课程设计报告二

姓名：马常风

院系：计算机科学与技术系

学号：171860637

邮箱：njumcf@126.com，171860637@smail.nju.edu.cn

一、项目简介

1. 题目选择

- 植物大战僵尸控制台版

2. 游戏简介

- 本游戏为控制台版的植物大战僵尸，无尽模式，玩家需要通过种植不同植物来抵挡僵尸，分数由抵挡时间和击杀僵尸数决定。

3. 其他

- 文件树

```
PVZ
├── build                           // build文件夹
│   ├── CMakeCache.txt
│   ├── CMakeFiles
│   │   └── ...
│   ├── cmake_install.cmake
│   ├── Game                        // 可执行文件
│   └── Makefile                    // Makefile
├── CMakeLists.txt                  // CMakeLists
├── common.h                        // 通用定义头文件
├── KeyBoard.cpp                    // 键盘读取模块
├── KeyBoard.h
├── MainControl.cpp                 // 主控类
├── MainControl.h
├── main.cpp                        // 程序入口
├── Object.cpp                      // 物品类
├── Object.h
├── Scene.cpp                       // 场景类
├── Scene.h
├── ScreenDraw.cpp                  // 屏幕绘制类
└── ScreenDraw.h
```

- 项目在Ubuntu16.04、CMAKE3.5.1、g++5.4.0环境下开发。`包含CMakeList.txt`。但由于使用了Linux系统特有的函数和库，因此不支持在Windows或其他系统下编译。编译成功后，生成可执行文件`Game`。
- 编译：进入`PVZ`文件夹

```
mkdir build
cd build
cmake ..
make
```

