

植物大战僵尸

组员：董欣然，高泽斌，刘珈征，刘雨薇，熊伟民，杨雨诺

目录

- 项目简介
- 项目功能
- Demo展示
- 项目分工
- 项目结构
- 特色 & 难点
- 总结 & 教训
- 个人开发体会

项目简介

- 项目选题

植物大战僵尸游戏

- 语言框架：纯JAVA

- 代码量：159K, 4730行

- 第三方库

- org\json：解析json对象，用于操作间传递信息参数

- org\apache\commons\io : 用于文件读入

项目功能

- 直接以plant_vs_zombie文件夹加载好java项目之后，通过src/core/game/Main.java中的main(src.core.game.Main)函数直接运行。
- 点击冒险模式可以进入选择植物的界面，选择满植物可以点击开始游戏。
- 进入游戏界面后，可以点击阳光进行收集，收集足够的阳光以及植物的缓冲时间结束后可以选择植物，种在虚影所在位置。
- 植物根据其特性攻击僵尸，僵尸掉血后根据其类别分别掉下自己的身体部位，最后死亡。
- 小推车有一次清理该行僵尸的能力，若僵尸进入房子则显示游戏失败。
- 当所有僵尸被清除，会显示战斗胜利，可以进入下一关的选择植物界面。
- 夜晚模式不随机掉落阳光。
- 当所有关卡均通过，会从最简单的关卡继续开始循环。

