//引入头文件

#include <stdlib.h>

#include <math.h>

#include <stdio.h>

#include <curses.h>//控制台清空屏幕使用

//引入OpenGL头文件

#ifdef \_\_APPLE\_\_

#define GL\_SILENCE\_DEPRECATION

#include <OpenGL/gl.h>

#include <GLUT/glut.h>

#else

#include <GL/gl.h>

#include <GL/glut.h>

#endif

//定义结构体，用于存放作图的变量和数据

**struct** DATA

{

**double** ratio; //用于调整视窗的投影比，确保图像不是真

**int** center\_horizontal;//图像位于中间的标号

**int** center\_vertical;//确定垂直显示的图像

**int** center\_horizontal\_MAX;

**int** center\_vertical\_MAX;//这是上面两个量的最大可取值

**double** delta\_x; //作图区域的水平移动

**double** delta\_y;//作图去的垂直移动

**double** pixel\_scale; //作图的像素大小

};

**struct** DATA data={0.0,1.0,1.0,0,0,0.0,0.0,0.0};

//定义热传导变量

**int** choice=0,method=0,option=0,switch\_to\_final;//分别为选择计算方式，选择迭代方式，选择运行模式

**int** Index=0,Index\_cache=0;//第一个为计算缓存索引，后者为作图缓存索引

**double** factor;//这是热传导计算的常数因子

**double** scale;//这是出迭代循环的条件

**double** \*\*\*p0,\*\*\*p1, \*\*\*temperature0, \*\*\*temperature1,\*\*\*temperature\_cache;

**double** dX,dt,K,t;//这是热传导常量

**int** nX,nY,nZ,nt;//这是空间时间格点数量

**double** sor\_w;//这是迟豫因子

**double** \*picture;//这是用于存图像时间的动态数组

**int** picture\_number=0;//保存的中间态数量

**char** Y\_OR\_N;//这是选择是否存中间态

**int** draw\_picture\_number=1;//这是选择作图方式

**double** final\_t;

**int** picture\_frame\_number=0;

**int** flag=0;

//热传导计算以及数据的存储

//将数据存到/Volumes/UDISK/picturei.txt

**void** save\_picture(**int** i)

{

FILE \*fp=**NULL**;

**char** fileName[50] = "/Volumes/UDISK/picture0.txt";

fileName[22]=49+i;

fp = fopen(fileName, "w");

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

**for**(**int** c=0;c<nZ;c++)

{

fprintf(fp,"%lf\n",temperature0[a][b][c]);

}

fclose(fp);

}

//将计算作图信息存到/Volumes/UDISK/

**void** file\_exe1(**void**)

{

FILE \*fp=**NULL**;

fp=fopen("/Volumes/UDISK/Information.txt","w");

fprintf(fp,"%d\t",picture\_frame\_number);

fprintf(fp,"%lf\n",dX);

fprintf(fp,"作者：侯肇帮\t学号：2021040003\nC语言程序设计课期末大作业\n选题：热传导仿真\n");

fprintf(fp,"=============================================================================\n");

**switch** (choice) {

**case** 1:

fprintf(fp,"仿真方法：显示差分法\n");

fprintf(fp,"空间精度：%lf\n",dX);

fprintf(fp,"时间精度：%lf\n",dt);

**if**(switch\_to\_final==1)

{

fprintf(fp,"模拟时长：%lf\n",t);

}

**else** **if**(switch\_to\_final==2)

{

fprintf(fp,"达到稳定态用时：%lf\n",final\_t);

}

**break**;

**case** 2:

fprintf(fp,"仿真方法：隐式差分法\n");

**switch** (method) {

**case** 1:

fprintf(fp,"迭代方式：Jacobi法\n");

**break**;

**case** 2:

fprintf(fp,"迭代方式：Gauss-Seidel法\n");

**break**;

**case** 3:

fprintf(fp,"迭代方式：Gauss-Seidel法\n");

fprintf(fp,"松弛因子: %lf\n",sor\_w);

**break**;

}

fprintf(fp,"空间精度：%lf\n",dX);

fprintf(fp,"时间精度：%lf\n",dt);

**if**(switch\_to\_final==1)

{

fprintf(fp,"模拟时长：%lf\n",t);

}

**else** **if**(switch\_to\_final==2)

{

fprintf(fp,"达到稳定态用时：%lf\n",final\_t);

}

**break**;

**case** 3:

fprintf(fp,"仿真方法：中间差分法\n");

**switch** (method) {

**case** 1:

fprintf(fp,"迭代方式：Jacobi法\n");

**break**;

**case** 2:

fprintf(fp,"迭代方式：Gauss-Seidel法\n");

**break**;

**case** 3:

fprintf(fp,"迭代方式：Gauss-Seidel法\n");

fprintf(fp,"松弛因子: %lf",sor\_w);

**break**;

}

fprintf(fp,"空间精度：%lf\n",dX);

fprintf(fp,"时间精度：%lf\n",dt);

**if**(switch\_to\_final==1)

{

fprintf(fp,"模拟时长：%lf\n",t);

}

**else** **if**(switch\_to\_final==2)

{

fprintf(fp,"达到稳定态用时：%lf\n",final\_t);

}

**break**;

**case** 4:

fprintf(fp,"仿真方法：交替方向隐式法\n");

fprintf(fp,"空间精度：%lf\n",dX);

fprintf(fp,"时间精度：%lf\n",dt);

**if**(switch\_to\_final==1)

{

fprintf(fp,"模拟时长：%lf\n",t);

