课程设计三实验报告

171860669 刘如岩
联系方式：18852005502

目录

一、主要内容

实现 GUI 植物大战僵尸小游戏。有两项规定：

a） 必须是 C++ GUI 程序——我使用的是 Qt 的 GUI 框架

b） 必须是 OOP 风格的 —— main 之外的全局函数尽量少，并合理运用继承，多态，泛型等特性。

二、实现的目标

1. 庭院布局采用 5 行 9 列，背景是一个草地的 png 图片。
2. 每隔一定时间，系统会产生阳光，种植的太阳花也可以产生阳光，玩家需要点击阳光来收集它们。
3. 实现 3 种僵尸：普通僵尸、路障僵尸和铁桶僵尸，同时实现 3 种僵尸属性：生命值、攻击力、速度。
4. 实现 6 种植物：向日葵、豌豆射手、寒冰射手、双发射手、坚果墙和樱桃炸弹，同时实现 3 种植物属性：购买所费阳光数、生命值、攻击力。
5. 地图左侧有一个选择栏，玩家点击需要的植物，然后在草地上面点击鼠标左键种植；点击右键会取消。
6. 地图左上角有一个阳光栏，记录阳光数；还有一个铲子图标，点击后可以删除植物。
7. 通过一定的策略产生僵尸：以 75s 为一个周期，一个周期内，僵尸的数量和等级随着时间递增。

三、主要类的设计

1. 类的数据和操作

1)Plant 植物类

```
class Plant:public QLabel
{
    Q_OBJECT
public:
    Plant(QWidget* parent=nullptr);
    ~Plant();
    void initInfoPlt(int r,int c,const QString& qstr);
    virtual void act();
    virtual void inHp(int x=-1);
    char isAlive();
    void clr();
    QTimer tmr;
    static Game *game;

protected:
    int row,col;
    int hp,hitPower;
    float coolTime;
    QMovie* qmv;

};
```

虚函数 void act()处理各个植物的行为，为不同的植物有不同的行为，需要动态绑定；虚函数 inHp()负责减血，因为坚果墙在不同的血量时有不同的图像：



而其他植物无此行为，所以要动态绑定。

2)ObjectMov 可移动实体类

```
class ObjMov:public QLabel
{
    Q_OBJECT
public:
    ObjMov(QWidget* parent=nullptr);
    ~ObjMov();
    void initInfo(int cycle=20,const QString& qstr="",const QString& qstr1="");
    void movePos();
    char isAlive();
    virtual void act();

    static Game* game;
    QTimer tmr;

protected:
    int hp;
    char movType;
    QPixmap pix;
    QMovie* qmv;
};
```

ObjMov 类继承自 QLabel 类，是所有可移动实体的基类（包括子弹、僵尸、阳光等实体），void act()是实体的行为，因为不同的实体有不同的行为，所以将其设置成虚函数；void movePos()负责根据 movType 来移动实体一个像素。

3)ObjMovTmp 临时实体类

```
class ObjMovTmp:public ObjMov{
public:
    ObjMovTmp(QWidget* parent=nullptr);
    void deTime();

protected:
    int yMax,timeCnt;
//    int yMax,id,timeCnt;
};
```

该实体会在移动到 yMax 时执行 deTime()，倒计时结束后自动消亡（适合阳光、铁桶、路障等部件）

4)Sun 阳光类

```
class Sun:public ObjMovTmp{
public:
    Sun(const QRect& rect,QWidget* parent=nullptr);
    void clr();
    void act();

};
```

Sun 继承了 ObjMovTmp 临时实体类，同时添加了自己的行为 act()，和清理函数 clr()。

5)Bullet 子弹类

```
class Bullet:public ObjMov{
public:
    Bullet(QWidget* parent=nullptr);
    ObjMov* searchCrash();
    void clr();
protected:
    int row;
    int hitPower;
};
```