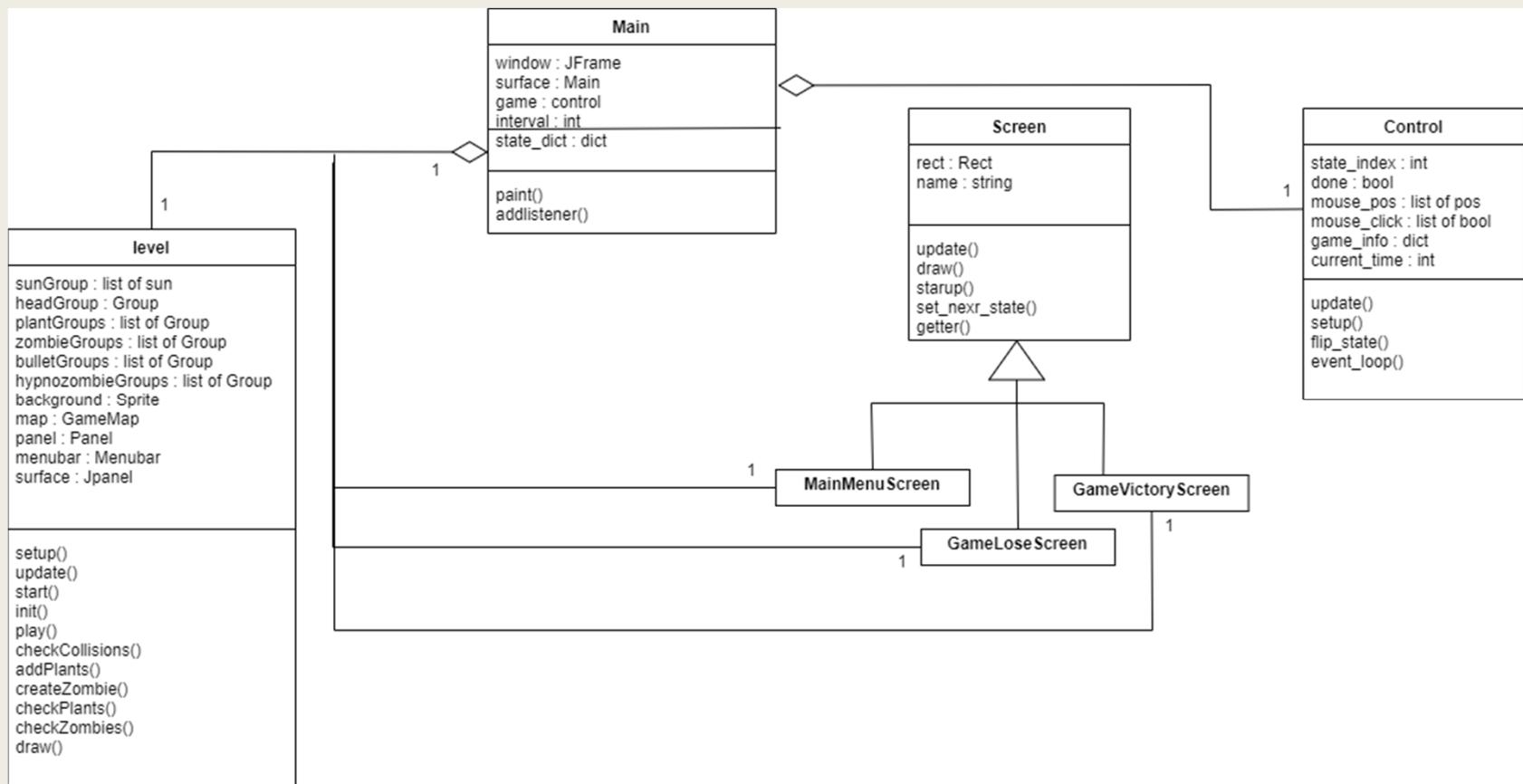
Demo展示



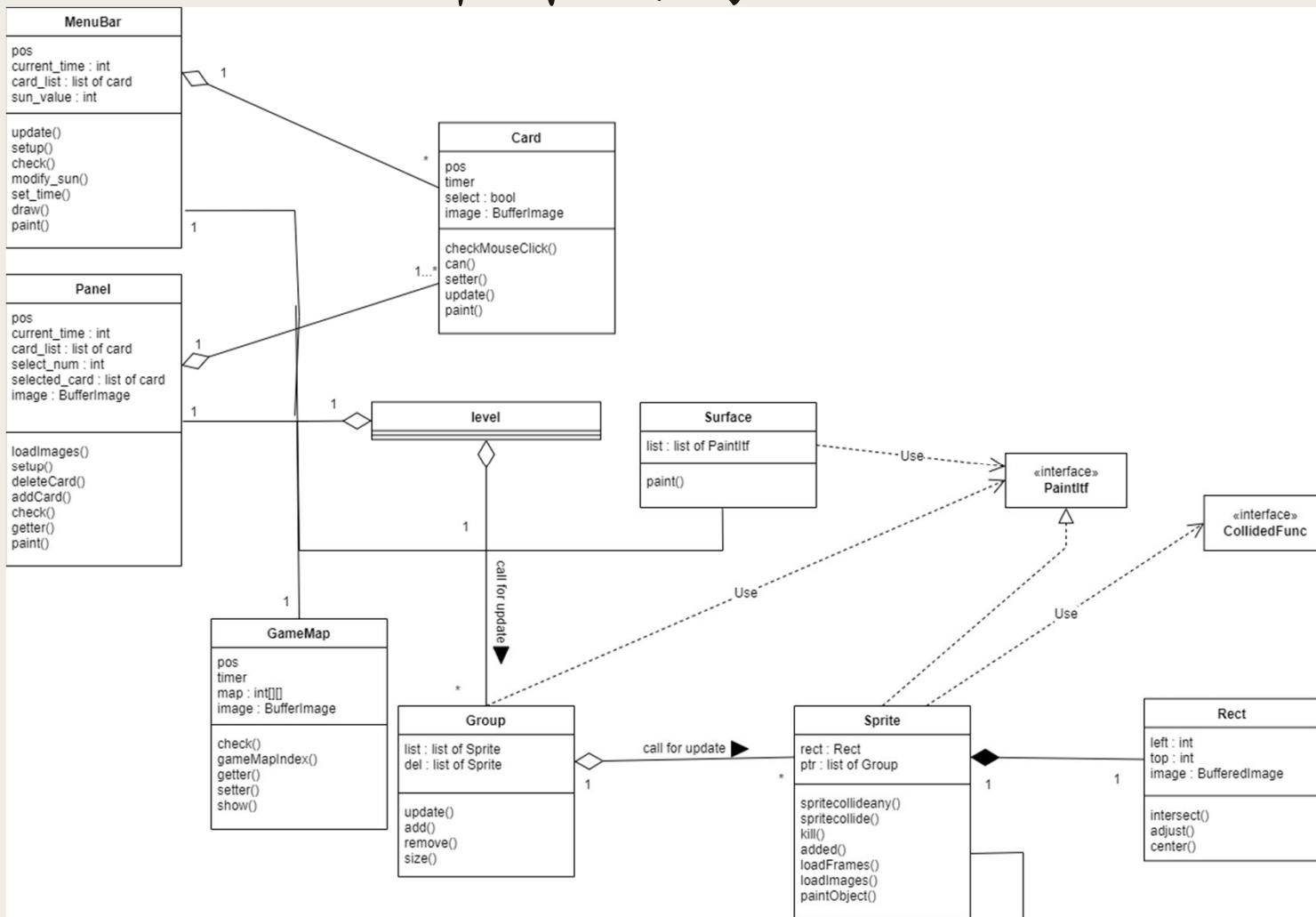
项目分工

- 董欣然：组长，负责僵尸部分的代码编写，图片工具和json文件的调试封装处理。
- 高泽斌：负责游戏主体level部分及游戏进行后UI的代码编写，及前期debug工作。
- 刘珈征：负责射手植物及其子弹、游戏开始界面的代码编写。
- 刘雨薇：负责小推车和阳光的少量代码编写，及所有文档和汇报工作。
- 熊伟民：负责植物选择界面的代码编写和UI调试测试。
- 杨雨诺：负责plant父类及其他植物及其子弹的代码编写。
- 全员：项目框架成型之后的debug和后期调试及集成测试。