}

**else** **if**(switch\_to\_final==2)

{

fprintf(fp,"达到稳定态用时：%lf\n",final\_t);

}

**break**;

}

**if**(Y\_OR\_N=='Y'||Y\_OR\_N=='y')

{

fprintf(fp,"picture文件中对应的时间分别是：\n");

fprintf(fp,"0. t = 0.000000\n");

**int** i=0;

**while**(i<picture\_frame\_number)

{

fprintf(fp,"%d. t = %lf\n",i+1,picture[i]\*dt);

i++;

}

**if**(switch\_to\_final==1)

{

fprintf(fp,"%d. t = %lf\n",picture\_frame\_number+1,t);

}

**else** **if**(switch\_to\_final==2)

{

fprintf(fp,"%d. t = %lf\n",picture\_frame\_number+1,final\_t);

}

}

fclose(fp);

}

//一下的一些方程为传统的显式和隐式方法

//用于检查方程的解是否已经收敛。

**int** check\_for\_limit(**void**)

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

**if**(fabs(temperature0[a][b][c]-temperature\_cache[a][b][c])>scale)

**return**(1);

**return**(0);

}

**return**(1);

}

//用于检查是否已经达到热传导的稳定态

**int** check\_for\_final(**void**)

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

**if**(fabs(temperature0[a][b][c]-temperature1[a][b][c])>0.0001)

**return**(0);

**return**(1);

}

**return**(0);

}

//用于在控制台打印完成度

**void** print\_complete\_proportion(**int** i)

{

system("clear");

printf("|");

**for**(**int** a=0;a<(**int**)(50\*i/nt);a++)

printf("#");

**int** a=0;

**while**(a<(50-(**int**)(50\*i/nt)-1))

{

printf("\_");

a++;

}

printf("|\n");

printf("已完成%f%%",(100\*(i+1)/t));

}

//计算方法一，显示差分

**void** calculate1(**void**)

{

**for**(**int** i=0;i<nt;i++)

{

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=(1-6\*factor)\*temperature1[a][b][c]+factor\*(temperature1[a+1][b][c]+temperature1[a-1][b][c]+temperature1[a][b+1][c]+temperature1[a][b-1][c]+temperature1[a][b][c+1]+temperature1[a][b][c-1]);

}

**if**(Index==0)

Index=1;

**else** **if**(Index==1)

Index=0;

**int** j=0;

**while**(j<picture\_number)

{

**if**(i==(picture[j]-1))

{

picture\_frame\_number++;

save\_picture(j);

}

j++;

}

**if**(switch\_to\_final==2)

**if**(check\_for\_final())

{

final\_t=(i+1)\*dt;

**break**;

}

**if**(switch\_to\_final==1)

print\_complete\_proportion(i);

}

save\_picture(picture\_frame\_number);

file\_exe1();

}

//计算方法二，隐式差分，j迭代

**void** j\_calculate2(**void**)

{

**for**(**int** i=0;i<nt;i++)

{

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=(temperature0[a+1][b][c]+temperature0[a-1][b][c]+temperature0[a][b+1][c]+temperature0[a][b-1][c]+temperature0[a][b][c+1]+temperature0[a][b][c-1])\*factor/(1.0+6.0\*factor)+temperature1[a][b][c]/(1.0+6.0\*factor);

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=(temperature\_cache[a+1][b][c]+temperature\_cache[a-1][b][c]+temperature\_cache[a][b+1][c]+temperature\_cache[a][b-1][c]+temperature\_cache[a][b][c+1]+temperature\_cache[a][b][c-1])\*factor/(1.0+6.0\*factor)+temperature1[a][b][c]/(1.0+6.0\*factor);

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

Index=1;

**else** **if**(Index==1)

Index=0;

**int** j=0;

**while**(j<picture\_number)

{

**if**(i==(picture[j]-1))

{

picture\_frame\_number++;

save\_picture(j);

}

j++;

}

**if**(switch\_to\_final==2)

**if**(check\_for\_final())

{

final\_t=(i+1)\*dt;

**break**;

}

**if**(switch\_to\_final==1)

print\_complete\_proportion(i);

}

save\_picture(picture\_frame\_number);

file\_exe1();

}

//隐式差分，gs迭代

**void** gs\_calculate2(**void**)

{

**for**(**int** i=0;i<nt;i++)

{

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=temperature1[a][b][c];

temperature\_cache[a][b][c]=temperature1[a][b][c];

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=(temperature0[a+1][b][c]+temperature\_cache[a-1][b][c]+temperature0[a][b+1][c]+temperature\_cache[a][b-1][c]+temperature0[a][b][c+1]+temperature\_cache[a][b][c-1])\*factor/(1.0+6.0\*factor)+temperature1[a][b][c]/(1.0+6.0\*factor);

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=(temperature\_cache[a+1][b][c]+temperature0[a-1][b][c]+temperature\_cache[a][b+1][c]+temperature0[a][b-1][c]+temperature\_cache[a][b][c+1]+temperature0[a][b][c-1])\*factor/(1.0+6.0\*factor)+temperature1[a][b][c]/(1.0+6.0\*factor);

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

Index=1;

**else** **if**(Index==1)

Index=0;

**int** j=0;

**while**(j<picture\_number)

{

**if**(i==(picture[j]-1))

{

picture\_frame\_number++;

save\_picture(j);

}

j++;

}

**if**(switch\_to\_final==2)

**if**(check\_for\_final())

{

final\_t=(i+1)\*dt;

