同濟大學
TONGJI UNIVERSITY

高级语言程序设计大作业

装

订

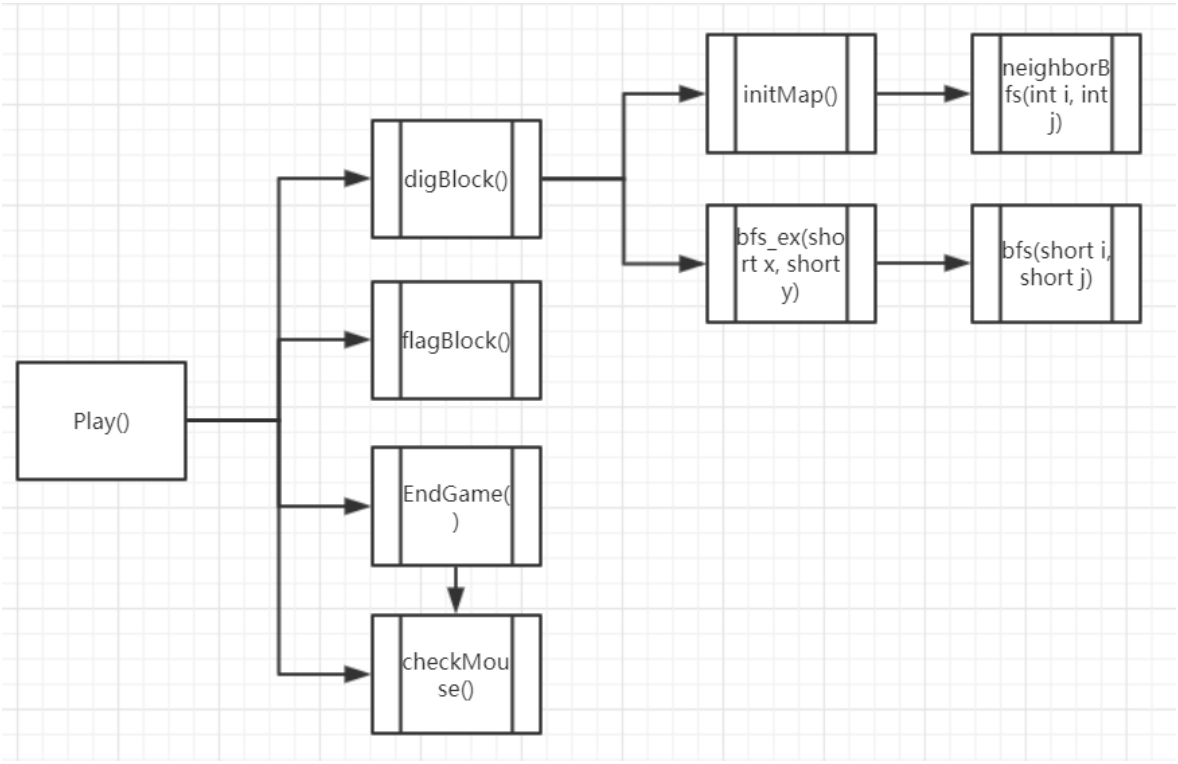
线

设计思路与功能描述

一、设计思路：

本次大作业主要在于读懂原给的代码，并对 GameEngine.cpp 文件进行改写和扩充。

本次大作业进行改写及补充的函数有 neighborBfs(int i, int j)，initMap()，bfs(short i, short j)，bfs_ex(short x, short y)，digBlock()，flagBlock()，checkMouse()，EndGame()，Play()，共 9 个。函数间调用关系为：



二、功能描述：

1. play()：开始游戏

第一部分为程序中所需的变量初始化，以便可以重复游戏。

第二部分为游戏开始，为原给的代码。

第三部分为显示已走步数、剩余雷数、游戏时间。利用 MovePos()、sprintf_s()、PutString()。实现显示。其中在时间显示部分，通过调用 GetTickCount64()，实现时间显示。

第四部分为判断游戏是否失败。如果失败，会显示全部地雷（在 digblock()部分实现），同时出现汉字表明游戏失败，调用 checkMouse()，单击鼠标左键回到主菜单。

