**基于qml实现的游戏开发**

----植物大战僵尸

开发小组成员：周盈盈，张静

1. **引言**

本项目是基于qml提供的felgo开发框架开发的一款经典小游戏---植物大战僵尸。开发过程中使用git版本控制工具, 所有项目源代码和相关文档都已上传到github. 链接附在文档末尾.

1. **项目需求**

本项目需要实现的功能包括：游戏界面切换，背景音乐播放与暂停，选择植物，随机掉落阳光，太阳花生产阳光，收集太阳，植物拖动种植，植物发出子弹，子弹打中僵尸，僵尸死亡，土豆遇僵尸爆炸，樱桃放在草坪上爆炸，僵尸攻击动画，铲子铲除植物，小车铲除僵尸，僵尸分波进攻，僵尸进入房子游戏失败，僵尸全部死亡游戏胜利，使用太阳购买植物。

1. **界面设计**

游戏分为三个场景，每一个场景对应一个游戏界面。即开始游戏界面，选择植物界面，游戏进行界面。以上三个界面的UI原型图如下所示：



图3-1 “开始游戏界面”UI原型

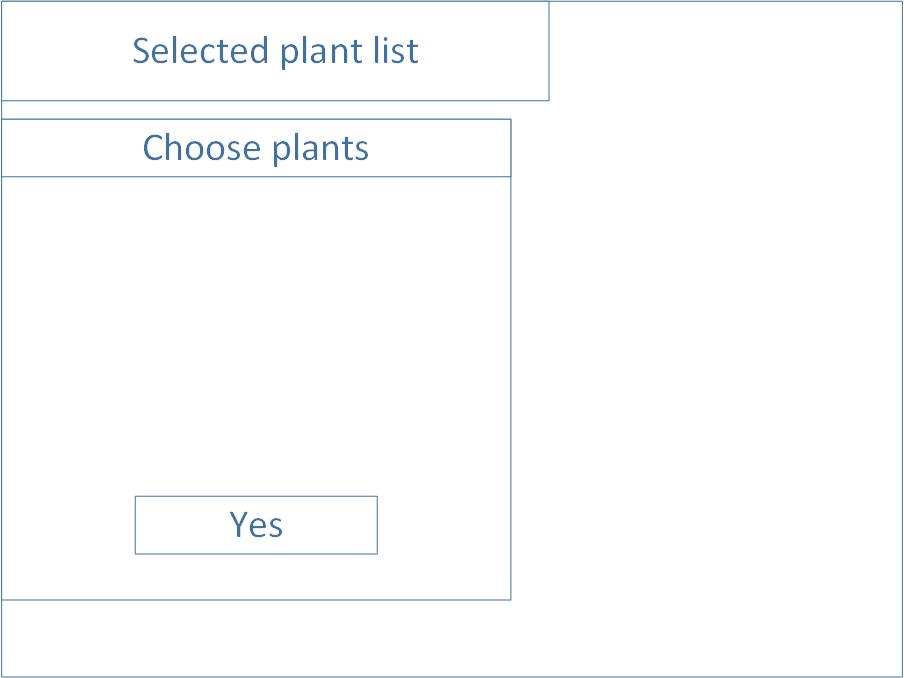


图3-2 “选择植物界面”UI原型

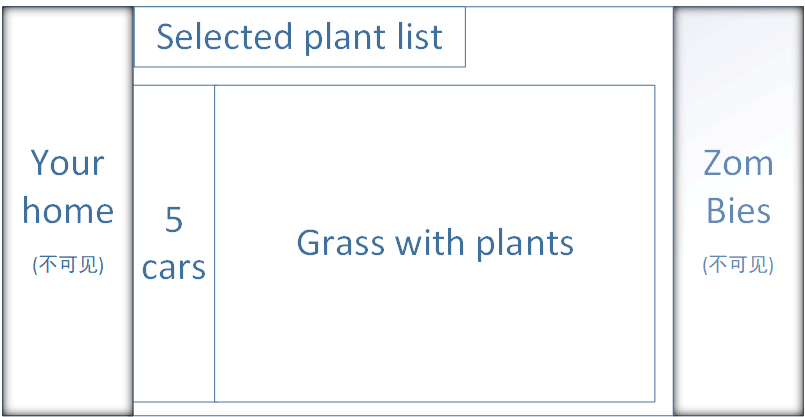


图3-1 “游戏界面”UI原型

1. **对象模型**

游戏基于面向对象的开发，使用面向对象语言qml和面向对象建模方法UML搭建项目的基本框架。并且依靠qml语言的对象树机制对游戏对象进行管理。其中主要的对象树有三个，分别对应Main.qml，Scene.qml，Welcome.qml三个文档文件；Main.qml为根对象树。据此画出的对象模型图如下所示：