**break**;

}

**if**(switch\_to\_final==1)

print\_complete\_proportion(i);

}

save\_picture(picture\_frame\_number);

file\_exe1();

}

//隐式差分，sor迭代法

**void** sor\_calculate2(**void**)

{

**for**(**int** i=0;i<nt;i++)

{

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=temperature1[a][b][c];

temperature\_cache[a][b][c]=temperature1[a][b][c];

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=sor\_w\*((temperature0[a+1][b][c]+temperature\_cache[a-1][b][c]+temperature0[a][b+1][c]+temperature\_cache[a][b-1][c]+temperature0[a][b][c+1]+temperature\_cache[a][b][c-1])\*factor/(1.0+6.0\*factor)+temperature1[a][b][c]/(1.0+6.0\*factor))+(1-sor\_w)\*temperature0[a][b][c];

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=sor\_w\*((temperature\_cache[a+1][b][c]+temperature0[a-1][b][c]+temperature\_cache[a][b+1][c]+temperature0[a][b-1][c]+temperature\_cache[a][b][c+1]+temperature0[a][b][c-1])\*factor/(1.0+6.0\*factor)+temperature1[a][b][c]/(1.0+6.0\*factor))+(1-sor\_w)\*temperature\_cache[a][b][c];

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

Index=1;

**else** **if**(Index==1)

Index=0;

**int** j=0;

**while**(j<picture\_number)

{

**if**(i==(picture[j]-1))

{

picture\_frame\_number++;

save\_picture(j);

}

j++;

}

**if**(switch\_to\_final==2)

**if**(check\_for\_final())

{

final\_t=(i+1)\*dt;

**break**;

}

**if**(switch\_to\_final==1)

print\_complete\_proportion(i);

}

save\_picture(picture\_frame\_number);

file\_exe1();

}

//中间差分，j迭代法

**void** j\_calculate3(**void**)

{

**for**(**int** i=0;i<nt;i++)

{

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=temperature1[a][b][c];

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=temperature1[a][b][c]\*(2-6\*factor)/(2+6\*factor)+(temperature1[a+1][b][c]+temperature1[a-1][b][c]+temperature1[a][b+1][c]+temperature1[a][b-1][c]+temperature1[a][b][c+1]+temperature1[a][b][c-1])\*factor/(2+6\*factor)+(temperature0[a+1][b][c]+temperature0[a-1][b][c]+temperature0[a][b+1][c]+temperature0[a][b-1][c]+temperature0[a][b][c+1]+temperature0[a][b][c-1])\*factor/(2+6\*factor);

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=temperature1[a][b][c]\*(2-6\*factor)/(2+6\*factor)+(temperature1[a+1][b][c]+temperature1[a-1][b][c]+temperature1[a][b+1][c]+temperature1[a][b-1][c]+temperature1[a][b][c+1]+temperature1[a][b][c-1])\*factor/(2+6\*factor)+(temperature\_cache[a+1][b][c]+temperature\_cache[a-1][b][c]+temperature\_cache[a][b+1][c]+temperature\_cache[a][b-1][c]+temperature\_cache[a][b][c+1]+temperature\_cache[a][b][c-1])\*factor/(2+6\*factor);

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

Index=1;

**else** **if**(Index==1)

Index=0;

**int** j=0;

**while**(j<picture\_number)

{

**if**(i==(picture[j]-1))

{

picture\_frame\_number++;

save\_picture(j);

}

j++;

}

**if**(switch\_to\_final==2)

**if**(check\_for\_final())

{

final\_t=(i+1)\*dt;

**break**;

}

**if**(switch\_to\_final==1)

print\_complete\_proportion(i);

}

save\_picture(picture\_frame\_number);

file\_exe1();

}

//中间差分法，ga迭代

**void** gs\_calculate3(**void**)

{

**for**(**int** i=0;i<nt;i++)

{

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=temperature1[a][b][c];

temperature\_cache[a][b][c]=temperature1[a][b][c];

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=temperature1[a][b][c]\*(2-6\*factor)/(2+6\*factor)+(temperature1[a+1][b][c]+temperature1[a-1][b][c]+temperature1[a][b+1][c]+temperature1[a][b-1][c]+temperature1[a][b][c+1]+temperature1[a][b][c-1])\*factor/(2+6\*factor)+(temperature0[a+1][b][c]+temperature\_cache[a-1][b][c]+temperature\_cache[a][b+1][c]+temperature\_cache[a][b-1][c]+temperature\_cache[a][b][c+1]+temperature\_cache[a][b][c-1])\*factor/(2+6\*factor);

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=temperature1[a][b][c]\*(2-6\*factor)/(2+6\*factor)+(temperature1[a+1][b][c]+temperature1[a-1][b][c]+temperature1[a][b+1][c]+temperature1[a][b-1][c]+temperature1[a][b][c+1]+temperature1[a][b][c-1])\*factor/(2+6\*factor)+(temperature0[a+1][b][c]+temperature0[a-1][b][c]+temperature0[a][b+1][c]+temperature0[a][b-1][c]+temperature0[a][b][c+1]+temperature0[a][b][c-1])\*factor/(2+6\*factor);

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

Index=1;

**else** **if**(Index==1)

Index=0;

**int** j=0;

**while**(j<picture\_number)

{

**if**(i==(picture[j]-1))

{

picture\_frame\_number++;

save\_picture(j);

}

j++;