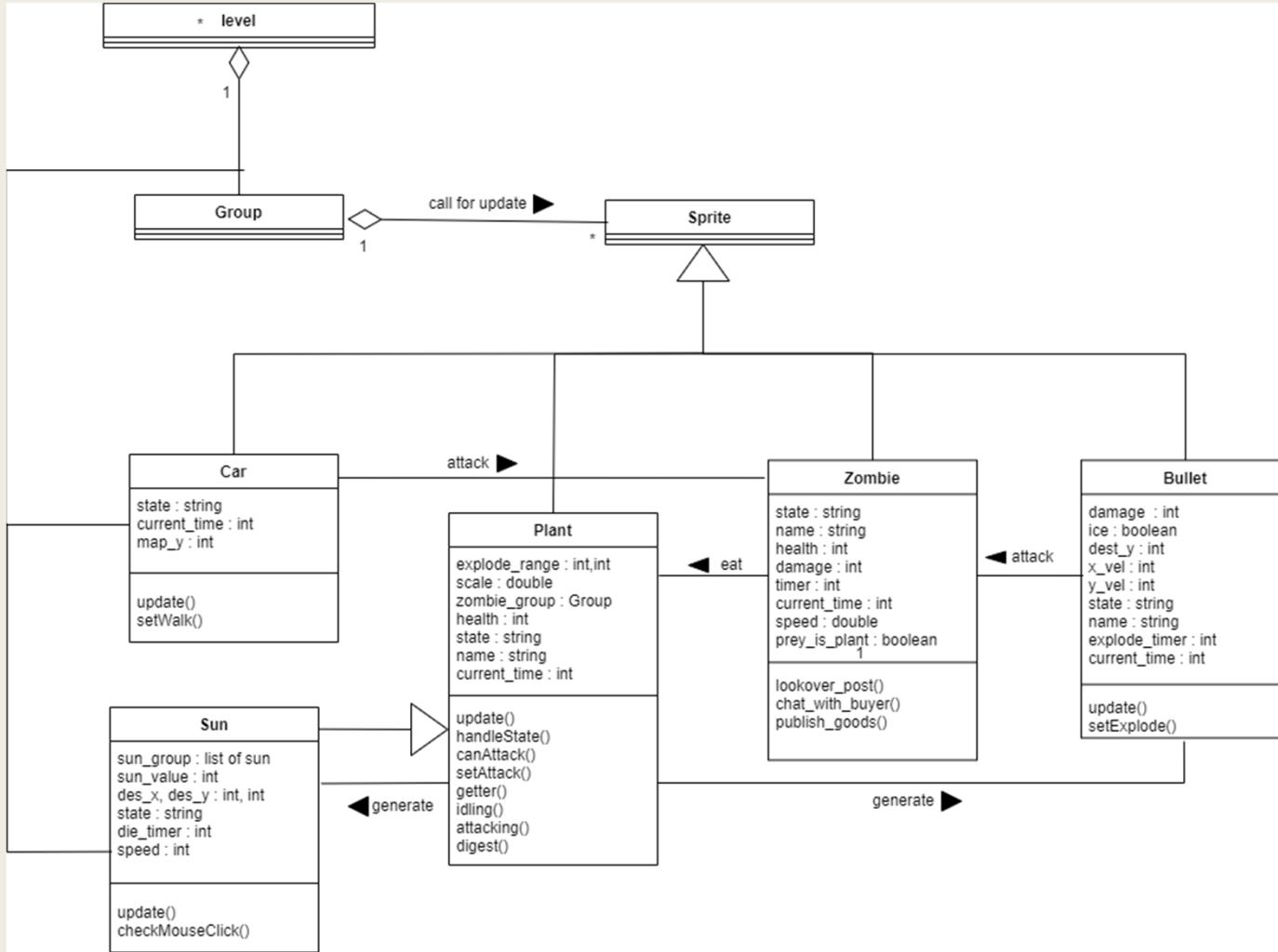
项目顶层1 - 界面框架



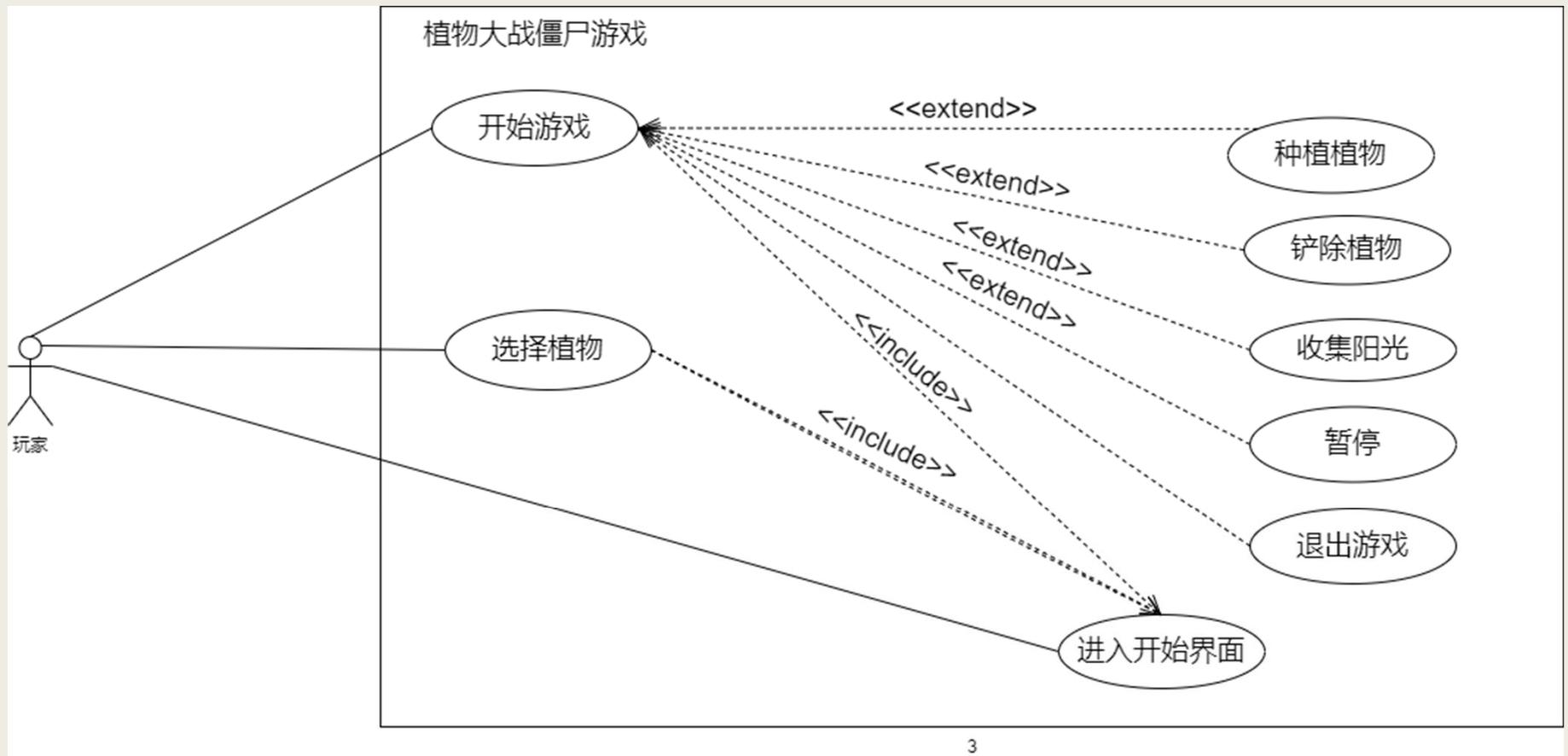
项目顶层2 - 单个对象



项目底层结构



项目设计



特点 & 难点

- 精致还原的动画
- 多类联合调度和更新
- 多类的生成
- 精准的类间判定
- 流畅的用户体验
- 合理的代码组织

精致的动画效果

- 进行良好定义和使用的图片包装，将图片定位封装
- 通过加载一系列图片进行操作
- 按interval切换图片

```
public void loadImages(){  
    walk_frames = new ArrayList<BufferedImage>();  
    attack_frames = new ArrayList<BufferedImage>();  
    losthead_walk_frames = new ArrayList<BufferedImage>();  
    losthead_attack_frames = new ArrayList<BufferedImage>();  
    die_frames = new ArrayList<BufferedImage>();  
    boomdie_frames = new ArrayList<BufferedImage>();  
  
    loadFrames(walk_frames,name,Tool.ZOMBIE_RECT.getJSONObject(name).getInt("x"),Constants.BLACK);  
    loadFrames(attack_frames,name + "Attack",Tool.ZOMBIE_RECT.getJSONObject(name + "Attack").getInt("x"),Constants.BLACK);  
    loadFrames(losthead_walk_frames,name+ "LostHead",Tool.ZOMBIE_RECT.getJSONObject(name+ "LostHead").getInt("x"),Constants.BLACK);  
    loadFrames(losthead_attack_frames,name+ "LostHeadAttack",Tool.ZOMBIE_RECT.getJSONObject(name+ "LostHeadAttack").getInt("x"),Constants.BLACK);  
    loadFrames(die_frames,name+ "Die",Tool.ZOMBIE_RECT.getJSONObject(name+ "Die").getInt("x"),Constants.BLACK);  
    loadFrames(boomdie_frames,Constants.BOOMDIE,Tool.ZOMBIE_RECT.getJSONObject(Constants.BOOMDIE).getInt("x"),Constants.BLACK);  
}
```

