泡泡堂 - 第一阶段报告

211240088 徐沐杰

完成进度

实时游戏 (Realtime game)

- 实时游戏的架构是一个`while`循环，并只在距离上次经过工作周期超过一个`GAMETICK`（游戏刻）所代表的时间段后进入工作周期，本次项目采用25ms的`GAMETICK`。
- 每个工作周期中，程序主要有三个任务：
    1. 检查游戏状态，包括游戏是否结束，以及玩家是否被光束轰击等等，此后执行对应方法。
    2. 减少计时器的计数，并在计时器计数达到0时执行计时事件。
    3. 处理特殊键盘输入，在此之后逐个调用在场玩家的处理输入方法（对于AI，调用AI方法）。
- 工作周期结束后，程序检查重绘标志是否被置为`True`并依此执行重绘。
- 值得一提的是，本项目采用了基于ANSI控制字符和全缓冲的彩色高速重绘，可以保证不闪屏，高刷新率，多彩显示和较低的跨平台成本。

主要代码结构 (伪`C++`)

```
#define GAMETICK 25
int mainGame::main(){
    while (true):
        if (clock() - lastTime < GAMETICK) continue;
        //update time infomation
        ...();
        //do what should be done every tick
        dealwithRealTime();
        if (_kbhit()):
            //deal with some special inputs
            ...();
            for (i in Players):
                i.keyCatch.deal(&i);
        //repaint if needed
        if (refreshFlag) refresh();
}
```

地图 (map data)

- 地图实际上是一个`unsigned int`类型的数组。
- 该`unsigned int`的低16位用于代表该位置的类型、是否存有道具以及道具的类型（属于某类型则该位为1，否则为0），而高16位用于描述道具属性值、炸弹等级以及其持续时间。
- 为方便解码编码，该变量托管于一个`node`类中，`node`类含有将该变量解码为更访问友好的`nodeInfo`数据结构，该结构用若干`char`和`int`描述这个地图元素，便于判断该元素的类型以及获取其属性值而不必使用各种位运算进行解码。
- 炸弹和光束拥有剩余时间属性，该属性每过一游戏刻减一，归零时光束直接消失，炸弹爆炸，并调用`triggerBomb()`函数以释放光束，炸弹的等级属性等于放下它的人物的等级属性。道具的使用见玩家类的介绍。

```
struct nodeInfo;
struct node{
    unsigned int type;
    nodeInfo getInfo();
}
class mapData{
    node mapbuf[MAXSIZE];
    int triggerBomb(const cursor &cur);
    //other methods...
}
```

玩家 (player) 、机器人(AI)

- 玩家拥有如下基本属性：坐标`pos`，速度`spd`，生命值`hp`，积分`sco`，名字和代表字符。
- `movCnt`是一个计时器，它在每个游戏刻后减一，它为0时才可以进行移动，移动后`movCnt`被重置为`spd`的值（因此，`spd`的意思是每隔`spd`个游戏刻玩家才能移动一次）。
- `keyCatcher`是一个为玩家处理键盘输入的类，并且**保存有该玩家每个动作所对应的键值**，实时程序调用它的`deal()`方法时，如果该玩家不是机器人，它将检查`movCnt`的状态以及每个动作对应键的状态，并调用玩家的移动方法`move()`（实际实现分为上下左右四个）。
- `isAI`标记了该玩家是否是机器人，如果该玩家是机器人，那么`keyCatcher::deal()`方法将调用`robot`对象指定的AI函数执行玩家的移动或放置炸弹操作，本项目的机器人暂时只在地图的一个角落进行固定回环逡巡的策略。
- 玩家在移动时碰到道具触发`eatItem()`函数的调用，获取道具的加成，并清除道具。
- 另外，玩家类型中拥有地图指针`mapData *myMap`用以在方法中获取或操作其所属的地图，这保证了一些更改地图（如放置炸弹）或需要获取地图信息（如移动）的方法的实现。

```
class player{
    //basic attributes...
    int movCnt;
    mapData *myMap;
    keyCatcher keyCatch;
    bool isAI;
    AI robot;
    public:
        int move(); //actually le(), ri(), dw() and up()
        int setBomb();
        int eatItem();
}
```

心得和致谢

- 其实还实现了一些奇奇怪怪的功能，比如开头的命令行启动游戏或使用高级功能，其中高级功能包括游戏、地图的存读档，和实时地图编辑器（内含debug功能，简直和写主程序花的时间差不多了……）（我的地图是用编写的地图编辑器编辑好保存到二进制文件再读的，这样能方便创建新地图和在多个地图间切换）等等。
- 感谢李晗助教提供的ANSI和全缓冲重绘解决方案。它真的比`system("cls")`加复位光标的性能高很多，否则实际游戏刻会很不稳定（因为重绘时长接近游戏刻时长）。另外，控制字符设置颜色要比调用API设置颜色好用很多，因为和语句相比，字符可以直接写进字符串里灵活读取输出。
- `_getch()`有很多缺点，比如同时长摁的时候丢键，而且转弯的时候之前输入堆积，手感很硬等等，因此我换用了`GetKeyState()`的API检测按键状态，改善了这一问题。