}

**if**(switch\_to\_final==2)

**if**(check\_for\_final())

{

final\_t=(i+1)\*dt;

**break**;

}

**if**(switch\_to\_final==1)

print\_complete\_proportion(i);

}

save\_picture(picture\_frame\_number+1);

file\_exe1();

}

//计算方法三,sor迭代法

**void** sor\_calculate3(**void**)//计算方法二也即隐式差分法

{

**for**(**int** i=0;i<nt;i++)

{

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=temperature1[a][b][c];

temperature\_cache[a][b][c]=temperature1[a][b][c];

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=sor\_w\*(temperature1[a][b][c]\*(2-6\*factor)/(2+6\*factor)+(temperature1[a+1][b][c]+temperature1[a-1][b][c]+temperature1[a][b+1][c]+temperature1[a][b-1][c]+temperature1[a][b][c+1]+temperature1[a][b][c-1])\*factor/(2+6\*factor)+(temperature0[a+1][b][c]+temperature\_cache[a-1][b][c]+temperature\_cache[a][b+1][c]+temperature\_cache[a][b-1][c]+temperature\_cache[a][b][c+1]+temperature\_cache[a][b][c-1])\*factor/(2+6\*factor))+(1-sor\_w)\*temperature0[a][b][c];

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=sor\_w\*(temperature1[a][b][c]\*(2-6\*factor)/(2+6\*factor)+(temperature1[a+1][b][c]+temperature1[a-1][b][c]+temperature1[a][b+1][c]+temperature1[a][b-1][c]+temperature1[a][b][c+1]+temperature1[a][b][c-1])\*factor/(2+6\*factor)+(temperature0[a+1][b][c]+temperature0[a-1][b][c]+temperature0[a][b+1][c]+temperature0[a][b-1][c]+temperature0[a][b][c+1]+temperature0[a][b][c-1])\*factor/(2+6\*factor))+(1-sor\_w)\*temperature\_cache[a][b][c];

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

Index=1;

**else** **if**(Index==1)

Index=0;

**int** j=0;

**while**(j<picture\_number)

{

**if**(i==(picture[j]-1))

{

picture\_frame\_number++;

save\_picture(j);

}

j++;

}

**if**(switch\_to\_final==2)

**if**(check\_for\_final())

{

final\_t=(i+1)\*dt;

**break**;

}

**if**(switch\_to\_final==1)

print\_complete\_proportion(i);

}

save\_picture(picture\_frame\_number+1);

file\_exe1();

}

//一下为交替方向隐式法(ADI)

**void** conventional\_calculate4(**void**)

{

**for**(**int** i=0;i<nt;i++)

{

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=(factor/(2\*factor+3))\*(temperature0[a-1][b][c]+temperature0[a+1][b][c]+temperature1[a][b-1][c]+temperature1[a][b+1][c]+temperature1[a][b][c-1]+temperature1[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=(factor/(2\*factor+3))\*(temperature\_cache[a-1][b][c]+temperature\_cache[a+1][b][c]+temperature1[a][b-1][c]+temperature1[a][b+1][c]+temperature1[a][b][c-1]+temperature1[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=(factor/(2\*factor+3))\*(temperature1[a-1][b][c]+temperature1[a+1][b][c]+temperature0[a][b-1][c]+temperature0[a][b+1][c]+temperature1[a][b][c-1]+temperature1[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=(factor/(2\*factor+3))\*(temperature1[a-1][b][c]+temperature1[a+1][b][c]+temperature\_cache[a][b-1][c]+temperature\_cache[a][b+1][c]+temperature1[a][b][c-1]+temperature1[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=(factor/(2\*factor+3))\*(temperature1[a-1][b][c]+temperature1[a+1][b][c]+temperature1[a][b-1][c]+temperature1[a][b+1][c]+temperature0[a][b][c-1]+temperature0[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=(factor/(2\*factor+3))\*(temperature1[a-1][b][c]+temperature1[a+1][b][c]+temperature1[a][b-1][c]+temperature1[a][b+1][c]+temperature\_cache[a][b][c-1]+temperature\_cache[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

Index=1;

**else** **if**(Index==1)

Index=0;

**int** j=0;

**while**(j<picture\_number)

{

**if**(i==(picture[j]-1))

{

picture\_frame\_number++;

save\_picture(j);

}

j++;

}

**if**(switch\_to\_final==2)

**if**(check\_for\_final())

{

final\_t=(i+1)\*dt;

**break**;

}

**if**(switch\_to\_final==1)

print\_complete\_proportion(i);

}

save\_picture(picture\_frame\_number);

file\_exe1();

}

**void** brain\_calculate4(**void**)

{

**for**(**int** i=0;i<nt;i++)

{

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=(factor/(2\*factor+3))\*(temperature0[a-1][b][c]+temperature0[a+1][b][c]+temperature1[a][b-1][c]+temperature1[a][b+1][c]+temperature1[a][b][c-1]+temperature1[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=(factor/(2\*factor+3))\*(temperature\_cache[a-1][b][c]+temperature\_cache[a+1][b][c]+temperature1[a][b-1][c]+temperature1[a][b+1][c]+temperature1[a][b][c-1]+temperature1[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=(factor/(2\*factor+3))\*(temperature1[a-1][b][c]+temperature1[a+1][b][c]+temperature0[a][b-1][c]+temperature0[a][b+1][c]+temperature1[a][b][c-1]+temperature1[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=(factor/(2\*factor+3))\*(temperature1[a-1][b][c]+temperature1[a+1][b][c]+temperature\_cache[a][b-1][c]+temperature\_cache[a][b+1][c]+temperature1[a][b][c-1]+temperature1[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

