《魔塔》设计文档

山东大学 王泽宇

一、项目目标

使用 JAVA 程序设计语言和 JavaFX, 实现经典魔塔。包括地图的显示、动态图块的显示、人物行走、状态栏、商店、事件、物品获取、战斗、门等。

二、程序结构

有 battle、commonFunctions、data、director、event、model、music、rpgSwitch、scene、spirit、terms 十一个包。

（一）数据（data）

常用数据常量和数据类型的定义。包括类 CharacterData、CommonData、EnemyCharaterData、ImageData、MapData。

1、CharacterData

定义了基础游戏对象需要的属性：名称，脸图路径，生命值，攻击力，防御力，可以通过访问修改函数进行访问修改。

2、CommonData

包含一些常用数据。包括运动动画的速度、精灵动画模式事件、游戏已设计最大层数。所有成员变量都是 public static final 的，且这个类不能被实例化。

3、EnemyCharacterData

预定义一些敌人单位。内涵 public static CharacterData 的 enemyXX。均为一些预定义好的敌人单位。

4、ImageData

一些 public static final String，存储了大量的图标地址。

5、MapData

存储地图信息。包括地图大小数据（图块文件高度宽度、图块显示高度宽度、脸图高都、脸图宽度），行走图及其对应脸图的路径，行走图编号（常量表示具体意义），一般地图行列数，一般地图宽度高度，人物行走方位常量，图块路径、通行状态、图块显示类型，图块编号，具体地图，具体地图大小，具体地图人物默认初始位置和朝向，获取编号为 k 的图块对应的图块精灵的方法。

（二）开关（rpgSwitch）

定义了游戏开关。

1、RpgSwitch

包含一个私有的可绑定布尔类型开关 rpgSwitch，可以获取/修改。主要是为了能够传引用。为了后面写公共事件方便。

（三）模块（model）

定义了一些随玩家动作会改变的量，用于以后写存读档。包含 Model 接口，ModelPlayerData 类,ModelSwitch 类

1、Model

要求实现 fileInput()和 fileOutput()两个函数。

2、ModelPlayerData

玩家数据。一系列绑定数据，如等级、hp、攻击等战斗数据、移动速度、坐标、钥匙数、金钱数等、所在楼层。拥有初始化方法。均为 public static，不可实例化。

3、ModelSwitch

定义了一系列开关。包括独立开关（私有开关）和公共开关。每个事件有且仅有一个独立开关（私有开关），用来控制事件的显示、可触发。公共开关为所有时间公有。通过注释的形式标注每个开关是否被使用和使用目的。包括获取公共/私有开关方法，如果为空还能赋初值。每个开关只有在第一次被获取时被分配空间并初始化。

（四）用语（terms）

包含了游戏常用用语常量。

1、Terms 类

定义了各类 public static final String 的游戏常用语言。如 public static final String attack = "攻击";

（五）音乐（music）

链接了一系列音乐资源

（六）通用函数（commonFunctions）

集合了一些常用的通用函数。

1、CommonFunctions 类

定义了在一张 Image 上按照恒定宽度高度，按照先行后列顺序，将图片标号，获得某个标号图块的 ImageView 或者直接将其绘制到给定的 GraphicsContext。

（七）精灵（spirit）

基本的动画单位。根据用途有 Spirit、SpiritGame 和 SpiritTile 三个类。

1、Spirit 类

有一个图片集合 imgSets，要求把动画的每一帧都摆放在其中。按照先行后列次序编号。动画序列用编号表示。本身是一个 Canvas，执行动画的原理是通过 Timeline 无限次定时运行 KeyFrame 中的 EventHandler，在 EventHandler 中对 Canvas 中的画布 GraphicsContext 进行逐帧绘制。包括自由帧数模式、单帧模式，同时也可以继承这个类自主编写 EventHandler

实现新的图片运行模式。

2、SpiritTile 类

图块精灵。显示在地图上的图块。魔塔游戏中，图块有静止和四帧动画两种模式。静止模式基类 Spirit 已经拥有，所以在 SpiritTile 中实现了四帧动画模式。

