# 项目总结

## 实现的功能

|  |  |
| --- | --- |
| 分数系统 | √ |
| 碰撞系统 | √ |
| 渲染精灵 | √ |
| 渲染动画精灵 | √ |
| 玩家跑步 | √ |
| 玩家跳跃 | √ |
| 玩家降落 | √ |
| 星星随机生成 | √ |
| 障碍物随机生成 | √ |

## 核心代码说明

* 游戏模块

游戏添加了两个模块，Player模块和Zombie模块，用来创建新的Player和新的Zombie，使用闭包形式。（PS：使用元表一直出问题）

1. 分数系统模块:
   1. 用全局变量监听当前获得的分数;
2. 碰撞系统模块collider(obj1s, obj2, action):
   1. 检测两个对象的x与y轴是否有交集，有则将两个对象放到执行函数的参数里;
3. 渲染精灵模块DrawSprite(sprObj):
   1. 由于游戏特性，精灵一直向着左边移动来模板地图在向右移;
   2. 由于游戏特性，离开地图的精灵将会在下个地图以新的位置出现;
   3. 由于有随机性，新的位置出现的精灵具有随机性;
   4. 精灵一旦离开屏幕将不会被渲染;
4. 动画精灵渲染模块DrawAni(aniObj, FPS):
   1. 被设定了一个新的帧率;
   2. 每帧将移动到下一帧，到尾部时重新回到开头继续计算;
5. 玩家运动模块InputMessage():
   1. 跑步时: 渲染动画精灵;
   2. 跳跃时: 有限制高度、有滞空最长时间、限制键盘操作;
   3. 降落时: 模仿物理特性，跑步时的重力特性同理;
6. 星星与障碍物随机生成:
   1. 在渲染精灵之后修改成新的x和y轴;

* Zombie模块

在构建时赋值：僵尸在table中的索引、图片ID、宽高、Y轴位置、左右方向、动画帧速度、行走速度。

函数返回：Render, GetPos, SetDead, isDead, CheckCollision。

Render：渲染僵尸到游戏画面

GetPos：获取僵尸位置

SetDead：设置僵尸死亡

isDead：判断僵尸是否死亡

CheckCollision：判断僵尸碰撞

* Player模块

在构建时赋值：图片ID、XY轴位置、宽高、动画帧速度。

函数返回：Render, KeyListener, isAttack, GetDirection, Destroy。

Render：渲染Player到游戏画面

KeyListener：对 A D键进行监听，如果按下改变人物方向并进入攻击模式

isAttack：返回人物攻击状态

GetDirection：获取人物方向

Destroy：销毁人物

* LuaGameHost

定义collider表存放左右攻击范围和人物受击范围。

定义zombies表存放Zombie模块生成的僵尸。

定义playerMethods表存放Player模块生成的游戏玩家。

在HandleFrame函数中进行对三种游戏场景的判断：开始游戏、进入游戏、游戏结束。

在进入游戏（HandleGameScene）函数中依次进行了：渲染得分、渲染生命值、创建僵尸、渲染僵尸、渲染玩家。

创建僵尸：使用定时器实现，定时1秒创建一只僵尸，通过对Zombie模块的调用，将返回的值用table.insert方法加入zombies表中。

渲染僵尸：通过使用迭代器对zombies表遍历，如果僵尸没有死亡则渲染，并对其与collider进行碰撞检测。

渲染玩家：如果playerMethods中没有渲染方法，则对Player进行调用，其返回值加入playerMethods。如果存在渲染方法，则对人物进行渲染并监听键盘。

## 耗费工作量

大概花了12小时，一开始没有将各个子功能独立出来，每个对象的渲染代码都写了出来，导致代码冗余，阅读不通畅，逻辑不清晰。后面重新写了一次，以将代码量缩减到最小，逻辑更清晰，将渲染代码独立出来成一个模块、碰撞检测独立成一个模块。游戏逻辑也独立出来，但是经验不足，有待继续优化。

## 完善计划

1. 细分场景模块:
   1. 游戏开始前;
   2. 游戏不同关卡开始进行时;
   3. 游戏结束时;
   4. 游戏拓展业务展示;
2. 随机系统优化:
   1. 星星的随机除了位置的随机还要个数的随机;
   2. 障碍物的随机要更加合理，例如椅子不能悬空等;
3. 关卡场景优化:
   1. 目前场景只有背景、地面、玩家、椅子、星星，可以拓展更多的内容增加可玩性;
   2. 使用更加合理的算法来完善逻辑、方便拓展与修改;
4. 渲染系统优化:
   1. 渲染系统目前还带有位置逻辑判断，应该将逻辑判断整合到逻辑系统;
   2. 渲染系统的渲染动画精灵需要补充触发函数;
5. 物理系统优化:
   1. 碰撞系统目前只是浅薄的知识去写，有待继续完善为更加合理的碰撞系统;
   2. 补充更多的物理特性;

## 学习体会

使用该游戏框架相比使用windows.h提供的api，可以节省很多实现功能，不用考虑具体的实现细节，让开发者只专注于游戏逻辑开发，节省很多的时间;

使用lua来编写代码，可以减去编译的时间，加大了开发者的效率。

余庆祥

2220631136

2220级数字媒体技术1班

2022/12/25