第五部分为判断游戏是否胜利结束。如果结束，调用 EndGame()，同时出现汉字表明游戏胜利，并显示走的步数和游戏时长，调用 checkMouse()，单击鼠标左键回到主菜单。

第六部分为处理每帧的事务，为原给的代码。

2. digBlock()：初始化地图以及翻开某位置

第一部分为初始化地图。调用 initMap()，实现初始化操作。之后观察这个位置是空，还是数字。如果是数字，则只翻开这个位置。如果是空，则调用 bfs_ex(short x,short y)，之后在其中调用 bfs(short i,short j)，实现完善相邻格的搜索，对连续的空白一并翻开，直到遇到数字的边界。

第二部分为不是第一步的情况。如果点到地雷，则通过对全图遍历，找到所有地雷并显示。

如果点到数字，则只翻开这个位置。如果点到空，则调用 bfs_ex(short x,short y)，之后在其中调用 bfs(short i,short j)，实现完善相邻格的搜索，对连续的空白一并翻开，直到遇到数字的边界。

3. initMap()：初始化地图

通过用时间作种子，调用随机数函数，得到十个地雷的位置。之后调用 neighborBfs(int i, int j) 函数，找到每个位置周边的雷数，实现全图的初始化。

4. neighborBfs(int i, int j)：找每个位置周边的雷数

通过 initMap()传入的位置 i、j，在其八邻域中找是否有雷，如果有雷，则该位置数字加 1，当然首先要判断八邻域是否越界。

5. bfs_ex(short x, short y)：BFS 判断（i,j）及周边是否为雷

首先这个位置一定为空，所以把位置放入队列。在队列不空的情况下，调用 bfs(short i, short j)，判断该位置八邻域是否为空或数字。最终达到完善相邻格的搜索，对连续的空白一并翻开，直到遇到数字的边界。

6. bfs(short i, short j)：BFS 判断（i,j）的八邻域是否要被翻开

在该位置没有越界并没有被访问过的情况下，进一步判断。若该位置为数字，则到此为止。若该位置为空，则将其位置放入队列，用于之后的进一步 BFS。

7. flagBlock()：为该位置标记旗子或撤掉旗子

如果该位置没有旗子，并且没有被访问过，则在该位置标记上旗子，并且剩余雷数减 1。如果该位置有旗子，则则在该位置撤掉旗子，并且剩余雷数加 1。

8. EndGame()：胜利结束游戏的后续处理

清屏，出现汉字表明游戏胜利，并显示走的步数和游戏时长，调用 checkMouse()，单击鼠标左键回到主菜单。

9. checkMouse()：检查是否有鼠标单击，用来结束游戏

调用 FlushInput()，刷新输入缓冲区，pos = GetCursorPos()，获取鼠标输入，GetCursorHitPos()，获取鼠标单击输入，如果 hitLeftPos.X == pos.X && hitLeftPos.Y == pos.Y，则有鼠标左击。

在实验过程中遇到的问题及解决方法

一、原给代码多，函数间调用关系复杂，使用的库函数多，刚开始做时理解较难。
解决方法：通过画图，厘清函数间拓扑关系，以及单步调试等理解程序。

二、剩余雷数显示不正常，由 10 变为 90。
解决方法：发现原因是在使用 sprintf_s、PutString 函数时出现字符串覆盖不完全的情况，即 9 这个字符覆盖了 1，却保留了 0，导致由 10 变为 90。解决方法为在前面加空格（这也是程序编写者让思考的那个问题）。

三、在游戏中点击某个位置，这个位置未发生变化。
解决方法：发现原因是 posChoice.X, posChoice.Y，对应的是数组的第 Y 行，第 X 列。所以把 X 和 Y 交换过来即可。

四、第一次游戏结束后，第二次游戏初始化以及游戏过程中出现问题。
解决方法：发现原因为未进行初始化。注意在每次游戏开始时对所涉及的变量等进行初始化操作。

五、时间显示出现问题，在第二次游戏时时间会变的极大。
解决方法：通过加入一个 GameStart 变量，使得在游戏开始时才开始显示时间，最终时间显示正常。

心得体会

一、程序和算法设计的体会

1. 程序设计中代码规范非常重要，其中代码格式规范、变量命名规范、分块编写规范、注释规范等都需要注意。

2. 程序设计要注重完备性、鲁棒性。对于程序要实现的功能，尽量写完整，写全面。对变量的处理，对指针的使用和回收都要注意。

3. 算法设计中使用搜索时，尽量不要使用递归操作，避免栈溢出。

4. 合理使用库函数。库函数具有封装好、效率高、bug 少的特点，使用库函数可以提高开发效率、减少臃肿的代码。但同时，对于涉及重要的数据结构和算法相关知识的部分，还是应当首先自己了解、熟悉并掌握，之后再酌情使用库函数。