- 运行：为了游戏正常运行，需要全屏控制台。

```
./Game
```

二、游戏详情

1. 游戏名称

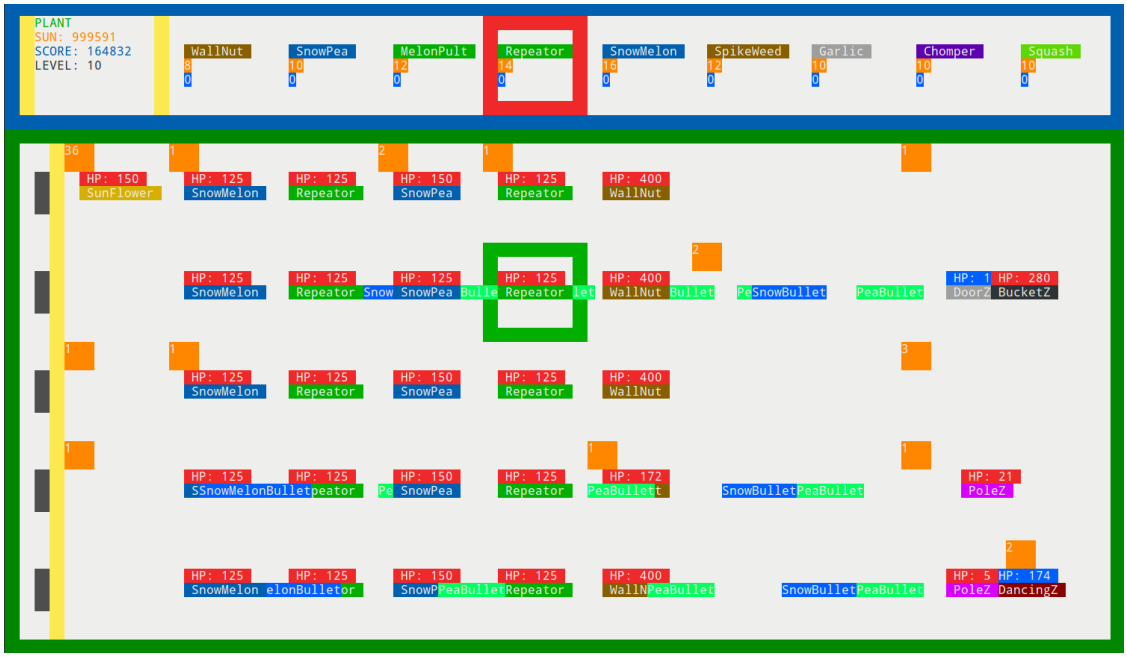
- 植物大战僵尸控制台版

2. 玩法介绍

- 植物和僵尸的特性基本同原植物大战僵尸。操作上，游戏分为三种状态`NORMAL`、`PLANT`、`REMOVE`，通过`f`切换。`NORMAL`状态用于收集太阳，`PALNT`状态用于种植植物，`REMOVE`状态用于移除植物。`w`、`a`、`s`、`d`移动选择框进行选择种植或收集的区域。`Enter`种植、收集或移除。通过`q`、`w`选择种植的植物。`p`暂停，再按继续。暂停状态下，`ESC`退出。

