# 元气骑士期末项目文档

## 一．项目信息介绍

### **1.项目名**：

《元气骑士》（《Soul Knight》）

### 2.项目进度时间表：

2021.5.01—2021.5.15 进行对cocos的学习

2021.5.17 确定游戏整体架构设计和方向

2021.5.25 完成场景基类的设计

2021.5.28 基本完成了地图的搭建

2021.6.1 完成了人物基类的设计

2021.6.4 完成了对玩家操控角色的制作

2021.6.8 完成了对子弹类和武器类的设计

2021.6.11 完成了敌人的对应设计

2021.6.13 添加入了音乐

2021.6.16 对游戏各个初始界面的功能进行了完善

2021.6.18-6.20 撰写项目文档，制作ppt

## 二．项目开发文档

### 1.游戏整体架构设计，小组分工：

##### （1）整体架构设计：

##### 

### 2.实现的功能点，实现思路：

#### 子弹的设计：

##### 子弹的移动：

对子弹的移动轨迹存在两个坐标点（Vec2），一个是发射子弹的人物所在位置，另一个则是鼠标所指位置，根据两个点相连所形成的直线即为子弹的运动轨迹。具体实现过程中根据endPosition（即鼠标点击位置）和startPosition（人物所在位置）作差从而得到每一帧的移动向量，再用getNormalized()函数对向量进行标准化，最后根据子弹速度（speed）与标准向量相乘即可获得每一帧子弹移动的向量。同时在子弹基类里提供一个外部接口以得到子弹移动的向量，在场景中调用子弹移动函数从而重新设置子弹的位置来实现子弹的移动，同时由于子弹位置的更新调用非常频繁，将其设置为inline函数。

##### 子弹的属性：

在游戏元气骑士中，子弹的属性主要包含了子弹的速度、攻击力和耗蓝量，在我们设计的子弹基类中，这些属性全部不设置初始值，因为基类有纯虚函数，在派生类的initbullet()更新子弹属性。

同时针对子弹碰到墙壁问题消失问题，我们对此的处理是写一个碰撞函数遍历vecObstacle和vecBullet并判断bullet是否与obstacle碰撞，为了节省时间，在update()中循环障碍物的时候不断调用它，并将其设置为inline函数以减少调用时的消耗，倘若发生碰撞则将该子弹从场景中删除，并将该子弹从vecBullet中删除。

#### 人物的设计：

游戏出现的角色的属性都包括人物所携带的武器类型、移动的速度、血量、蓝量、护盾量、人物矩形大小以及受到的伤害，因此我们也将这些属性设在了人物基类里面，后续游戏出现的player和Enemy均是通过继承人物基类而产生的。

针对人物矩形与障碍物碰撞的问题，我们用到了预碰撞的处理方法，即当操控者进行移动时我们会先对移动后的位置进行碰撞的判断，倘若两者发生碰撞则函数返回而不进行下一帧的移动，这样以获得无法移动到墙外的效果。

我们共设置了三个可供操控的角色，分别是游侠、圣骑士和骑士，我们也将其用全局变量表示，每个角色也有各不相同的属性，如游侠有更高的机动性，圣骑士有更高额的伤害，骑士则是比较朴实廉价的存在。我们还在player的派生类中加入了不同角色的特殊技能，比如游侠可以提高自己的灵活性等等。当操控者选择好角色时，我们通过intiRole（）函数对所选角色的移动速度，血量、蓝量、护盾和携带武器进行相应的初始化，并将人物设定初始位置设到中间以及设置好初始人物矩形。当人物发生移动时同时对人物矩形的位置和武器位置进行更新，同时每当人物发射子弹都会对蓝量进行扣除，受到伤害会对护盾值进行扣除，同时对护盾和蓝量进行判断，若不为满则每秒进行一定量的回复直到为满为止。

针对Enemy我们共设置了两种形态，分别是水晶虫和矿工。其中水晶虫为近程敌人，矿工为远程敌人，两种敌人均用vector进行储存。对敌人的移动思路，我们是将endPosition设为玩家所在位置的坐标，将startPosition为自身所在位置来得到移动的单位方向向量，若是近程敌人就按该方向靠近玩家，若是远程敌人则是按照该方向远离玩家，得到移动坐标后更新敌人的位置和敌人矩形。如果子弹与怪物的矩形发生碰撞则调用alterBlood\_函数对怪物血量进行扣除，如果敌人的血量小于等于0则将其删去。当玩家所在矩形与怪物矩形发生碰撞，则通过判断敌人的damage对玩家进行血量扣除。需要注意的是在我们设计的人物基类中实际上有些属性和函数对于Enemy是没有意义的，比如将初始位置设为中央等等。

#### 地图的设计：

**游戏中我们共设计了六类场景分别是StartGameScene、RoleSelectScene、SuspensionScene、GameEndScene以及GameScene。**

##### StartGameScene、SuspensionScene和GameEndScene

在StartGameScene中比较简单，分别是游戏的菜单界面、游戏作者信息页面以及游戏的设置页面。游戏菜单界面可以跳转到另外两个界面也可以进入到角色选择页面。点击设置即可进入设置页面对游戏的声音等进行调节，点击叹号即跳转至游戏作者信息页面，这里可以看到小组成员的基本信息。