一、编程过程的体会：

1. 实际工程中的代码量往往很大，如何快速理清代码结构、找到自己需要的部分，非常重要。所以要提高阅读代码的呃能力。

2. 编程过程中要善于 Debug。程序中出现 bug 非常正常，通过断点、单步调试、变量监控等往往能快速锁定 bug。

3. 搜索引擎是自学过程中的好老师。在遇到自己不了解的部分时，可以去 cppreference、微软 c++ 文档、csdn、stackoverflow 等地方寻找解决方法。

源代码：

一、GameEngine.cpp 部分:

```
// 如果你有新加的变量，建议加在下方和预置变量做区别
    queue<short> queBfs;                    // 用于BFS的队列，存放位置x,y
    int idx[100];                           // 判断每个位置是否进行过BFS
    int mapArraytrue[100];                   // 实际操作的数组
    int mineSum;                            // 地雷个数
    int timeStart;                          // 游戏开始时间
    int timeEnd;                            // 游戏结束时间
    int gameStart;                          // 判断游戏是否开始
    int gameEnd;                            // 判断游戏是否结束
    int randomCount;                        // 随机生成的地雷计数器
    int stepCount;                          // 游戏步数计数器
    char strMine[32] = "";                    // 用于打印雷数
    char strStep[32] = "";                    // 用于打印步数
    char strTime[32] = "";                    // 用于打印时间
    int clickMouse = 0;                     // 用于结束游戏
```

```
    int gameFail = 0;                                   // 判断游戏是否失败

/****************************************
    Function:    neighborBfs()
    Parameter: int i int j
    Return:       None(void)
    Description:
    找到每个位置周边的雷数
    ****************************************/
    void neighborBfs(int i, int j)
    {
        //当前位置的左上位置有雷则当前位置加 1，下面的同理
        if (i - 1 >= 0 && j - 1 >= 0 && mapArraytrue[(i - 1) * 10 + j - 1] == 9)
            mapArraytrue[i * 10 + j]++;
        if (i - 1 >= 0 && mapArraytrue[(i - 1) * 10 + j] == 9)
            mapArraytrue[i * 10 + j]++;
        if (j - 1 >= 0 && mapArraytrue[i * 10 + j - 1] == 9)
            mapArraytrue[i * 10 + j]++;
        if (i - 1 >= 0 && j + 1 <= 9 && mapArraytrue[(i - 1) * 10 + j + 1] == 9)
            mapArraytrue[i * 10 + j]++;
        if (i + 1 <= 9 && j - 1 >= 0 && mapArraytrue[(i + 1) * 10 + j - 1] == 9)
            mapArraytrue[i * 10 + j]++;
        if (i + 1 <= 9 && j + 1 <= 9 && mapArraytrue[(i + 1) * 10 + j + 1] == 9)
            mapArraytrue[i * 10 + j]++;
        if (i + 1 <= 9 && mapArraytrue[(i + 1) * 10 + j] == 9)
            mapArraytrue[i * 10 + j]++;
        if (j + 1 <= 9 && mapArraytrue[i * 10 + j + 1] == 9)
            mapArraytrue[i * 10 + j]++;
    }

    /****************************************
    Function:    initMap()
    Parameter: None(void)
    Return:       None(void)
    Description:
    初始化地图
    ****************************************/
    void initMap() {
        //memset(mapArray, 0, sizeof(mapArray));        //mapArray 清零
        memset(mapCanvas, 0, sizeof(mapCanvas));        //mapArray 清零
        memset(mapArraytrue, 0, sizeof(mapArraytrue));   //mapArraytrue 清零
        srand(time(NULL));                              //时间做种子
        randomCount = 0;                                //随机生成的地雷数初始化

        //随机位置生成地雷
        for (int i = 0;; i++)
        {
            int x = rand() % 10;
            int y = rand() % 10;
            if ((x != posChoice.Y || y != posChoice.X) && mapArraytrue[x * 10 + y] == 0)
            {
                mapArraytrue[x * 10 + y] = 9;
                randomCount++;
```

```
            }
            if (randomCount == 10)
                break;
        }

        //为整个地图初始化
        for (int i = 0; i < 10; i++)
            for (int j = 0; j < 10; j++)
            {