{

temperature0=p1;

temperature1=p0;

}

**else** **if**(Index==1)

{

temperature0=p0;

temperature1=p1;

}

**do**

{

**if**(Index\_cache==0)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature\_cache[a][b][c]=(factor/(2\*factor+3))\*(temperature1[a-1][b][c]+temperature1[a+1][b][c]+temperature1[a][b-1][c]+temperature1[a][b+1][c]+temperature0[a][b][c-1]+temperature0[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**else** **if**(Index\_cache==1)

{

**for**(**int** a=1;a<(nX-1);a++)

**for**(**int** b=1;b<(nY-1);b++)

**for**(**int** c=1;c<(nZ-1);c++)

{

temperature0[a][b][c]=(factor/(2\*factor+3))\*(temperature1[a-1][b][c]+temperature1[a+1][b][c]+temperature1[a][b-1][c]+temperature1[a][b+1][c]+temperature\_cache[a][b][c-1]+temperature\_cache[a][b][c+1])+temperature1[a][b][c]\*((3-4\*factor)/(3+2\*factor));

}

}

**if**(Index\_cache==0)

Index\_cache=1;

**else** **if**(Index\_cache==1)

Index\_cache=0;

}**while**(check\_for\_limit());

**if**(Index==0)

Index=1;

**else** **if**(Index==1)

Index=0;

**int** j=0;

**while**(j<picture\_number)

{

**if**(i==(picture[j]-1))

{

picture\_frame\_number++;

save\_picture(j);

}

j++;

}

**if**(switch\_to\_final==2)

**if**(check\_for\_final())

{

final\_t=(i+1)\*dt;

**break**;

}

**if**(switch\_to\_final==1)

print\_complete\_proportion(i);

}

save\_picture(picture\_frame\_number);

file\_exe1();

}

//计算方法选择器，调用计算函数

**void** calculate\_method\_choice(**void**)

{

**switch** (choice)//选择计算的方式

{

**case** 1:

calculate1();

**break**;

**case** 2:

**switch** (method) {

**case** 1:

j\_calculate2();

**break**;

**case** 2:

gs\_calculate2();

**break**;

**case** 3:

sor\_calculate2();

**break**;

}

**break**;

**case** 3:

**switch** (method) {

**case** 1:

j\_calculate3();

**break**;

**case** 2:

gs\_calculate3();

**break**;

**case** 3:

sor\_calculate3();

**break**;

}

**break**;

**case** 4:

conventional\_calculate4();

**break**;

}

}

//选择作图模式

**void** choice\_draw\_picture(**void**)

{

**do**

{

system("clear");

printf("请选择操作：\n");

printf("1. 查看温度场分布图\n");

printf("2. 退出程序并保存数据\n");

printf("3. 退出程序并删除数据\n");

scanf("%d",&draw\_picture\_number);

**if**(draw\_picture\_number==2)

{

free(p0);

free(p1);

free(temperature\_cache);

exit(0);

}

**else** **if**(draw\_picture\_number==3)

{

**char** fileName[50] = "/Volumes/UDISK/picture0.txt";

**for**(**int** i=0;i<picture\_frame\_number+2;i++)

{

fileName[22]=48+i;

**if**(remove(fileName)!=0)

printf("删除失败！\n");

}

**char** fileName1[50]="/Volumes/UDISK/Information.txt";

**if**(remove(fileName1)!=0)

printf("删除失败！\n");

free(p0);

free(p1);

free(temperature\_cache);

exit(0);

}

fflush(stdin);

}**while**(!(draw\_picture\_number>=1&&draw\_picture\_number<=3));

**do**

{

printf("请选择用于作温度分布图的平面：\n");

printf("1. X-Y 平面\n");

printf("2. Y-Z 平面\n");

printf("3. X-Z 平面\n");

fflush(stdin);

scanf("%d",&draw\_picture\_number);

}**while**(!(draw\_picture\_number>=1&&draw\_picture\_number<=3));

}

//对输入的数据排序

**void** sort\_by\_value(**void**)

{

**double** temp=0.0;

**if**(picture\_number>1)

{

**for**(**int** i=0;i<(picture\_number-1);i++)

**for**(**int** j=0;j<(picture\_number-i-1);j++)

{

**if**(picture[j]>picture[j+1])

{

temp=picture[i];

picture[i]=picture[i+1];

picture[i+1]=temp;

}

}

}

}

//读取已有的数据

**void** read\_condition(**void**)

{

FILE \*fp=**NULL**;

**char** fileName[50] = "/Volumes/UDISK/Information.txt";

fp = fopen(fileName, "r");

fscanf(fp, "%d\t",&picture\_frame\_number);

fscanf(fp, "%lf\n",&dX);

fclose(fp);

}

//输入仿真信息

**void** input\_condition(**void**)

{

printf("作者：侯肇帮\t学号：2021040003\nC语言程序设计课期末大作业\n选题：热传导仿真\n");

printf("=============================================================================\n");

**do**

{

printf("请选择：\n1. 利用现有的数据作图（请将文件保存至指定的路径）\n2. 计算得出新的数据并作图\n");

scanf("%d",&option);

}**while**(!(option==1||option==2));

**if**(option==1)

{

read\_condition();