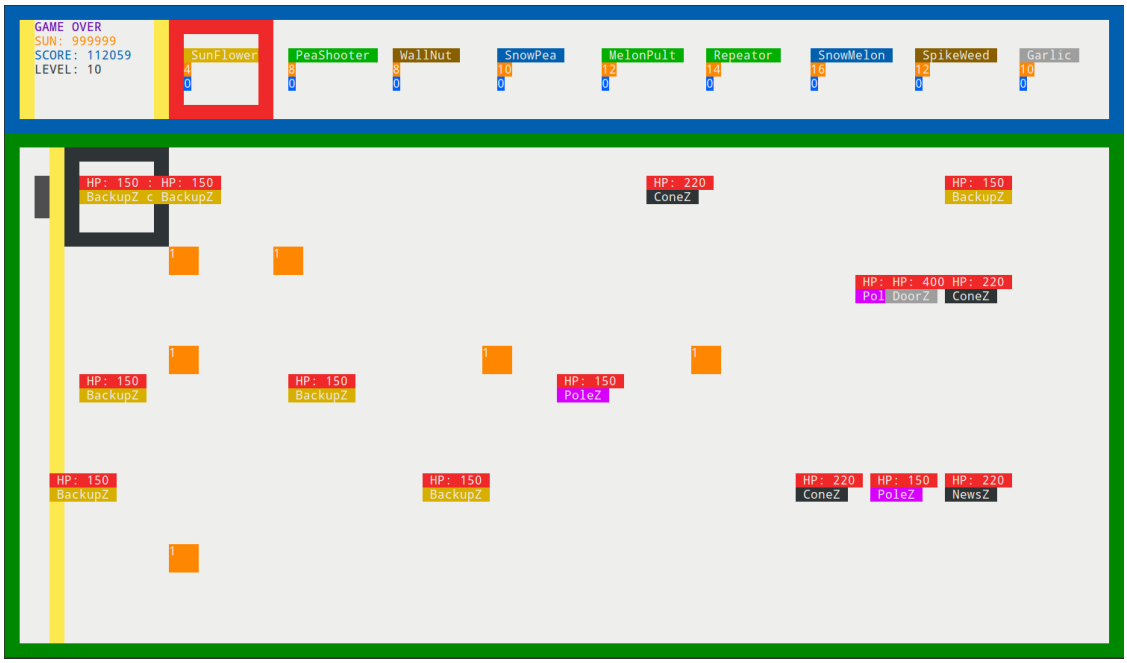
3. 游戏截图

- 种植模式



- 暂停

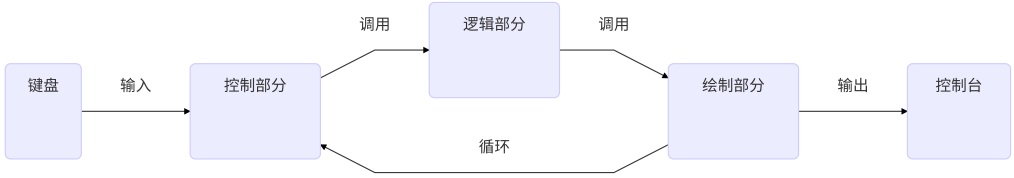
- 结束



三、游戏设计

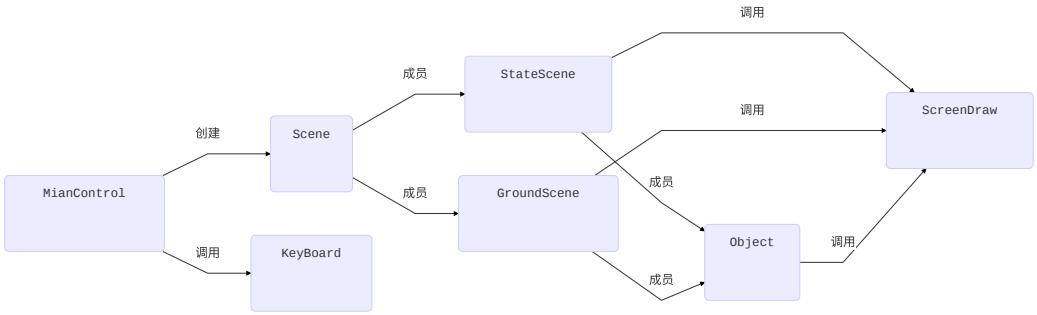
1. 设计思路

- 将控制、逻辑、绘制分离

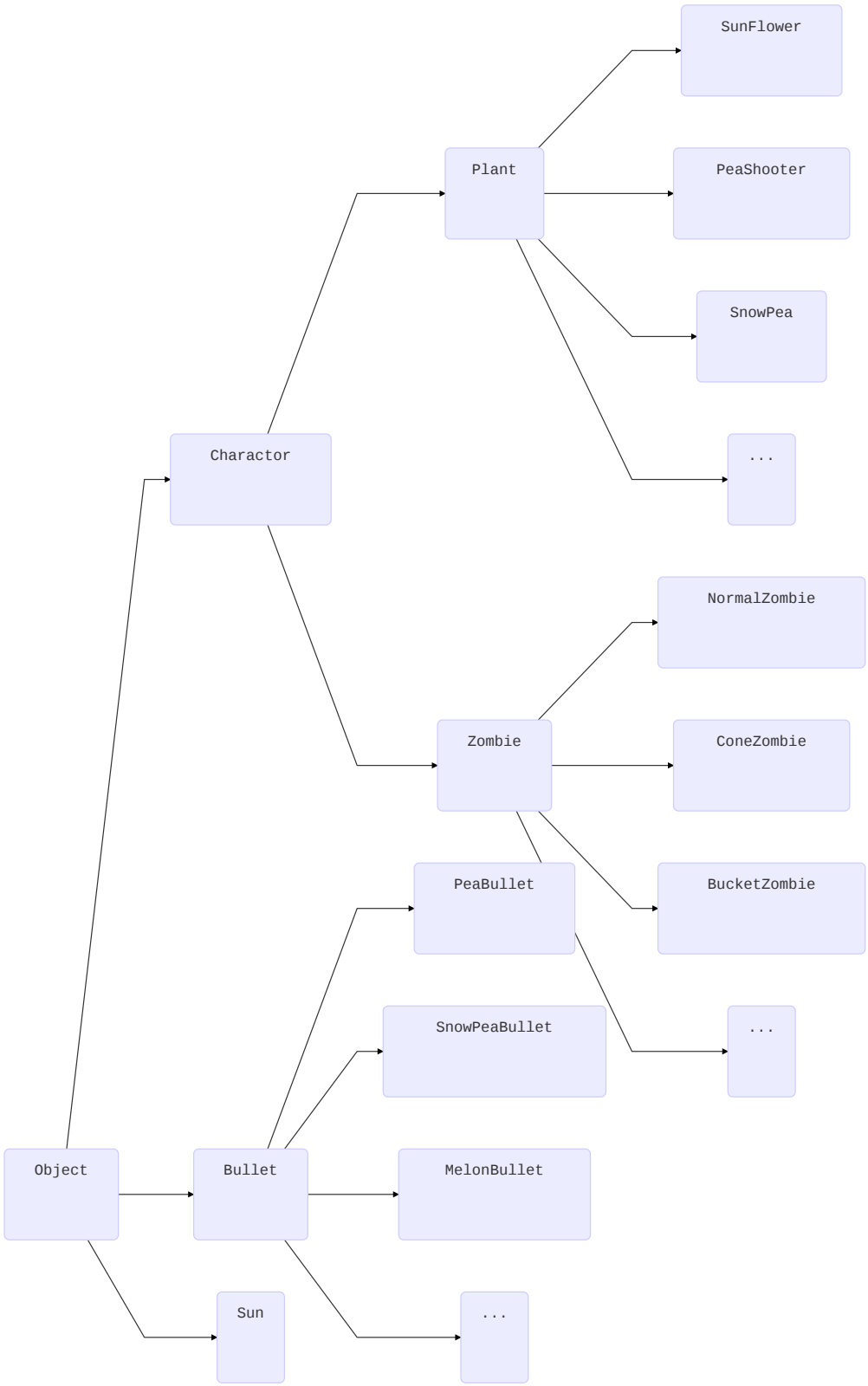


- 主控类负责接收键盘信息并操控场景类，场景类包含物品类负责逻辑，然后调用绘制模块进行绘制。

- 类关系



- `Object`类关系



- 设计细节
  - 主控`MainControl`：负责创建`Scene`并根据键盘信息来调用相关的接口。每次循环接收一次键盘信息，操作一次`Scene`，`Scene`更新`update`一次，绘制`draw`一次。
  - 场景`Scene`：场景包含状态场景`StateScene`和地面场景`GroundScene`。每次`update`会调用两个场景的更新`update`接口，接收`update`返回的信号，然后处理信号。每次`draw`会调用两个场景的`draw`接口。当场景接收到`Enter`的调用后，会调用两个场景的信号发送接口，然后处理对方的信号，执行相应的收集、种植、移除操作。**也就是说，两个场景类通过**

发送和处理信号来进行交互。信号的定义在`Scene.h`中。`Scene`还负责等级的提升和状态的统一设置。