                if (mapArraytrue[i * 10 + j] != 9)
                    neighborBfs(i, j);
            }
}

/*****************************************
Function:    bfsNext()
Parameter: short i short j
Return:       None(void)
Description:
BFS 判断（i,j）的八邻域是否要被翻开
*****************************************/
void bfsNext(short i, short j)
{
        if (i < 0 || i>9 || j < 0 || j>9 || idx[i * 10 + j] != 0)//越界或者已经访问过
            return;
        else if (mapArraytrue[i * 10 + j] != 0)                    //该位置为数字的边界
        {
            mapCanvas[i * 10 + j] = mapArraytrue[i * 10 + j];
            idx[i * 10 + j] = 1;
        }
        else                                                       //该位置为空白
        {
            mapCanvas[i * 10 + j] = 10;
            queBfs.push(i);
            queBfs.push(j);
            idx[i * 10 + j] = 1;
        }
}

/*****************************************
Function:    bfsFirst()
Parameter: short x short y
Return:       None(void)
Description:
bfsNext 判断（i,j）是否为雷
*****************************************/
void bfsFirst(short x, short y)
{
        mapCanvas[x * 10 + y] = 10;
        queBfs.push(x);
        queBfs.push(y);
        idx[x * 10 + y] = 1;

        //判断该位置的八邻域
```

```
        while (!queBfs.empty())
        {
            short i = queBfs.front();
            queBfs.pop();
            short j = queBfs.front();
            queBfs.pop();
            bfsNext(i - 1, j - 1);
            bfsNext(i - 1, j);
            bfsNext(i - 1, j + 1);
            bfsNext(i, j - 1);
            bfsNext(i, j + 1);
            bfsNext(i + 1, j - 1);
            bfsNext(i + 1, j);
            bfsNext(i + 1, j + 1);
        }
    }

    /**************************************
    Function:    digBlock()
    Parameter: None(void)
    Return:      None(void)
    Description:
    初始化地图以及翻开某位置
    ****************************************/
    void digBlock() {
        if (isFirst) {
            // 如果是第一步走，则先初始化地图，注意不要在落点设置一个地雷
            initMap();
            isFirst = false; // 将第一步设置为否
            memset(idx, 0, sizeof(idx));//idx 数组清零
            if (mapArraytrue[posChoice.Y * 10 + posChoice.X] == 0)//如果该位置为空白
                bfsFirst(posChoice.Y, posChoice.X);
            else if (mapArraytrue[posChoice.Y * 10 + posChoice.X] > 0 &&
mapArraytrue[posChoice.Y * 10 + posChoice.X] < 9)//如果该位置为数字
            {
                mapCanvas[posChoice.Y * 10 + posChoice.X] = mapArraytrue[posChoice.Y * 10 +
posChoice.X];
                idx[posChoice.Y * 10 + posChoice.X] = 1;
            }
            //updateMap(); // 更新地图画板
            renderMap(); // 绘制当前地图
            gameStart = 1;//游戏开始
            timeStart = GetTickCount64();//开始计时
            mineSum = 10;//雷数为 10
            stepCount = 1;//步数为 0
            return;
        }

        //不是第一步
        if (mapArraytrue[posChoice.Y * 10 + posChoice.X] == 9)//如果该位置为雷
        {
            stepCount++;
            mapCanvas[posChoice.Y * 10 + posChoice.X] = 9;
```

```
                for (int i = 0; i < 10; i++)//显示全部的雷，结束游戏
                    for (int j = 0; j < 10; j++)
                    {
                        if (mapArraytrue[i * 10 + j] == 9)
                            mapCanvas[i * 10 + j] = 9;
                    }

                //updateMap(); //  更新地图画板
                renderMap(); //  绘制当前地图
                gameFail = 1;
            }