}

**else** **if**(option==2)

{

printf("请选择差分法的方式：\n1. 显示差分\n2. 隐式差分\n3. 中间差分\n4. 交替方向隐式法\n");

**do**

{

scanf("%d",&choice);

**if**(!(choice==1||choice==2||choice==3||choice==4))

printf("请重新输入！\n");

}**while**(!(choice==1||choice==2||choice==3||choice==4));

printf("请输入空间精度:");

scanf("%lf",&dX);

printf("请输入时间精度:");

scanf("\n%lf",&dt);

printf("请输入热传导系数:");

scanf("\n%lf",&K);

printf("请选择是否计算得出热传导稳定态\n1. 只计算固定时间点以前的热传导\n2. 计算得出热传导稳定态\n");

**do**{

scanf("\n%d",&switch\_to\_final);

}**while**(!(switch\_to\_final==1||switch\_to\_final==2));

**if**(switch\_to\_final==1)

{

printf("请输入模拟的时长:");

scanf("\n%lf",&t);

}

**else** **if**(switch\_to\_final==2)

{

t=1000000000000000;

}

**if**(choice==2||choice==3)

{

**do**

{

printf("请输入你想要的迭代的方法：\n1. Jacobi法\n2. Gauss-Seidel法\n3. Successive Over Relaxation法 \n");

scanf("\n%d",&method);

}**while**(!(method==1||method==2||method==3));

**if**(method==3)

{

**do**

{

printf("请输入松弛因子(0<w<2):");

scanf("\n%lf",&sor\_w);

}**while**(!(sor\_w>0&&sor\_w<2));

}

}

printf("是否需要储存中间数据？(Y/N):");

fflush(stdin);

Y\_OR\_N=getchar();

**if**(Y\_OR\_N=='Y'||Y\_OR\_N=='y')

{

printf("请输入想要存储的中间时间点个数：");

scanf("\n%d",&picture\_number);

picture=(**double** \*)malloc((picture\_number)\***sizeof**(**double**));

**for**(**int** a=0;a<picture\_number;a++)

{

**int** picture\_exe=0;

printf("请输入第%d个时间点(注意时间点会按照时间间隔的选取进行相应的约化处理):",a+1);

**do**

{

fflush(stdin);

scanf("%lf",&picture[a]);

}**while**(!(picture[a]>0&&picture[a]<t));

picture\_exe=picture[a]/dt;

picture[a]=picture\_exe;

}

}

sort\_by\_value();

}

nX=100/dX;

nY=5/dX;

nZ=10/dX;

nt=t/dt;

/\*定义计算数组\*/

//将本次的仿真的参数记录到文件中

}

//开辟动态数组

**void** data\_initial(**void**)

{

//开辟动态数组

p0=(**double** \*\*\*)malloc((nX)\***sizeof**(**double**));

**if**(p0==**NULL**) fprintf(stderr,"error!");

**for**(**int** a=0;a<nX;a++)

{

p0[a]=(**double** \*\*)malloc((nY)\***sizeof**(**double**));

**if**(p0[a]==**NULL**) fprintf(stderr,"error!");

}

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

{

p0[a][b]=(**double** \*)malloc((nZ)\***sizeof**(**double**));

**if**(p0[a][b]==**NULL**) fprintf(stderr,"error!");

}

p1=(**double** \*\*\*)malloc((nX)\***sizeof**(**double**));

**if**(p1==**NULL**) fprintf(stderr,"error!");

**for**(**int** a=0;a<nX;a++)

{

p1[a]=(**double** \*\*)malloc((nY)\***sizeof**(**double**));

**if**(p1[a]==**NULL**) fprintf(stderr,"error!");

}

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

{

p1[a][b]=(**double** \*)malloc((nZ)\***sizeof**(**double**));

**if**(p1[a][b]==**NULL**) fprintf(stderr,"error!");

}

temperature\_cache=(**double** \*\*\*)malloc((nX)\***sizeof**(**double**));

**if**(temperature\_cache==**NULL**) fprintf(stderr,"error!");

**for**(**int** a=0;a<nX;a++)

{

temperature\_cache[a]=(**double** \*\*)malloc((nY)\***sizeof**(**double**));

**if**(temperature\_cache[a]==**NULL**) fprintf(stderr,"error!");

}

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

{

temperature\_cache[a][b]=(**double** \*)malloc((nZ)\***sizeof**(**double**));

**if**(temperature\_cache[a][b]==**NULL**) fprintf(stderr,"error!");

}

}

//温度分布的初始化

**void** temperature\_initial(**void**)

{

factor=dt\*K/(dX\*dX);

scale=factor\*0.000001;

/\*温度的初始化\*/

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

**for**(**int** c=0;c<nZ;c++)

{

p0[a][b][c]=50.0;//默认整个物体初始是为60度的常温

p1[a][b][c]=50.0;

temperature\_cache[a][b][c]=50.0;

}

//计算常数因子

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

**for**(**int** c=0;c<nZ;c++)

{

**if**(a==0)

{//左右

p0[a][b][c]=100.0;//ABGE有一个温度为100度均匀气体持续加热

p1[a][b][c]=100.0;

temperature\_cache[a][b][c]=100.0;

}

**else** **if**(a==(nX-1))

{

p0[a][b][c]=20.0;//DFHC保持常温20度

p1[a][b][c]=20.0;

temperature\_cache[a][b][c]=20.0;

}

}

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

**for**(**int** c=0;c<nZ;c++)

{

**if**(c==0||c==(nZ-1))

{

p0[a][b][c]=100-a\*75/(nX-1);

p1[a][b][c]=100-a\*75/(nX-1);

temperature\_cache[a][b][c]=100-a\*75/(nX-1);

