть.

Size - число точек.

Если конструктор пустой - то создаем пустой узел, если нет - сразу записываем в него точки.

|  |
| --- |
| function pointInRectangle(point,min,max){        return point.x < max.x && point.x > min.x && point.z < max.z && point.z > min.z;    } |

Функция определяет, принадлежит ли точка заданному квадрату.

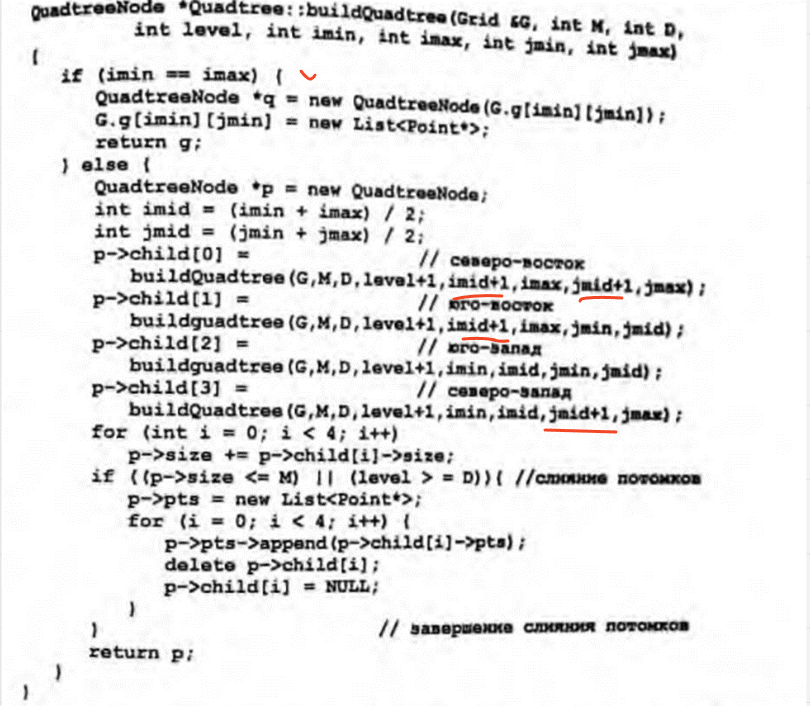
|  |
| --- |
| function traverseQuadTreeDrow(node) {//тут рисуем  if (!node) {  return;  }  // Рекурсивно обходим каждого потомка узла  for (let i = 0; i < 4; i++) {  var points = [  new BABYLON.Vector3(node.rectangle[0], 0, node.rectangle[2]),  new BABYLON.Vector3(node.rectangle[0], 0, node.rectangle[3]),  new BABYLON.Vector3(node.rectangle[1], 0, node.rectangle[3]),  new BABYLON.Vector3(node.rectangle[1], 0, node.rectangle[2]),  new BABYLON.Vector3(node.rectangle[0], 0, node.rectangle[2])  ];  var lines = BABYLON.MeshBuilder.CreateLines("lines", { points: points }, this.scene);  if (node.child[i] != 0) {  traverseQuadTreeDrow(node.child[i]);  }  }  } |

Функция для отрисовки линий.

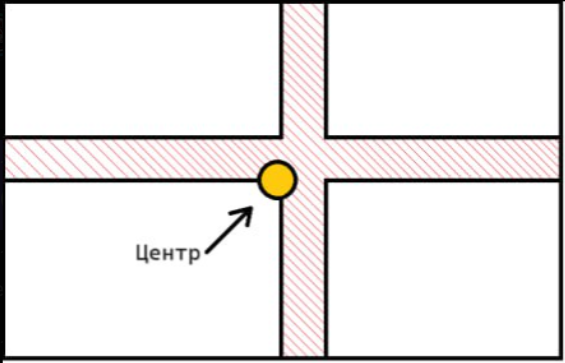
|  |
| --- |
| function traverseQuadTree(node, leftConor, rightConor) {//тут рисуем  if (!node) {  return;  }  const leftX = Math.floor(leftConor.x);  const leftY = Math.floor(leftConor.z);  const rightX = Math.ceil(rightConor.x);  const rightY = Math.ceil(rightConor.z);//получили углы  if (leftX <= node.rectangle[1] && node.rectangle[0] <= rightX && leftY <= node.rectangle[3] && node.rectangle[2] <= rightY) {  // console.log("asdad")  if (node.child[1] != 0) {//значит точно есть дети  for (let i = 0; i < 4; i++) {  traverseQuadTree(node.child[i], leftConor, rightConor);  }  }  else {  let getSphere = node.points//получили массив точек  var sphereMaterial = new BABYLON.StandardMaterial("sphereMaterial", this.scene);  sphereMaterial.diffuseColor = new BABYLON.Color3(1, 0, 0); // Красный цвет  for (let g = 0; g < getSphere.length; g++) {  if (pointInRectangle(getSphere[g].position, leftConor, rightConor)) {  getSphere[g].material = sphereMaterial;  }  }  }  }  } |

Функция покраски: если узел попадает в зону, смотрим его дочерние элементы, пока не дойдем до внешнего узла.

Места отклонения от оригинала:



1. Автор предлагает делать рекурсию, пока квадраты не сойдутся, что приведет к большой глубине рекурсии, что является излишним, так как пользователь сам задает необходимую глубину.
2. При создании дочерних элементов автор некорректно задает точки, пропуская часть территории.



Исполнение алгоритма на сайте [итмо](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D0%B2%D0%B0%D0%B4%D1%80%D0%BE%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F)