            else if (mapArraytrue[posChoice.Y * 10 + posChoice.X] > 0 && mapArraytrue[posChoice.Y * 10 + posChoice.X] < 9 && idx[posChoice.Y * 10 + posChoice.X] ==0)//如果该位置为数字
            {
                stepCount++;
                mapCanvas[posChoice.Y * 10 + posChoice.X] = mapArraytrue[posChoice.Y * 10 + posChoice.X];
                idx[posChoice.Y * 10 + posChoice.X] = 1;
            }
            else if(mapArraytrue[posChoice.Y * 10 + posChoice.X] ==0 && idx[posChoice.Y * 10 + posChoice.X] == 0)//如果该位置为空白
            {
                stepCount++;
                bfsFirst(posChoice.Y, posChoice.X);
            }

            //
            //updateMap(); //  更新地图画板
            renderMap(); //  绘制当前地图
        }

        /**************************************
        Function:    flagBlock()
        Parameter: None(void)
        Return:       None(void)
        Description:
        为该位置标记旗子或撤掉旗子
        **************************************/
        void flagBlock() {
            //插旗子或撤销旗子
            if (mineSum != 0 && mapCanvas[posChoice.Y * 10 + posChoice.X] == 0 || mapCanvas[posChoice.Y * 10 + posChoice.X] == 9)//如果这里没有旗子
            {
                stepCount++;
                mapCanvas[posChoice.Y * 10 + posChoice.X] = 11;
                idx[posChoice.Y * 10 + posChoice.X] = 1;
                mineSum--;//雷数减 1
            }

            //如果这里有旗子
            else if (mapCanvas[posChoice.Y * 10 + posChoice.X] == 11)
            {
                stepCount++;
```

```
            mapCanvas[posChoice.Y * 10 + posChoice.X] = 0;
            idx[posChoice.Y * 10 + posChoice.X] = 0;
            mineSum++;//雷数加 1
        }

        //
        //updateMap(); //  更新地图画板
        renderMap(); //  绘制当前地图
    }

    /****************************************
    Function:    checkMouse()
    Parameter: None(void)
    Return:       None(void)
    Description:
    检查是否有鼠标单击，用来结束游戏
    ****************************************/
    void checkMouse()
    {
        clickMouse = 0;//没有点击鼠标
        FlushInput();//  刷新输入缓冲区
        pos = GetCursorPos();//  获取鼠标输入
        COORD hitLeftPos = GetCursorHitPos();//  获取鼠标单击输入
        if (hitLeftPos.X == pos.X && hitLeftPos.Y == pos.Y) {
            clickMouse = 1;//完成单击鼠标
        }
    }

    /****************************************
    Function:    endGame()
    Parameter: None(void)
    Return:       None(void)
    Description:
    胜利结束游戏的后续处理
    ****************************************/
    void endGame()
    {
        ClearScreen();//清屏
        MovePos(14, 2); //  移动坐标到  14,2
        PutString("你赢了！ ");
        MovePos(14, 4); //  移动坐标到  14,4
        PutString(strStep);
        MovePos(14, 6); //  移动坐标到  14,6
        PutString(strTime);
        MovePos(14, 8); //  移动坐标到  14,8
        PutString("鼠标左击返回主菜单");
        Update();//  将地图更新到屏幕
        bool runFlag = true;//判断有没有点击鼠标
        while (runFlag) {
            checkMouse();    //  检查输入
            if (clickMouse == 1)
                runFlag = false;
```

```
        }
    }

    /****************************************
    Function:    Play()
    Parameter: None(void)
    Return:       None(void)
    Description:
    开始游戏
    ****************************************/
    void Play() {
        //主要变量的初始化
        timeStart = 0;
        timeEnd = 0;
        gameFail = 0;
        stepCount = 0;
        gameStart = 0;
        isFirst = true;
        memset(strMine, 0, sizeof(strMine));
        memset(strStep, 0, sizeof(strStep));