Bullet 类继承了 ObjMov 类，可以移动，还添加了 ObjMov* searchCrash()函数，负责检查碰撞，然后返回撞到的物体的指针。

6)Zombie 僵尸类

```
class Zombie:public ObjMov
{
    Q_OBJECT
public:
    Zombie(int r,QWidget* parent=nullptr);
    ~Zombie();
    void setCold(char cold=1);
    void setCommon();
    void setFired();
    void putHead();
    void clean(int frameId);
    void setAttack(char sure);
    void hitPlt();
    void inHp(int x);

    char isEnd();
    void act();

    QTimer tmrHit;
    QMovie* attack;

signals:
    void gameover();
protected:
    int row,rowTar,colTar;
    int hitPower;
    char cold,end,hiting;
    QString dir;
    QMovie* die;
    QLabel* headLab;
    QMovie* headMv;
    QLabel* iceLab;

//    QMovie* attack;
};
```

void setCold(char cold=1)负责让僵尸冰冻，void setFired()负责让僵尸烧成灰，void putHead()负责让僵尸的头和身体分离，void clean(int frameId)负责在 gif 的最后一帧时清理该僵尸对象，void setAttack(char sure)负责设置僵尸是移动状态还是攻击状态，void hitPlt()负责袭击植物，void act()是继承自 ObjMov 的虚函数，负责处理僵尸的行为。

7)GameHelper 辅助类

```
class GameHelper:public QObject
{
    Q_OBJECT
    Game* game;
public:
    GameHelper(Game* g);
    Plant* whichPlant(int row,int col);
    void putZmbs();

    void setCyclePut(int tmp);

    QPoint posAlignBottom(const QRect &destRect,const QSize &srcSize);
    QPoint posAlignCenter(const QRect &destRect,const QSize &srcSize);
    QPoint posAlignTop(const QRect &destRect,const QSize &srcSize);
    bool isPosInRect(const QPoint& pos,const QRect& rect);

    QPoint posWhiteCell(bool add);
    int findPtrId(QList<ObjMov*> &list,void *ptr);
    char checkCrash(const QLabel& label0,const QLabel& label1,char strict=1);
    std::pair<int,int> rowAndCol(bool center=1,int x=-1,int y=-1);
private:
    QTimer tmrPut;
    int zmbCnt;
    int cyclePut;
};
```

该类负责选择植物，放置僵尸，设置上对齐、下对齐和居中对齐，bool isPosInRect()查看一个点是否在一个矩形内，char checkCrash()检测碰撞，std::pair<int,int> rowAndCol()负责查看一个点在草地的行数和列数。

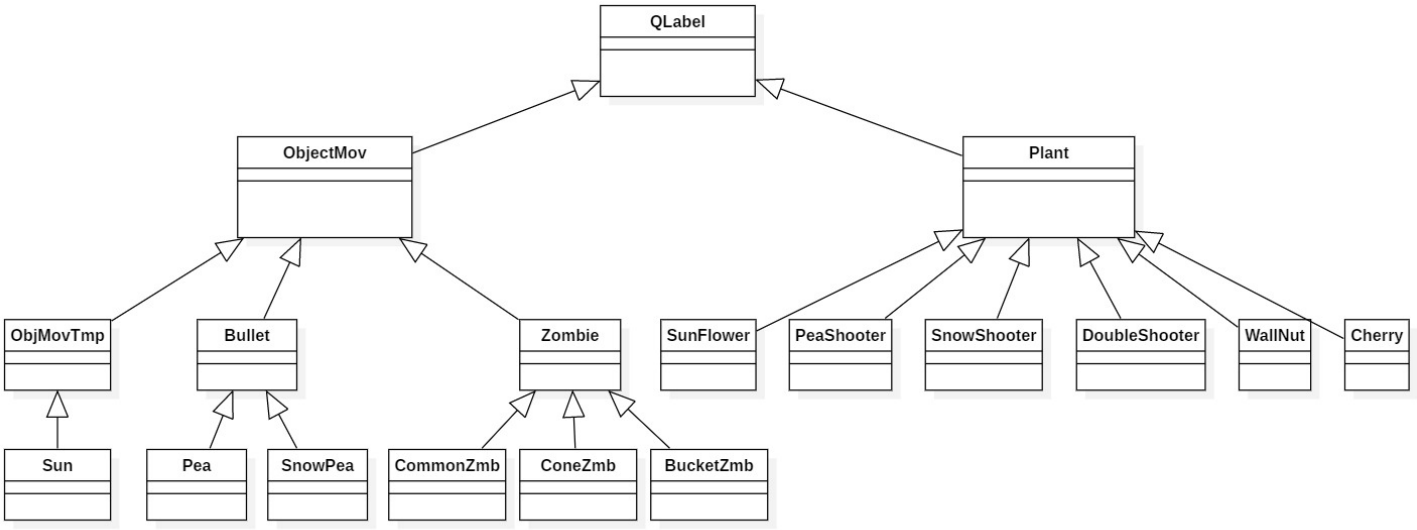
8)Game 游戏类

```
class Game:public QObject
{
    Q_OBJECT
public:
    Game(QWidget*parent=nullptr);
    ~Game();
    void initInfo();
    char findSun();
    char clicked(char sure);
    void bulletsAct();
    void setStart(bool sure);
    void end();

    void clrList(QList<ObjMov*> &objList,int id=-1);
    void clrObjects();

    char plantHandle(bool add);
//    void zombieHandle(bool add);
//    void bulletHandle(bool add);

    void putSysSun();
    void updateCursorPix();

    friend GameHelper;

    GameHelper gameHelper;
    Plant* plants[ROW_NUM][COL_NUM];
    QList<ObjMov*> zombies[ROW_NUM];
    QList<ObjMov*> bullets[ROW_NUM];
    QList<ObjMov*> suns;//heads;

    QWidget* mainWin;
    QPoint cursorPos;
    int curPltId;
    int sunNum;
    char state;

private:
    QPixmap cursorPixs[NUM_PLANT];

    QTimer tmrSys,tmrZmb,tmrBlt;
    QPoint shovelBegin;
    QLabel *cursorPlt;
    QLabel *shovel;
    QLabel *shovelBack;
    QLabel *sunBack;
    QLabel *sunNumTxt;
    QLabel *whiteCell;
    QLabel *endLab;
    bool showCursorImg;

};
```