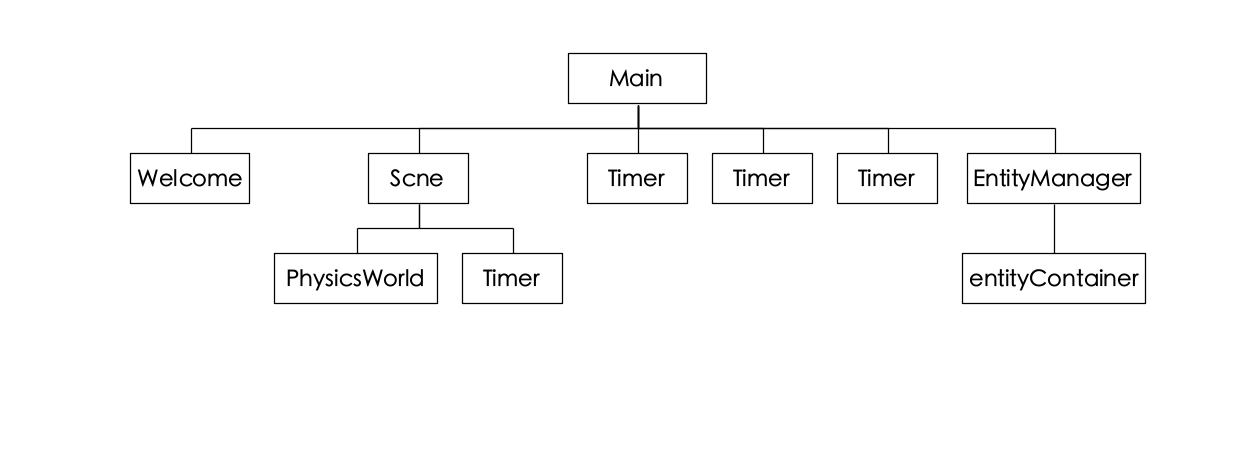


图4-1 “Main.qml”对象树

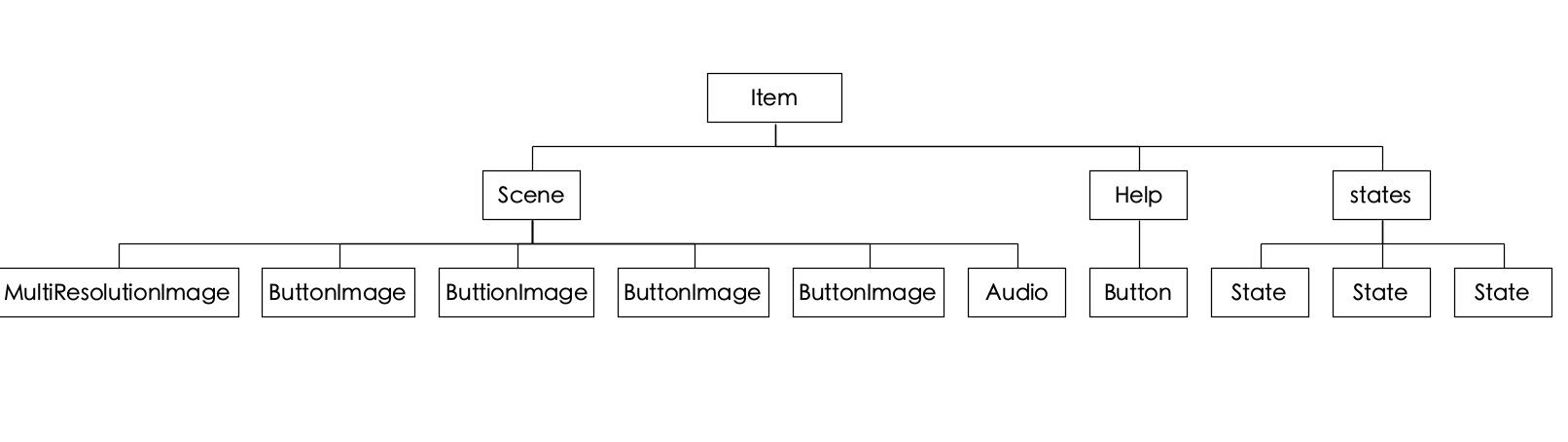


图4-2 “Welcome.qml”对象树

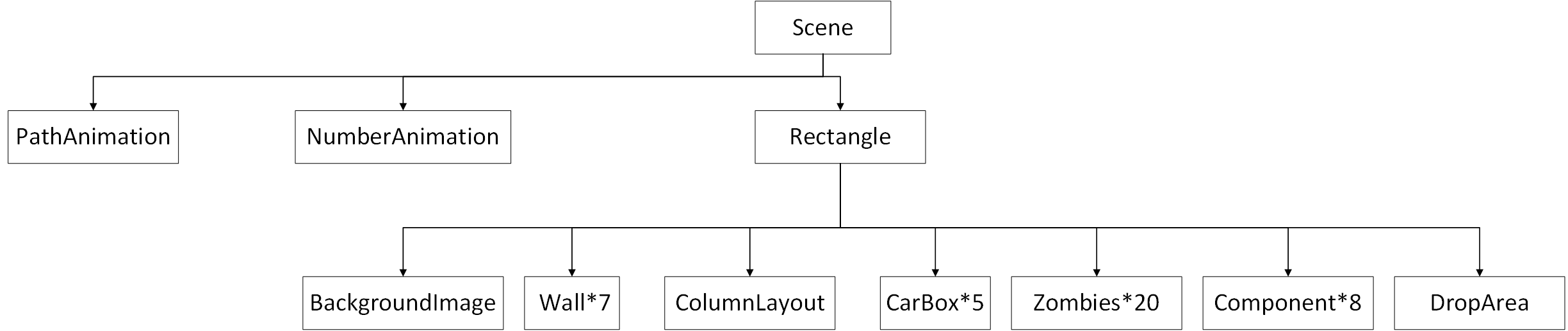


图4-3 “Scene.qml”对象树

1. **游戏逻辑**
2. 选择植物

功能概述：从植物卡池中选择植物到已选植物栏上，使玩家可以在开始游戏后拖动植物到草坪上进行战斗

实现逻辑: 游戏界面切换到选择植物界面后,初始化植物卡池和已选植物栏,此时植物卡池和已选植物栏都包含7张可选择的植物卡,但已选植物栏内的植物卡可见性被设置成false(visible=false),并且已选植物栏内的植物卡可见性与植物卡池中的植物卡接收的点击事件绑定.若要选择植物卡池里的植物,则点击对应卡片,同时已选植物栏中对应卡片可见性变为true,从而实现选择植物功能.

1. 场景移动

功能概述：预览战斗场景，先看到房子移动画面到草坪，再看路边的僵尸，最后定格到草坪上

实现逻辑: 游戏场景Scene中创建一个尺寸比游戏窗口的尺寸大的矩形对象, 命名为backGround. 并且该矩形只有与游戏窗口重合的部分可见, 设置该backGround的背景图片为游戏背景. 利用qml提供的PathAnimation对象类型实现矩形backGround的路径动画(左->右->左), 呈现出的效果即为草坪->房屋->僵尸.

