软件工程第三次上机作业

班级： 软工1506班

小组成员：

黄健民 U201517147

薛睿 U201517136

问题描述：

In a box(3D) bounded by [-1, 1], given m balloons(they cannot overlap) with variable radio r and position mu. And some tiny blocks are in the box at given position {d};balloons cannot overlap with these blocks. find the optimal value of r and mu which maximizes

sum r^2

算法思想：

主要思想与实验1类似，不同之处就是,多了几个点，只需要把这些点当成半径为0的园加入气球队列即可。二是二维变成了三维。所以需要增加一个维度。

主要方法和实验一的一样，还是是通过计算出所有的情况来寻找最优解。只不过由于 增加了一个维度，运算速度非常慢。

由于圆心x、y坐标的精度是有限的，所以圆的位置是有限的。通过计算每个位置满足条件的最大半径r，从而找出这些点的最大半径； 在寻找最大半径时，每个位置的最大半径的精度只能取到小数点后五位。把满足条件的最大圆将放入一个链表中，这样一来之后的圆在进行位置判断时可以与已放入的圆进行比较，从而判断其是否满足条件；

代码：

#include <stdlib.h>

#include <stdio.h>

#include <math.h>

typedef struct ball { //气球结构体

double x; //气球球心x坐标

double y; //气球球心y坐标

double z; //气球球心z坐标

double r; //气球球心半径

}Ball;

typedef struct ballList { //用链表来储存已经放置的气球

struct ballList \* next;

Ball ball;

}BallList;

void insert(Ball ball);

double distance(Ball b1, Ball b2);

int judge(Ball b);

void putBall();

void putPoint(double x, double y,double z);

BallList \*head = NULL;

double step = 0.01; //改变气球位置的最小单位

int num = 0; //放置气球的个数

double sumr = 0; //用来记录r^2之和

int main() {

int n, pointNum, i;

printf("请输入气球总数量: ");

scanf("%d", &n);

printf("请输入点的数量: ");

scanf("%d", &pointNum);

for (i = 0; i < pointNum; i++) {

printf("请输入第%d个点的坐标(空格隔开)：", i + 1);

double x, y, z;

scanf("%lf %lf", &x, &y,&z);

putPoint(x, y, z);

}

printf("\n球编号\t x坐标\t y坐标\t z坐标\t 半径\t r^2之和\n");

for (i = 0; i < n; i++) {

putBall();

}

printf("\nr^2之和最大为:\t %lf\n", sumr);

system("pause");

return 0;

}

//将设置的点作为半径为0的点插入气球队列

void putPoint(double x, double y,double z) {

Ball ball = { x, y, z, 0 };

insert(ball);

}

void insert(Ball ball) { //将气球插入气球队列

BallList \* newBall = (BallList \*)malloc(sizeof(BallList));

newBall->ball = ball;

newBall->next = head;

head = newBall;

}

//改变气球的初始位置，求的满足条件的气球

void putBall() {

Ball ball = { -1 + step, -1 + step, -1 + step, 0 };

Ball maxBall = ball;

int i, j, k;

for (i = 0; ball.x < 1; ++i) {

ball.x += step;

ball.y = -1 + step;

for (j = 0; ball.y < 1; ++j) {

ball.y += step;

ball.z = -1 + step;

for (k = 0; ball.z < 1; ++k) {

ball.z += step;

ball.r = 0;

double rstep = 0.1;

while (rstep > 0.00001) {

if (ball.r > maxBall.r) {

maxBall = ball;

}

ball.r += rstep;

if (!judge(ball)) { //此气球不合适，减小半径与步长重新尝试

ball.r -= rstep;

rstep /= 10;

}

}

}

}

}

if (judge(maxBall)) {

insert(maxBall);

num++;

sumr += maxBall.r \* maxBall.r;

printf("%d\t %.3lf\t %.3lf\t %.3lf\t %.3lf\t %lf \n", num, maxBall.x, maxBall.y, maxBall.z, maxBall.r, sumr);

}

}

//判断气球之间的距离

double distance(Ball b1, Ball b2) {

double x1 = b1.x;

double y1 = b1.y;

double z1 = b1.z;

double x2 = b2.x;

double y2 = b2.y;

double z2 = b2.z;

return pow((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2) +(z1 - z2)\*(z1 - z2), 0.5);

}

//判断新加入的气球是否符合规则

int judge(Ball b) {

//将气球限制在[-1，1]内

if ((fabs(b.x) + b.r) > 1 || (fabs(b.y) + b.r) > 1 || (fabs(b.z) + b.r) > 1) {

return 0;

}

//依次比较气球b与已有气球是否相交

BallList \*tmp = head;

while (tmp) {

Ball ball = tmp->ball;

if (distance(b, ball) < b.r + ball.r) {

return 0;

}

tmp = tmp->next;

}

return 1;

}

运行结果

请输入气球总数量: 4

请输入点的数量: 2

请输入第1个点的坐标(空格隔开)：0.1 0.1 0.2

请输入第2个点的坐标(空格隔开)：0.3 0.4 0.5

球编号 x坐标 y坐标 z坐标 半径 r^2之和

1 -0.320 -0.320 -0.320 0.675 0.455085

2 0.440 0.450 0.460 0.540 0.746577

3 0.400 0.530 -0.540 0.460 0.958177

4 0.540 -0.540 0.540 0.458 1.168216

r^2之和最大为: 1.168216

请按任意键继续. . .