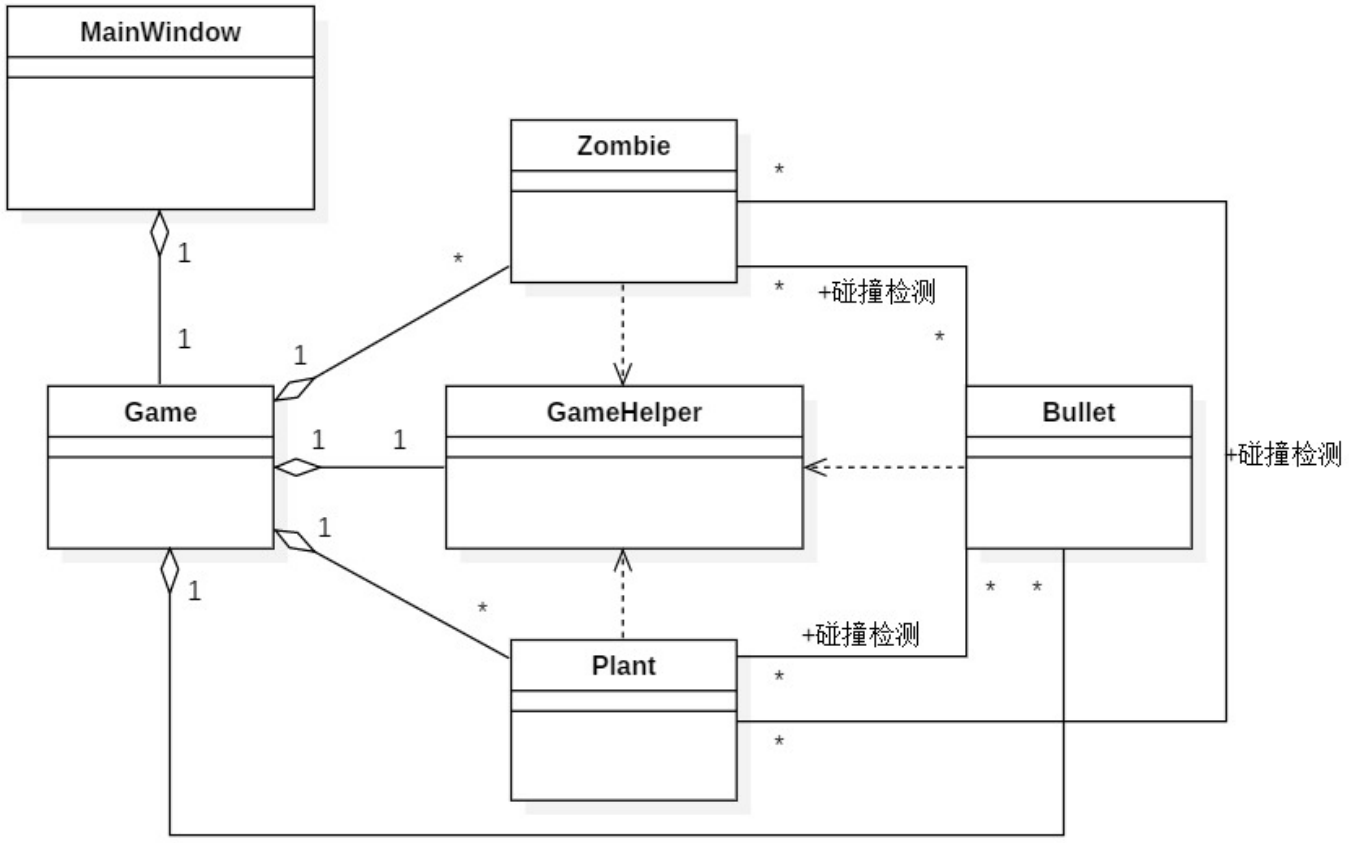
char findSun()负责查找地图中太阳，char clicked()负责处理玩家的鼠标点击事件（包括左键确定和右键取消），void bulletsAct()负责处理子弹的移动（因为子弹移动的周期相同，所以用统一的函数），char plantHandle(bool add)负责处理种植植物（add==1 时），或铲除植物（add==0 时），void updateCursorPix()负责更新与鼠标同步的图片的位置（比如种植状态会产生一个相应的植物图片跟随光标，铲除状态会产生一个铲子图片跟随光标），