}

}

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

**for**(**int** c=0;c<nZ;c++)

{

**if**(b==0)

{

p0[a][b][c]=p0[a][b][nZ-1]\*(0.2+0.8\*c/(nZ-1));

p1[a][b][c]=p1[a][b][nZ-1]\*(0.2+0.8\*c/(nZ-1));

temperature\_cache[a][b][c]=temperature\_cache[a][b][nZ-1]\*(0.2+0.8\*c/(nZ-1));

}

**else** **if**(b==(nY-1))

{

p0[a][b][c]=40.0;//上表面AEFD温度保持40度

p1[a][b][c]=40.0;

temperature\_cache[a][b][c]=40.0;

}

}

//将初始化的温度存储

FILE \*fp=**NULL**;

**char** fileName[50] = "/Volumes/UDISK/picture0.txt";

fp = fopen(fileName, "w");

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

**for**(**int** c=0;c<nZ;c++)

{

fprintf(fp,"%lf\n",p0[a][b][c]);

}

fclose(fp);

}

//计算初始化

**void** calculate\_initial(**void**)

{

input\_condition();

data\_initial();

**if**(option!=1)

{

temperature\_initial();

}

}

//后面为OpenGL作图函数

//读取已有图像数据

**void** load\_data(**int** i,**double** \*\*\*p)

{

FILE \*fp=**NULL**;

**char** fileName[50] = "/Volumes/UDISK/picture0.txt";

fileName[22]=i+48;

fp = fopen(fileName, "r");

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

**for**(**int** c=0;c<nZ;c++)

{

fscanf(fp,"%lf\n",&p[a][b][c]);

}

fclose(fp);

}

//读取左中右三个图片

**void** load\_picture(**void**)

{

**if**(data.center\_horizontal>0&&data.center\_horizontal<(data.center\_horizontal\_MAX-1))

{

load\_data(data.center\_horizontal,temperature\_cache);

load\_data(data.center\_horizontal-1,p0);

load\_data(data.center\_horizontal+1,p1);

}

**else** **if**(data.center\_horizontal==0)

{

load\_data(data.center\_horizontal,temperature\_cache);

load\_data(data.center\_horizontal+1,p1);

}

**else** **if**(data.center\_horizontal==(data.center\_horizontal\_MAX-1))

{

load\_data(data.center\_horizontal,temperature\_cache);

load\_data(data.center\_horizontal-1,p0);

}

}

//作图

**void** draw\_quads(**int** a,**int** b,**double** temperature)

{

glColor3f(temperature/100.0, 0.0f, (100.0-temperature)/100.0);

glBegin(GL\_QUADS);

glVertex3f(a\*data.pixel\_scale, b\*data.pixel\_scale, 0.0f);

glVertex3f(a\*data.pixel\_scale, (b+1)\*data.pixel\_scale, 0.0f);

glVertex3f((a+1)\*data.pixel\_scale, (b+1)\*data.pixel\_scale, 0.0f);

glVertex3f((a+1)\*data.pixel\_scale, b\*data.pixel\_scale, 0.0f);

glEnd();

}

//匹配作图截面

**void** draw(**int** center\_x, **int** center\_y,**double** \*\*\*p)

{

**switch**(draw\_picture\_number){

**case** 1:

{

**for**(**int** a=0;a<nX;a++)

**for**(**int** b=0;b<nY;b++)

{

draw\_quads(a,b,p[a][b][center\_y]);

}

}

**break**;

**case** 2:

{

**for**(**int** b=0;b<nY;b++)

**for**(**int** c=0;c<nZ;c++)

{

draw\_quads(c, b, p[center\_y][b][c]);

}

}

**break**;

**case** 3:

{

**for**(**int** c=0;c<nZ;c++)

**for**(**int** a=0;a<nX;a++)

{

draw\_quads(a, c, p[a][center\_y][c]);

}

}

**break**;

**default**:

**break**;

}

}

//作图函数

**void** myDisplay(**void**)

{

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

**if**(data.center\_horizontal>0)

{

glPushMatrix();

glTranslatef(data.delta\_x, data.delta\_y, 0.0f);

glTranslatef(-200.0f, 0.0f, -100.0f);

draw(data.center\_horizontal-1,data.center\_vertical,p0);

glPopMatrix();

}

**if**(data.center\_horizontal<(data.center\_horizontal\_MAX-1))

{

glPushMatrix();

glTranslatef(data.delta\_x, data.delta\_y, 0.0f);

glTranslatef(200.0f, 0.0f, -100.0f);

draw(data.center\_horizontal+1,data.center\_vertical,p1);

glPopMatrix();

}

**if**(data.center\_vertical>0)

{

glPushMatrix();

glTranslatef(data.delta\_x, data.delta\_y, 0.0f);

glTranslatef(0.0f, -100.0f, -100.0f);

draw(data.center\_horizontal,data.center\_vertical-1,temperature\_cache);

glPopMatrix();

}

**if**(data.center\_vertical<(data.center\_vertical\_MAX-1))

{

glPushMatrix();

glTranslatef(data.delta\_x, data.delta\_y, 0.0f);

glTranslatef(0.0f, 100.0f, -100.0f);

draw(data.center\_horizontal,data.center\_vertical+1,temperature\_cache);

glPopMatrix();

