雷电游戏作业报告

yujie6@sjtu.edu.cn

2018 年 10 月 28 日

目录

1 主要功能的实现

1.1 物理碰撞

物理碰撞是这个游戏不可或缺的一环,几乎每个功能都要用到碰撞,如果写一个crash函数来判定会方便很多,然而当我想到可以优化时我几乎已经做完了,所以我没有进行这个强大的优化,这导致我的代码关于碰撞的部分十分冗长丑陋,而且这样的部分有十来个,这个故事告诉我写代码前要好好规划.

实际上想要实现碰撞的判定还是很容易的,只需判断两个object是否相交,这里我们默认其为矩形,最后我们可以将其转化为横纵坐标以及长宽的一个不等关系来判定,同样的我们可以处理玩家或敌机越界问题.

1.2 子弹的连续发射

实现子弹的发射是容易的,只需定义子弹坐标的数组以及子弹状态的数组,按space键后状态变为true,初始坐标为玩家的坐标,如果击中敌机或是射出屏幕状态变为false.

首次尝试的我遇到了一个小问题,就是子弹发射得及其连续,我们需要长按空格时控制子弹的发射频率以获得更好的体验,经过思考,我发现可以通过定义一个spacetime来解决,每按一次空格,spacetime在mod 15意义下加一,当spacetime为零时才发射.我们可以用相同的操作实现敌机爆炸特效.

```
if( keyboard[KEY_SPACE]){
        if (spacetime == 0) {shoot;} //shoot is abbr
        spacetime = (spacetime+1) % 15;
        }
```

1.3 道具系统

道具系统大概是我大作业中最有趣的功能,首先我用结构体object定义了三个道具,效果分别是超级武器加一,生命加一与子弹多一列,它们的出现并非随机,而是按分数取模后为一个特定值时,在死亡的敌机处产生,从而控制出现的频率.通过常规的物理碰撞的判定来决定是否出现效果.

```
struct Object{
int width;
int height;
PointD pos;
bool out;};

Object SWplus, liveplus, bullet, morebullet[20];
```

生命与超级武器加一的效果十分容易实现.唯一有些麻烦的是子弹多一列的实现,因为我只能定义结构数组,一个数组代表一列子弹,想要多一列子弹必须多定义一个数组.事实上我觉得我可以实现三列甚至更多的子弹列,而且这肯定大幅能提升游戏性,但由于我很懒,我最后只实现了两列.

```
if (bulletplus.out){
  if (crash){  // crash is abbr
        bulletplus.out = false;
        bulletok = true;
        score += 5;}  }
```

就这样通过obeject自带的out状态量来判断是否显示这个道具,用新定义的bool量`bulletok`来判断是否发射两列子弹.如果是,则像之前那样稍微改变一下子弹的初始横坐标就行了.我还设置了一个小细节,如果player在`bulletok`状态下受到伤害,两列子弹的效果会消失.

1.4 Enemy系统

同样的我用object定义了boss,根据游戏进行的时间boss会出现三次,并且其血量会逐渐增加,超级武器会对boss造成大量伤害但不会直接杀死,杀死boss可以恢复生命值并获得大量点数.为了凸显boss的强大以及维持平衡性,我用了一个比较宽的坦克作为boss,player可以很容易地命中它.

```
if (timeoff == 600) {boss.out = true;}
if (timeoff == 2400) {
boss.out = true; boss.pos = PointD(SCREEN_WIDTH/4, 0);
bossblock = 80; velocityboss = PointD(0.3, 0.7);}
if (timeoff == 4200) {
boss.out = true; boss.pos = PointD(SCREEN_WIDTH/2, 0);
bossblock = 120; velocityboss = PointD(0, 0.5);}
```

`timeoff`是我设置的时间变量每次work它的值都会加一.除了boss之外还有三架随机出现的敌机,它们十分愚蠢只会直走,而且子弹也并不智能,你只要命中5发子弹就能击杀,每次击杀可以获得5个点数,player也可以按`"f"`使用超级武器来清除所有敌机和敌人的子弹.