3、SpiritGame 类

游戏单位精灵。能够行走并发出行走音效。重写了 Spirit 中执行的 EventHandler。采用较为复杂的逻辑处理人物行走。用 dir 表示当前朝向，state 表示行走状态（静止 1，迈左脚，静止 2，迈右脚），nowFrame 表示当前在第几帧，walkDir 表示当前行走方向（不走时为-1，同时也能检测是否在行走动画）。每行走一次要执行 4 帧动画，通过 tmpFrame 监测动画到第几帧了。4 帧动画分别为静止、迈脚、迈脚、静止，tmpFrame 监测行走到第几帧，并具体决定当前显示帧 nowFrame。由 state 决定是迈哪只脚。

拥有改变朝向、向四个方向走一格的方法，以及改变行走速度的方法。行走速度是播放行走动画一帧的时间（ms）。

（八）战斗（battle）

处理魔塔战斗相关。包含 BattleData 和 BattleUI 两个类。BattleData 用于处理战斗数据，BattleUI 用于处理战斗显示。

1、BattleData

包含两个 CharacterData:enemy 和 player，通过构造方法浅拷贝传输进来。可以通过直接修改这两个 CharacterData 来直接操控 UI。包含三个方法：battleHP 返回战斗损失血量（如果我放攻击力小于等于地方防御力返回-1），canBattle 返回能否发生战斗，nextRound 表示向下进行一回合，并操纵构造方法传入的 enemy 和 player。

2、BattleUI

包括一个游戏主场景的引用 rpgSceneGame 用语往上面绘图，还有玩家信息和敌人信息，以及一系列面板文字。构造方法直接构造出战斗面板。有绘制函数 draw 将战斗面板绘制到游戏主场景上并运行战斗。

Draw 返回是否能够战斗。Draw 首先判断能够战斗，不能战斗返回 false，能战斗执行战斗逻辑。首先将战斗面板 battlePane 加入到游戏主场景 rpgSceneGame 中，然后用 Timeline 充当计时器，每 550ms 执行一次战斗逻辑，即 BattleData 中的 nextRound。战斗结束后取消面板显示，并通过标记的形式让 timeline 不再执行任何指令。同时根据战斗结果，相应修改玩家数据。

（九）事件（event）

事件是玩家触碰后会运行的代码。主要是 Event 这一个抽象类，所有的具体事件都要继承这个类。此外，包内还定义了一系列需要传入独立开关的公共事件类，如红钥匙、红门、怪物等等。这里不再介绍。主要介绍 Event 类。

1、Event 类

包含一个精灵，用于事件的显示。还有游戏主场景和游戏界面，用于绘图。包含一个 run 方法，用于启动事件，监测事件独立开关是否开启，开启则运行事件，将键盘聚焦到当前面板，并且禁止人物移动（通过设定移动开关），然后执行 work 方法。Work 方法需要自行实现。

包含一些列定义好的事件方法。如场景跳转事件、战斗事件。

（十）场景（scene）

用于游戏界面显示。包括 RpgSceneEvent、RpgSceneEventMain、RpgSceneMain、RpgScenePane、RpgSceneTile、RpgSceneWalk、RpgStorePane。

包含关系为：

RpgSceneMain 为主场景，包含了一个 stackPane rpgSceneGame 游戏主场景、左侧面板 RpgScenePane 和右侧 RpgStorePane。内有按键交互管理。左侧面板和右侧商店较为简单，不再介绍。rpgSceneGame 堆叠了游戏的心脏：RpgSceneTile，RpgSceneWalk 和 RpgSceneEventMain。提到的顺序为自下而上。RpgSceneTile 为图块场景，向 GridPane 中加入精灵。RpgSceneWalk 为行走场景，提供人物行走方法，显示游戏对象精灵（SpiritGame）。RpgSceneMain 是事件主场景，负责具体的各个事件的加入，有事件获取方法，内有一个 RpgSceneEvent 场景。这个场景是一个 GridPane，每一个单位加入了一个固定事件，用于事件显示，有事件获取和加入方法。

（十一）Director

包含一个类 director。没啥用，就是播放 BGM，然后实例一个 RpgSceneMain，加入到一个 Scene 中，再加入到 Stage，然后运行。