- 状态场景`StateScene`：负责记录分数，太阳数，当前状态；负责展示全部植物，选择植物，进行购买时的判断，植物冷却等。当需要发送信号时，状态场景根据当前的状态发送信号，比如，当状态为`PLANT`时，状态场景根据当前选定的植物是否冷却完毕，是否有充足的太阳来购买来发送信号。处理信号同理，比如，一旦植物可以种植，状态场景会立刻扣除相应的太阳，并将对应的植物置入冷却倒计时。
- 地面场景`GroundScene`：负责保存全部的物体，全部物体之间的交互，更新全部的物体，绘制全部的物体等。`update`时先根据物体的类型，进行交互，比如僵尸`Zombie`和植物`Plant`之间的交互，子弹`Bullet`和僵尸`Zombie`之间的交互等。然后进行物体的更新`update`。全部的物体使用一个`vector`数组储存，然后每次更新时将死亡的物体删除。然后每个物体会发送一个信号，场景再处理信号。根据不同的信号，地面场景会将信号再传递给`StateScene`。

  `update`流程：

  1. 不同物体之间的交互
  2. 物体更新
  3. 物体发送信号
  4. 处理物体信号

  物体信号定义在`Object.h`。

  同时`GroundScene`负责随机生成僵尸。生成僵尸的时间、数量和生成间隔根据等级决定，等级越高，数量越多，生成间隔越短。

  地面场景有相同的场景信号发送类和处理类，在此不多赘述。

- 物体`Object`：程序中所有物体的基类，包含坐标，类型等成员变量，更新，绘制，发送物体信号等虚函数。因仅有特定的类才能交互，故没有将交互函数加入基类。
- 角色`Charactor`：继承自`Object`，是`Plant`和`Zombie`的基类。新增了攻击和防御的部分并增加了生命值，根据攻击速度和当前的状态（攻击计时器）来决定是否攻击。当角色受到攻击时调用防御`defense`，会损失生命值，计算方法如下：