SuspensionScene的功能则是使游戏暂停，当进入暂停画面背景板全黑，并可以在这里对音量的大小进行调节或是返回菜单，实现比较简单。

GameEndScene有游戏成功和失败两种情况，这里的实现也比较简单。

##### RoleSelectScene

进入到角色选择页面时，可以看到在画面上有三个玩家可操控的角色，选择你想操控的角色即可跳转至角色的属性面板，在这里点击箭头即可进入游戏，也可以点击返回键重新返回到角色选择页面重新选择。

##### GameScene

该部分我们先设计了基本的场景基类，因为在这部分上地图上涉及到很多元素的互动。首先是在操控层面，我们用键盘监听事件对输入进行判断，对于长按和释放，我们采用的方式是声明一个bool型的变量来判断某按键是否被长按，若一个键位被按下则为true，直到释放时才会将true改为false来停止该动作的继续执行，这种情况下对于移动的实现就比较简便，在moveRole函数中只需判断移动方向的几个键位是true还是false，若为true则先进行预碰撞的判断再进行移动。另外当人物在场景切换时出现的位置，我们调用setInitialLocation函数判断，倘若人物的位置在左边切换场景后即出现在场景的右边，反之同理。

我们还将update相关的函数加在了这里以来实现每一帧各个人物矩形、子弹移动和血量等属性更新的功能。

在前面场景基类的基础上我们做了三个相应派生的小屋，分别是玩家出生的小屋、敌人存在的小屋和放有传送门的小屋。对于场景的生成，我们使用Tiled生成的.tmx文件去创建一个TMXTiledMap类的对象，然后将这个tileMap直接addChild，这样.tmx文件里面的图块内容就都进去了。再分别提取对象层和对象层中的对象，根据提取出来的对象在预定的位置上加入敌人。Titled允许自行设置对象的属性，其中"x","y"是自带的属性，"Length"和"Width"则可以自行添加。

进入游戏后游戏的左上角也会显示当前操控角色的血量、蓝量和护盾量，这里我们用到的方法同样是Tiled贴图并取对象，同时获取主角的血量、蓝量和护盾量以及他们的最大值，并每一帧对这些属性进行更新以达到始时展示角色属性的功能。

#### 武器的设计：

##### （1）武器基类的设计：

武器的设计同样包括了武器所带子弹的类型，子弹速度，子弹伤害以及武器耗蓝量这几个基础属性，因此在写BasiceWeapon基类时同样包括这几个属性并在后续派生类进行初始化。

##### （2）武器的派生类：

我们制作了破旧的手枪和先驱者两种武器由player使用，当武器确定后用initWeapon函数对武器的子弹等属性进行初始化，并在每一帧更新其位置使其与玩家操控角色处在同一位置。

#### 音乐的添加：

针对音乐的添加，我们include了"AudioEngine.h"头文件，以一个bool变量来表示音乐的播放与暂停，并且用到了AudioEngine的stop、resume、pause等函数来控制音乐的开始、停止、暂停和继续，用setVolume函数来控制音量的大小，玩家。

### 3.项目的技术难点，解决方案和解决过程：

##### （1）场景搭建与对象获得的交互

在场景搭建后倘若通过代码来取场景中的一些对象如墙等等会十分不好处理，对此，在群里学长的点拨下我们进行了对于Tiled的学习，通过网上的资料查找和摸爬滚打之后大概掌握了Tiled的使用，借由Tiled我们可以对地图进行定量的刻画并且取得所需要的对象层，在对象层我们可以取得墙的位置以判断碰撞问题，也可以在设置的对象位置生成相应的敌人，这为我们在交互层面简化了不少。

##### （2）人物移动的处理

对于人物移动的处理主要集中在三个方面的问题，一个是人物与障碍物碰撞的判断问题，一个是人物移动过程中键位的长按以及释放的判断，还有一个就是敌人的移动方向。通过资料和思考后，我们对人物与障碍物碰撞的处理方式是采用预碰撞在角色未移动到相应位置后先对位移后的位置进行判断。对于键位长按的处理我们则用了bool变量来表示键位的执行情况，若该键位为true则执行，反之则不执行，用这样来解决长按的问题。针对敌人的移动方向，我们考虑到敌人的属性，将敌人和玩家操控角色作为两个点来获取移动的方向，近程敌人靠近玩家而远程敌人则是远离敌人。

##### （3）场景的设计

因为当每次进行场景切换时我们的做法都是重新生成场景及人物，因此在切换场景后我们还需要保证对人物属性的更新，为此我们将玩家选择的人物、武器、血量、护盾和蓝量作为全局变量并在人物基类写一个updateRoleAttribution函数在每次场景切换时都对这个函数进行调用来做到对人物属性的更新。

### 4.亮点：

##### （1）人物碰撞

我们对人物碰撞的判断采用了预碰撞的方式即先假使人物发生移动再对移动后的人物是否与存在矩形发生碰撞进行判断，倘若发生碰撞则直接返回而不发生此次移动，如不发生碰撞则正常移动。

##### （2）敌人的设计

游戏设计了近程敌人和远程敌人两种，并且两种敌人均有不同的特性。水晶虫通过靠近英雄并用碰撞来对操控者产生伤害，而矿工作为远程敌人则是通过与英雄拉开距离而进行远距离输出。

### 5.运行截图：