多类联合调度和更新

- 活用多态和继承
- 集成的统一调度

```
ArrayList<Sun> sunGroup;
Group headGroup;
ArrayList<Group> plantGroups;
ArrayList<Group> zombieGroups;
ArrayList<Group> hypnoZombieGroups;
ArrayList<Group> bulletGroups;
public void setupGroups() {
    ArrayList<Object> list = new ArrayList<>();
    list.add(game_info);
    for (int i = 0; i < map_y_len; ++i) {
        bulletGroups.get(i).update();
        plantGroups.get(i).update();
        zombieGroups.get(i).update();

        checkBulletCollisions();
        checkZombieCollisions();
        checkPlants();
        checkCarCollisions();
        checkGameState();
    }
}
```

多类的生成

■ 工厂模式

```
public class Level extends State {  
    //temporary;  
    int mouseX, mouseY;  
    int map_y_len;  
    int backgroundType;  
    Sprite background;  
    GameMap map;      gzb1128, 2周前 ·  
    JSONObject mData;  
    double zombieStartTime;  
    ArrayList<ZombieListItem> zombieList;  
    String state;  
    int barType;  
    ArrayList<Card> cardPool;  
    JPanel panel;  
    MenuBar menubar;  
    // elements added into surface  
    JPanel surface;  
    Start start;  
  
    // work in map directory
```

```
    int y = attackIndex.get(index - 1);  
    if (plantName.equals(c.WALNUT)) {  
        newPlant = new WallNut(x, y);  
    }  
    else if (plantName.equals(c.SQUASH)) {  
        newPlant = new Squash(x, y);  
    }  
    else if (plantName.equals(c.SUNFLOWER)) {  
        newPlant = new SunFlower(x, y, sunGroup);  
    }  
    else if (plantName.equals(c.PEASHOOTER)) {  
        newPlant = new PeaShooter(x, y, bulletGroups.get(map_y));  
    }  
    else if (plantName.equals(c.SNOWPEASHOOTER)) {  
        newPlant = new SnowPeaShooter(x, y, bulletGroups.get(map_y));  
    }  
    else if (plantName.equals(c.CHERRYBOMB)) {  
        newPlant = new CherryBomb(x, y);  
    }  
    else if (plantName.equals(c.THREEPEASHOOTER)) {  
        newPlant = new ThreePeaShooter(x, y, bulletGroups, map_y);  
    }  
    else if (plantName.equals(c.REPEATERPEA)) {  
        newPlant = new RepeaterPea(x, y, bulletGroups, map_y);  
    }  
}
```

精准的类间判定

- 活用接口和封装，进行统一的抽象判定

```
lic class Sprite implements PaintItf {           You, 现在 • Uncommitted changes
    //矩形碰撞体积
    public Rect rect;
    /// a pointer to delete elements in group when kill is ready
    public LinkedList<Group> ptr;
    /// 可选参数以表明原型的碰撞体积
    public double radius;

    /// 将图片画出来
    public void paintObject(Graphics g) { ...
    public Sprite(){...
    public Sprite(Rect rect) {...
    public Sprite(BufferedImage image) {...
    /// 检查一个精灵是否和一个组中的元素相交，若相交，返回检查到的第一个
    /// param 默认情况可以为1.0,表示相交区域的缩放程度
    public Sprite spritecollideany(Group group, CollidedFunc collidedFunc) { ...
    public ArrayList<Sprite> spritecollide(Group group, boolean doKill,
    CollidedFunc collidedFunc) { ...
    public void update() {

    }
```