        memset(strTime, 0, sizeof(strTime));

        //游戏开始
        gameFlag = true;
        while (gameFlag) {
            checkChoice(); //  检查输入

            //  查看当前坐标是否需要更新背景
            if (posChoice.X != posChoiceOld.X || posChoice.Y != posChoiceOld.Y) {
                clearChoiceBackground(posChoiceOld);
                posChoiceOld = posChoice;
            }
            renderChoiceBackground(posChoice);

            //  在 0,0 处放置当前选择位置的字符串，注意结尾有空格留白，可以思考为什么要加这么多空格
            MovePos(0, 0);
            char str[32] = "";
            sprintf_s(str, "当前选择(%u, %u)", posChoice.X, posChoice.Y);
            PutString(str);
            //  执行相应操作
            switch (operation) {
            case 1:
                //  翻开地块
                digBlock();
                break;
            case 2:
                //  标记为地雷
                flagBlock();
                break;
            }

            //显示雷数、步数、时间
```

```
if (gameStart == 1)
{
	//显示剩余雷数
	MovePos(0, 0);
	sprintf_s(strMine, "剩余雷数%d ", mineSum);
	PutString(strMine);

	//显示已走步数
	MovePos(0, 0);
	sprintf_s(strStep, "步数%d ", stepCount);
	PutString(strStep);

	//显示游戏时间
	timeEnd = GetTickCount64();
	int mintue = (timeEnd - timeStart) / 60000;
	int second = ((timeEnd - timeStart) / 1000) % 60;
	int smallsecond = ((timeEnd - timeStart) / 10) % 100;
	MovePos(0, 0);
	//根据分钟、秒、百分之一秒是否大于 10，确定是否要显示零
	if (mintue < 10 && second < 10 && smallsecond < 10)
		sprintf_s(strTime, "时间:0%d:0%d:0%d", mintue, second, smallsecond);
	else if (mintue < 10 && second < 10)
		sprintf_s(strTime, "时间:0%d:0%d:%d", mintue, second, smallsecond);
	else if (mintue < 10 && smallsecond < 10)
		sprintf_s(strTime, "时间:0%d:%d:0%d", mintue, second, smallsecond);
	else if (mintue < 10)
		sprintf_s(strTime, "时间:0%d:%d:%d", mintue, second, smallsecond);
	else if (second < 10 && smallsecond < 10)
		sprintf_s(strTime, "时间:%d:0%d:0%d", mintue, second, smallsecond);
	else if (second < 10)
		sprintf_s(strTime, "时间:%d:0%d:%d", mintue, second, smallsecond);
	else if (smallsecond < 10)
		sprintf_s(strTime, "时间:%d:%d:0%d", mintue, second, smallsecond);
	else
		sprintf_s(strTime, "时间:%d:%d:%d", mintue, second, smallsecond);
	PutString(strTime);
}

//游戏是否失败
if (gameFail == 1)
{
	MovePos(14, 2); // 移动坐标到 14,2
	PutString("很遗憾，失败了"); // 在这个坐标放置一个 string
	MovePos(14, 4); // 移动坐标到 14,4
	PutString("鼠标左击返回主菜单"); // 在这个坐标放置一个 string
	Update();// 将地图更新到屏幕
	bool runFlag = true;//判断有没有点击鼠标
	while (runFlag) {
		checkMouse();   // 检查输入
		if (clickMouse == 1)
			runFlag = false;
	}
```

```
            return;
        }

        //游戏是否胜利结束
        gameEnd = 0;
        if (operation != 0)
        {
            for (int i = 0; i < 10; i++)
                for (int j = 0; j < 10; j++)
                {
                    if (mapCanvas[i * 10 + j] == 0)
                        gameEnd = 1;
                }
            if (gameEnd == 0)//游戏结束了
            {
                endGame();
                return;
            }

            else//游戏没结束
            {
                gameEnd = 0;
            }
        }

        // 以下内容不建议修改 处理每帧的事务
        //updateMap(); // 更新地图画板
        Update();       // 更新操作到
        frame++;    // 渲染帧数自增
        clock_t elapsed = 25 - (clock() - tic); // 检查上一帧渲染时间，并计算与 25 的差值
        Sleep(elapsed > 0 ? elapsed : 0);           // 若差值大于零，则休眠该差值的毫秒数，以
确保每帧渲染不超过 50 帧
        tic = clock();                                  // 更新上一次记录的时间
    }
}
```