2. 类之间的关系

首先讲的是游戏中的元素的继承关系。因为各个基类中都含有一个虚函数 void act()，负责处理各个对象的行为，一个对象只需要重写 void act()，就可以通过基类指针访问子类对象自己的 act()函数。



最后讲的是各个类的聚合、关联和依赖关系。

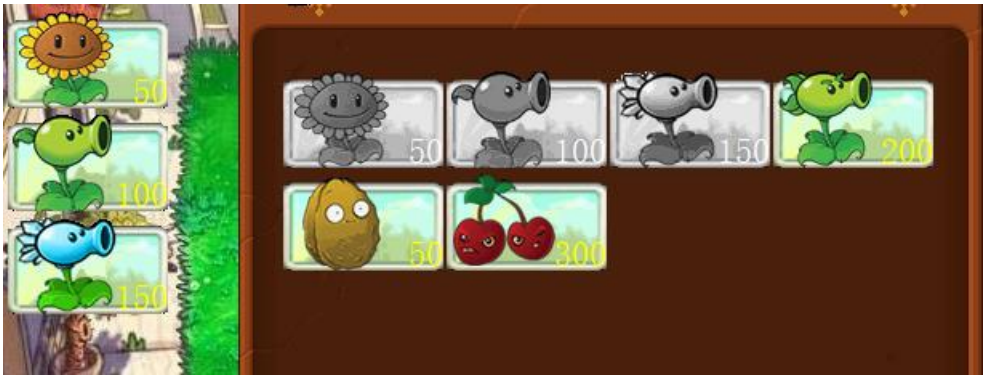


四、程序的功能特点和操作方法

1. 功能特点

① 有一个选择栏，共有 6 个待选择的植物，都是鼠标点击确定。

② 选择状态下，玩家点击后，右侧被选中的卡片变为灰色，左侧添加被选中的卡片



点击左侧卡片，取消选择，左侧栏卡片减一，右侧栏相应的卡片变为亮色。

③ 运行状态下，玩家选中一个植物，会产生一个相应植物的图片跟随光标移动，而且被选中的格子会有白色区间提示。

④ 种植成功后，相应的植物卡片会进入缓冲状态，等待 CD，阳关不足的植物，卡片会变灰黑色。

⑤ 铲除状态下，会产生一个跟随光标移动的铲子。

⑥ 寒冰射手的子弹会使僵尸减速，同时产生一个冰层障碍。

⑦ 僵尸死亡有两个状态，头与实体分离和被烧成灰（樱桃的爆炸效果）

⑧ 坚果会根据生命值变成不同的状态

⑨ 游戏结束，显示“僵尸吃掉了你的脑子”。

2. 操作方法

   与普通的植物大战僵尸游戏相同。

五、遇到的问题和解决方案

1. 经常访问到空指针，或数组越界。

   访问前先检测一下是否为空，或下标是否超过最大限度。

2. 在删除 QList 的元素时，经常溢出。

   反向遍历，从 size-1 开始遍历到 0，遇到 hp 为 0 的对象则将其删除