1. 购买植物

功能概述：种植植物时太阳光总值减少

实现逻辑：设置一个全局变量totalsun作为玩家当前拥有的太阳总量，当拖动植物到草坪，创建植物时调用植物类型内部的Component.oncompleted（）函数，使totalsun减少对应的值，如果totalsun的值小于植物的价格，禁用拖动处理器。

1. 收集阳光

功能概述：天上随机掉落太阳，向日葵生产太阳，点击后阳光向左上方渐变消失，同时太阳总数加25

实现逻辑: 点击太阳时，让totalsun值增加25且摧毁这个实体的Taphandler避免重复点击时太阳的值多次增加

1. 种植植物

功能概述：用户拖动植物卡片，将植物种植在草坪上

实现逻辑：通过Drap和MouseArea结合，使植物可拖动且拖

动过程中产生虚影有一个动态拖动的过程，进入拖动区域后onDrapped时，通过传入drop.x和drop.y调用locationx和locationy两个自定义函数计算植物在草坪的位置坐标realx和realy,根据实体管理器entityManager.createEntityFromComponentWithProperties函数动态创建植物实体对象并在创建的同时将坐标传入，固定在草坪

1. 僵尸出现

功能概述：僵尸分批加载，出现在草坪上

实现逻辑: 使用计时器实现一波僵尸内部僵尸的出现时间, 计时器触发则开始僵尸的动画, 同时开启下一个僵尸的计时器. 定义一个全局变量zombiesdie记录当前死亡僵尸的数量, 若第一波僵尸已经被全部消灭, 即zombiesdie等于第一波僵尸数量时, 开启第二波僵尸动画的计时器.

1. 植物发射子弹

功能概述：部分植物检查到僵尸进入草坪后，发射子弹，子弹发出有时间间隔且当检查到草坪上没有僵尸后，植物不再发射子弹

实现逻辑: 创建一个自定义qml类型Wall作为墙. 一共设置五面墙在草坪右侧边缘, 且每一面墙与一个布尔状态变量flag绑定(初始值都为false). 若僵尸与墙发生碰撞,证明僵尸进入了草坪地界, 设置flag的值为true, 能发射子弹的植物检测到flag的值为true, 则开始发射子弹. 若子弹与墙发生碰撞, 证明僵尸已被消灭,植物停止发射子弹. 植物的判定逻辑需要同时符合植物在墙的flag值为true且植物的y坐标为当前列, 具体代码如下:



1. 子弹攻击僵尸

功能概述：植物的子弹与僵尸之间产生碰撞，僵尸血量减一，子弹产生特效：炸开消失

实现逻辑: 子弹检测到碰撞物体的id为以zombie开头的字符串, 则使用实体管理器创建一个代表碰撞特效的实体PeaHit,同样以qml文件的方式定义, 子弹碰撞后消失, 子弹的特效在显示一段时间后由Timer触发销毁. 僵尸检测到碰撞实体为子弹(同样通过判断对方实体id的方式确定), 碰撞处理时血量自动减一.

1. 僵尸攻击植物

功能概述：僵尸碰撞到植物有一个攻击植物的动作，且僵尸的血量不同，攻击动作也不一样

实现逻辑：僵尸的碰撞器检查到碰撞后，僵尸的状态改变为攻击状态

1. 植物死亡

功能概述：植物在被僵尸碰撞多次后，血量减少为零，植物消失

实现逻辑: 在僵尸类型内部定义反弹动画, 检测当碰撞实体为普通植物(除了爆炸型植物以外的植物)时, 碰撞检测处理器执行反弹动画, 使得僵尸在前进过程中能够多次对植物进行碰撞, 植物每被碰撞一次, 血量减一, 直到血量为0则销毁植物实体.

1. 植物被铲除

功能概述：拖动铲子到种植植物的草坪上，植物消失

实现逻辑: 拖动铲子的原理与从植物选择栏拖动植物原理相同, 鼠标拖动铲子到需要被铲除的植物的位置, 就创建铲子的实体对象, 碰撞后立即销毁. 若植物检测到碰撞对象为铲子, 则自行销毁.