流畅的用户体验

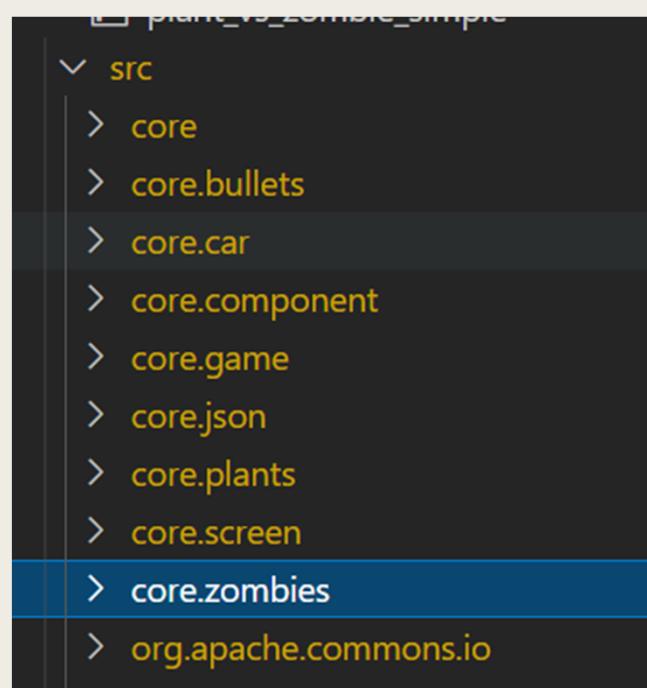
- 设计合理的顶层结构
- 封装良好的用户行为判定

```
    ...
    MouseAdapter l = new MouseAdapter() {
        @Override
        public void mousePressed(MouseEvent e) {
            x = e.getX();
            y = e.getY();
            if(e.getButton()==MouseEvent.BUTTON1)
                left_click = true;
            if(e.getButton()==MouseEvent.BUTTON2)
                right_click = true;
        }

        @Override
        public void mouseMoved(MouseEvent e) {
            x = e.getX();
            y = e.getY();
        }
    };
    surface.addMouseListener(l);
    surface.addMouseMotionListener(l);
```

合理的代码组织

- 顶层 - 底层两段式构造（见类图）
- 合理分package



展望

- 增加更多的玩法，如传送带玩法，坚果保龄球等等。
- 增加更多地图，更多植物和僵尸。
- 动画体验更加流畅。
- 增加更多选择界面，如关卡选择，音量、难度调整等等。

总结 & 教训

总结

- 采用了面向对象 (OOA - OOD - OOP) 的方法
 - 从对象入手规划我们的需求和需要实现的效果
 - 目标和任务更加清晰，避免了很多的中间转换环节和多余劳动
 - 加快了系统开发的进程
- 采用了敏捷开发的开发模式
 - 更新迭代整体而言是速度较快
 - 先层与层之间分离，后集成进行测试

教训

- 进行合理的单元测试后，再集成

个人开发体会

■ 刘珈征

- 这是我第一次在Github上与其他人一起，只有素材，从零开始搭建一个项目。深刻地感受到了Github的高效。非常感谢室友给我安装的GitHub Desktop，非常好用！调试的过程比较有成就感，比如看到点击开始游戏会跳转到下一个界面，可以种下植物，植物可以发射子弹，僵尸可以被打死等等。
- 但也反映出一定的问题。由于我们是一人完成一部分，因此在对接的时候会不清楚其他人的代码结构，传参、类型定义、以及某些变量的统一都出了很大的问题，又是发现，有很多时候费了千辛万苦，才发现是一行刷新变量的代码没有加上，导致了很大的错误，往往哭笑不得。
- 总体来说是非常棒的体验！非常感谢五位组员！在我调试的时候，每完成一步就给我很强的正反馈，让我有动力继续做下去！因为大家的共同努力才有了今天的这款游戏！
- 同时也感谢唐大仕老师一学期的指导！感谢助教们一学期的帮助与陪伴！

个人开发体会

■ 杨雨诺

- 我负责一半植物的编写包括父类plant.java，这部分代码的继承和多态设计比较关键。感谢组员们的信任和帮助让项目能够顺利完成。

■ 刘雨薇

- 这一次项目我负责的代码部分较少，更主要的工作是前期的文档工作和后期的汇报和总结工作，也在中期帮助了调试工作。很感谢组员们能够信任我的能力让我能够利用自己的经验给项目进行一个很好的说明，和一个足够展示亮点的报告。

Thanks!