$$\text{受到的攻击值}\left(1-\frac{\text{自身的防御}}{\text{自身的防御}+\text{预设防御}}\right)$$

  其中预设防御为50。

- 僵尸`Zombie`：继承自`Charactor`，是全部僵尸类的基类。新增移动速度和与植物交互的函数。重写`update`方法进行移动。交互时，传入一个植物对象的参数，在满足一定条件时，攻击植物，即两者坐标满足一定条件并且僵尸处在攻击状态，调用植物的`defense`。僵尸在攻击开始时，停止移动。
- 植物`Plant`：继承自`Charactor`，是全部植物类的基类。新增与僵尸交互的函数。但不同的是，发射子弹的植物在攻击时，通过发送物体信号使`GroundScene`将特定种类的子弹加入`Object`数组。
- 子弹`Bullet`：继承自`Object`，是全部子弹的基类。增加了威力和移动速度，包含一个与僵尸交互的函数。通过坐标判定来确定是否攻击僵尸。
- 其他僵尸、植物、子弹子类：继承自各个父类，通过重写更新，交互，发送信号来实现不同的特点。详细实现的类别见下表：

植物	僵尸	子弹
太阳花	普通僵尸	豌豆子弹
豌豆射手	路障僵尸	寒冰豌豆子弹
寒冰射手	铁桶僵尸	西瓜子弹
西瓜投手	铁门僵尸	寒冰西瓜子弹
冰瓜	读报僵尸	
地刺	撑竿跳僵尸	
大蒜	舞王僵尸	
食人花	舞伴僵尸	
倭瓜		
土豆地雷		
火爆辣椒		
坚果墙		
双发射手		

- tricky实现
  - 可移动的商店选择。商店显示仅有9个位置，但有13种植物。当移动到最右侧时，将整个框内展示的植物左移，此时可以显示全部植物。向左移动同理。
  - 统一管理全部物体。使用`Object`数组统一调用接口，实现全部物体的更新和绘制。无需特殊处理。
  - 使用信号传递和处理来进行类与类的交互。分为场景信号和物体信号。场景信号负责`StateScene`和`GroundScene`的交互，物体信号负责物体与`GroundScene`的交互。信号类型预先定义，信号同时还包含一定的数据。在指定的类型中对信号处理，从而达到信号处理的目的。可以实现，从低级类别到高级类型的通信和同级类型的通信。
  - 物体类型的确定。在`Object`数组进行遍历，做两两交互的时候如何确定所属小类型是否为植物、僵尸还是子弹，现做如下定义。

```
#define OBJ_TYPE_NULL             0
#define OBJ_TYPE_SUN              1
#define OBJ_TYPE_CHARACTOR        2
#define OBJ_TYPE_PLANT            3
#define OBJ_TYPE_ZOMBIE           4
#define OBJ_TYPE_BULLET           5

#define OBJ_TYPE_SUNFLOWER        13
#define OBJ_TYPE_PEASHOOTER       23
...
#define OBJ_TYPE_REPEATOR         133

#define OBJ_TYPE_NORMALZOMBIE     14
#define OBJ_TYPE_CONEZOMBIE       24
...
#define OBJ_TYPE_BACKUPZOMBIE     84
```

```
#define OBJ_ZOMBIE_NUMBER         8

#define OBJ_TYPE_PEABULLET        15
#define OBJ_TYPE_SNOWBULLET       25
...
#define OBJ_TYPE_SNOWMELONBULLET  45
```

可见，仅需对类别做模10处理，判断是否与大类别相等即可。

- 显示遮挡处理。当有若干物体因坐标靠近导致绘制遮挡时，随机调整绘制顺序即可。但防止闪烁的出现，现每秒随机调整一次顺序。
- 实现了西瓜的溅射伤害。因为`Object`数组遍历没有顺序性，无法在子弹爆炸时确定每一个受到伤害的僵尸，此时，需要用数组记录全部在子弹附近的僵尸，当本次交互结束并且子弹爆炸，才对数组中的僵尸造成伤害。

四、问题及解决方案

- 颜色过少的问题
  - 采用256色输出。现在的绘制模块全部采用256色输出。
- 代码被误删并且无法找回
  - 这个没有办法解决，除非重写。没办法，我就是重写了一边，还是方法的实现模块。这件事告诉我们，写项目一定要用`git`记录，否则代码丢了都找不回来。

五、项目未来改进

- 可增加宏定义插入指定新对象的功能。以此来实现快速插入新的植物或僵尸的功能。
- 修改宏定义类别的定义方式，达到可以任意修改大类型的目的。