1. 僵尸死亡

功能概述：僵尸被攻击血量清零，僵尸倒地消失，其中血量减少有掉脑袋的过程

实现逻辑：

1. 小车铲除僵尸

功能描述：当僵尸第一次尝试进入房子，与小车发生碰撞，小车铲除当前行所有僵尸

实现逻辑: 小车检测到与僵尸发生碰撞, 开始动画向右移动. 僵尸检测到碰撞实体为小车, 即销毁僵尸实体.

1. 判断游戏结束

功能概述：僵尸全部死亡后，判断游戏胜利；僵尸进入房子，判断游戏失败

实现逻辑: 胜利--初始化游戏通关的图片并设置可见性为false, 定义一个全局变量zombiesdie记录当前死亡僵尸的数量, 若zombiedie等于所有僵尸数,即所有僵尸都已经被消灭, 玩家通关, 设置游戏通关图片可见性为true. 失败--初始化游戏失败的图片并设置可见性为false, 在房屋的门口设置一面透明的墙Wall (可碰撞的实体), 当僵尸与该墙发生碰撞, 证明僵尸已经进入了房子,游戏失败, 设置游戏失败图片的可见性为true.

1. **开发难点**
2. 太阳随机掉落

问题描述：

解决办法：

1. 太阳收集动画

问题描述：

解决办法：

1. 植物拖动效果

问题描述：

解决办法：

1. 植物动态创建

问题描述：

解决办法：

1. 植物坐标确定

问题描述：

解决办法：

1. 禁止植物在已有植物的草坪播种

问题描述: 植物能够在同一块草坪重复播种, 需要摧毁后来种植的植物, 保留草坪上原有的植物.

解决办法: 设置一个全局变量plantnumber记录已经种植的植物的数量, 每个植物类型设置一个变量number记录自身的序数, 每当植物创建时, 执行Component.oncompleted函数, 使得number=plantnumber. 若玩家想要在同一块草地多次种植植物, 那么两个植物之间一定会产生碰撞, 故在植物类型定义里规定, 若与植物发生碰撞, 且碰撞时自身的number值和plantnumber相等, 即该植物是最新创建的植物, 则销毁自身, 而草坪上原有的植物不受影响.

1. 植物在僵尸出现时开始射击，在僵尸全被消灭后停止射击

问题描述：

解决办法：

1. 僵尸重复攻击同一个植物

问题描述: 僵尸在与植物进行碰撞后会继续往前走, 并且碰撞此时只会发生一次. 要实现僵尸被植物阻挡的效果和同一个僵尸能对同一植物多次进行攻击导致植物血条减少直至死亡的功能.

解决办法: 在僵尸类型内部定义反弹动画, 检测当碰撞实体为普通植物(除了爆炸型植物以外的植物)时, 碰撞检测处理器执行反弹动画, 反弹动画结束后僵尸继续前进, 使得僵尸能够实现多次对植物进行碰撞攻击.

1. **总结**

本次课程设计是一次学以致用的成果, 是一次较为完整的项目开发. 由于此前没有接触过felgo开发框架, 而目前国内关于qml的资源很少, 所以本项目的开发有些坎坷. 但在实践学习的过程中, 我们习惯了查阅官方的英文文档, 然后通过调试代码达到自己想要实现的功能. 这是一次对于自我学习能力的极大锻炼, 我们能够在十多天的时间内熟悉felgo游戏开发的部分核心内容, 例如碰撞检测, 实体管理器, 物理世界等等.

本项目基于felgo框架开发, 使用git版本控制系统进行管理. 开发过程中通过代码的编写对qml的对象树机制理解更加深刻, 通过版本的推送和拉取合并对项目开发的过程更加熟悉, 通过任务分工合作理解了团队合作的重要性以及节奏划分的重要性. 总之我们收获良多, 也很感谢有这次经历.

1. **参考资料**

本项目参考felgo官方文档和felgo官方模板项目示例进行开发.