#include <stdlib.h>

#include "..\ExternDoc.h"

#include "..\UI\Zoom.h"

#include "..\UI\MsgView.h"

#include "..\Graphics\DrawFunc.h"

#include "..\Example\Example.h"

#define ERROR\_NUMBER -1

typedef struct \_node

{

int row;

int col;

int right;

int down;

int left;

int up;

} room;

room\*\* maze;

int rcnt=0;

int ccnt=0;

//function prototype

static void drawDirect(CDC \*pDC);

static void drawBuffered();

//Start of user code

#include <float.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* function : bool readFile(const char\* filename)

\* argument : cons char\* filename - filename to be opened

\* return : true if success, otherwise flase

\* remark : After read data file, phisycal view must be set;

\* otherwise drawing will not be executed correctly.

\* The window will be invalidated after readFile()

\* returns true.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool readFile(const char\* filename){

//start of the user code

char ch;

int i=0,j=0;

FILE\* fp = fopen( filename, "r" );

if( fp == NULL ) return false;

while(fscanf(fp,"%c",&ch) != EOF&&(ch != '\n'))

{

if(ch == '-')

{

++ccnt;

}

}

while(fscanf(fp,"%c",&ch) != EOF)

{

if(ch == '\n')

{

++rcnt;

}

}

rcnt/=2;

rewind(fp);

maze = (room\*\*)malloc(sizeof(room\*)\*rcnt);

for(i=0;i<rcnt;++i)

{

maze[i] = (room\*)malloc(sizeof(room)\*ccnt);

}

while(fscanf(fp,"%c",&ch) != EOF)

{

if(ch == '+')

{

continue;

}

else if(ch == '\n')

{

break;

}

else if(ch == '-')

{

maze[0][j].up = 1;

}

}

i=0;

while(fscanf(fp,"%c",&ch) != EOF)

{

if(i == rcnt)

{

break;

}

if(ch == '\n')

{

j=0;

continue;

}

else if(ch == ' ')

{

fscanf(fp,"%c",&ch);

if(ch == ' ')

{

maze[i][j].right=0;

}

else

{

maze[i][j].right=1;

}

++j;

}

else if(ch == '+')

{

fscanf(fp,"%c",&ch);

if(ch == '\n')

{

++i;

j=0;

continue;

}

if(ch == ' ')

{

maze[i][j].down=0;

}

else

{

maze[i][j].down=1;

}

++j;

}

}

for(j=0;j<ccnt;++j)

{

maze[rcnt-1][j].down = 1;

}

for(i=1;i<rcnt;++i)

{

maze[i][0].left = 1;

for(j=1;j<ccnt;++j)

{

maze[i][j].left = maze[i][j-1].right;

maze[i][j].up = maze[i-1][j].down;

}

}

fclose( fp );

setWindow(0,0,rcnt,ccnt,1);

return true; //edit after finish this function

//end of usercode

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* function : bool FreeMemory()

\*

\* remark : Save user data to a file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void freeMemory(){

//start of the user code

rcnt=0;

ccnt=0;

//end of usercode

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* function : bool writeFile(const char\* filename)

\*

\* argument : const char\* filename - filename to be written

\* return : true if success, otherwise false

\* remark : Save user data to a file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool writeFile(const char\* filename){

//start of the user code

bool flag;

flag = 0;

return flag;

//end of usercode

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* fucntion : void drawMain(CDC\* pDC)

\*

\* argument : CDC\* pDC - device context object pointer

\* remark : Main drawing function. Called by CMFC\_MainView::OnDraw()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void drawMain(CDC \*pDC){

//if direct drawing is defined

#if defined(GRAPHICS\_DIRECT)

drawDirect(pDC);

//if buffered drawing is defined

#elif defined(GRAPHICS\_BUFFERED)

drawBuffered();

#endif

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* function : static void drawDirect(CDC \*pDC

\*

\* argument : CDC\* pDC - device context object pointer

\* remark : Direct drawing routines here.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void drawDirect(CDC \*pDC){

//begin of user code

//Nothing to write currently.

//end of user code

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* function : static void drawBuffered()

\*

\* argument : CDC\* pDC -0 device object pointer

\* remark : Buffered drawing routines here.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void drawBuffered(){

//start of the user code

int i,j;

for(j=0;j<ccnt;++j)

{

DrawLine\_I(j,0,j+1,0,1,RGB(128,128,128));

DrawLine\_I(j+1,rcnt-1,j+1,rcnt-1,1,RGB(128,128,128));

}

for(i=0;i<rcnt;++i)

{

DrawLine\_I(0,0,0,rcnt,1,RGB(128,128,128));

DrawLine\_I(ccnt,0,ccnt,rcnt,1,RGB(128,128,128));

}

for(i=0;i<rcnt;++i)

{

for(j=0;j<ccnt;++j)

{

if(maze[i][j].right)

{

DrawLine\_I(j+1,i,j+1,i+1,1,RGB(128,128,128));

}

if(maze[i][j].down)

{

DrawLine\_I(j,i+1,j+1,i+1,1,RGB(128,128,128));

}

}

}

//end of the user code

}