}

glPushMatrix();

glTranslatef(data.delta\_x, data.delta\_y, 0.0f);

draw(data.center\_horizontal,data.center\_vertical,temperature\_cache);

//1draw\_axis();

glPopMatrix();

glFlush();

glutSwapBuffers();

}

//调整窗口投影比例

**void** changeSize(**int** w, **int** h)

{

**if**(h == 0)

h = 1;

**float** ratio = 1.0\* w / h;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glViewport(0, 0, w, h);

gluPerspective(45.0f, ratio, 0.1f, 10000.0f);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(0.0f, 0.0f, 500.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);

}

//检测按键操作

**void** inputKey(**int** key, **int** x, **int** y)

{

**switch** (key)

{

**case** GLUT\_KEY\_UP :

**if**(data.center\_vertical<(data.center\_vertical\_MAX-1))

{

data.center\_vertical++;

}

**break**;

**case** GLUT\_KEY\_DOWN :

**if**(data.center\_vertical>0)

{

data.center\_vertical--;

}

**break**;

**case** GLUT\_KEY\_LEFT:

**if**(data.center\_horizontal>0)

{

data.center\_horizontal--;

load\_picture();

}

**break**;

**case** GLUT\_KEY\_RIGHT:

**if**(data.center\_horizontal<(data.center\_horizontal\_MAX-1))

{

data.center\_horizontal++;

load\_picture();

}

**break**;

}

}

//检测按键操作

**void** processNormalKeys(**unsigned** **char** key,**int** x,**int** y)

{

**if**(key==27)

exit(0);

}

//初始化作图信息

**void** init\_graphics(**void**)

{

**switch**(draw\_picture\_number){

**case** 1:

data.center\_horizontal\_MAX=picture\_frame\_number+2;

data.center\_vertical\_MAX=nZ;

data.pixel\_scale=5\*dX;

data.delta\_x=-(nX/2)\*data.pixel\_scale;

data.delta\_y=-(nY/2)\*data.pixel\_scale;

**break**;

**case** 2:

data.center\_horizontal\_MAX=picture\_frame\_number+2;

data.center\_vertical\_MAX=nX;

data.pixel\_scale=15\*dX;

data.delta\_x=-(nZ/2)\*data.pixel\_scale;

data.delta\_y=-(nY/2)\*data.pixel\_scale;

**break**;

**case** 3:

data.center\_horizontal\_MAX=picture\_frame\_number+2;

data.center\_vertical\_MAX=nY;

data.pixel\_scale=5\*dX;

data.delta\_x=-(nX/2)\*data.pixel\_scale;

data.delta\_y=-(nZ/2)\*data.pixel\_scale ;

**break**;

}

load\_picture();

}

//作图菜单的元素

**enum** menu {up,down,left,right,xy,yz,zx,quit};

**enum** menu MEUN;

//处理右键菜单

**void** processMenuEvents(**int** option)

{

**switch** (option) {

**case** up :

**if**(data.center\_vertical<data.center\_vertical\_MAX)

{

data.center\_vertical++;

}

**break**;

**case** down :

**if**(data.center\_vertical>0)

{

data.center\_vertical--;

}

**break**;

**case** left :

**if**(data.center\_horizontal>0)

{

data.center\_horizontal--;

load\_picture();

}

**break**;

**case** right:

**if**(data.center\_horizontal<data.center\_horizontal\_MAX)

{

data.center\_horizontal++;

load\_picture();

}

**break**;

**case** xy:

draw\_picture\_number=1;

data.center\_vertical=1;

data.center\_horizontal=1;

init\_graphics();

**break**;

**case** yz:

draw\_picture\_number=2;

data.center\_vertical=1;

data.center\_horizontal=1;

init\_graphics();

**break**;

**case** zx:

draw\_picture\_number=3;

data.center\_vertical=1;

data.center\_horizontal=1;

init\_graphics();

**break**;

**case** quit:

free(p0);

free(p1);

free(temperature\_cache);

exit(0);

**break**;

}

}

//定义右键菜单

**void** createGLUTMenus(**void**)

{

// 创建菜单并告诉 GLUT， processMenuEvents 处理菜单事件。

MEUN = glutCreateMenu(processMenuEvents);

//给菜单增加条目

glutAddMenuEntry("up",up);

glutAddMenuEntry("down",down);

glutAddMenuEntry("left",left);

glutAddMenuEntry("right",right);

glutAddMenuEntry("xy", xy);

glutAddMenuEntry("yz", yz);

glutAddMenuEntry("zx", zx);

glutAddMenuEntry("quit", quit);

// 把菜单和鼠标右键关联起来。

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

//主函数

**int** main(**int** argc, **char** \*argv[])

{

//计算热传导初始化

calculate\_initial();

//匹配调用计算函数

calculate\_method\_choice();

//选择作图模式

choice\_draw\_picture();

//初始化作图

init\_graphics();

//OpenGL作图初始化

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowPosition(10, 10);

glutInitWindowSize(1400,800);

glutCreateWindow("热传导数值仿真");

//OpenGL按键操作

glutIgnoreKeyRepeat(1);

glutSpecialFunc(inputKey);

glutKeyboardFunc(processNormalKeys);

//OpenGL作图

glutDisplayFunc(myDisplay);

glutReshapeFunc(changeSize);

glutIdleFunc(myDisplay);

glEnable(GL\_DEPTH\_TEST);

//OpenGL构建右键菜单

createGLUTMenus();

//OpenGL循环作图

glutMainLoop();

**return** 0;

}