2 代码优化

2.1 struct的使用

最初的时候没有想到用struct,导致我用了一个PointD数组`posbullet[20]`和一个bool数组`outbullet[20]`来表示子弹,后来才发现几乎每一个元素都需要这么两个数组,这个时候定义一个object就会方便许多.再后来我发现每个元素还有长宽这两个性质,于是又在object中加入`width`和`height`,用`getimagesize()`来读入.

```
getImageSize(imageliveplus,liveplus.width,liveplus.height);
getImageSize(imageSWplus,SWplus.width,SWplus.height);
getImageSize(imageBulletplus,bulletplus.width,bulletplus.height);
getImageSize(imageboss,boss.width,boss.height);
```

2.2 使用Git和Github

由于我是在Linux下编写程序,想要在Windows下修改和加入一些东西变得有些麻烦而Git恰好可以解决这个问题.之前一次上机课讲了Git的使用,~~然而我几乎摸不着头脑~~,后来经过一定实践我粗略的掌握了Git和GitHub.下面主要讲讲我关于Git的学习.

2.2.1 远程仓库

添加SSHkey后就能将GitHub与本地仓库连接了,通常我们先在GitHub上创建repository,然后再从本地克隆.或者在本地创建,然后用remote add连接GitHub上的仓库,之后就可以进行推送,但此时对应仓库例不能有任何东西(包括readme),否则会出现rejected,当然我们也可以加-f参数来强制推送.

```
$ git remote -v
$ git remote add origin git@github.com:yujie6/repositoryname.git
$ git clone git@github.com:yujie6/repositoryname.git
$ git push -f origin master
$ git push -u origin master
$ git push origin branchname
```

origin是远程库默认的名字,我们可以用git remote命令来查看远程库.添加远程库之后我们可以用push来继续推送,第一次推送时加上-u参数来连接相应的分支.

2.2.2 分支

分支是git中十分有用的功能.我们用git branch来创建和查看分支,同时也可在不同分支中自由转换, 添加-d参数还可进行分支的删除.

```
$ git chekout -b dev
$ git branch dev
$ git checkout dev
$ git branch -d dev

$ git merge --no-ff -m"message" dev
$ git log --graph
```

不同分支还可进行合并,但是合并可能会有冲突,解决冲突就是把Git合并失败的文件手动编辑为我们希望的内容,再提交.同时还可以用git stash来暂存分支,这样做是因为我们在dev中进行的修改在工作区,工作区和暂存区是一个公开的工作,任何分支都会用到,并能看到工作台上最新的内容,只要在工作区和暂存区的改动未能够提交到某一个版本库(分支)中,那么在任何一个分支下都可以看得到这个工作区暂存区的最新实时改动.使用git stash就可以将暂存区的修改藏匿起来,使整个工作台看起来都是干净的.所以要清理整个工作台,那么前提是必须先将工作区的内容都add到暂存区中去.之后在干净的工作台上可以做另外一件紧急事件与藏匿起来的内容是完全独立的.

```
$ git stash
$ git stashlist
$ git stash pop
```

3 反思与建议

3.1 大规模程序编写体会

本次大作业我花了大概20个小时去完成,最初的时候根本摸不着头脑,大概花了一个小时才粗略地明白该怎样操作.这也导致我最初编写的时候不太有大局观,定义了许多琐碎的变量与数组,后来才发现有许多可以统

一和简化的地方(然而我已经懒得去改了).不仅如此,代码的美观性也十分重要,否则当代码量上去之后,想要进行修改或是分享给他人都十分困难,独立的功能可以写成函数或库,这次我的代码中有许多重复的内容,可读性也较差.

上手之后我便发现其实整个程序的编写并不困难,大多数都是对先前的代码进行再加工,然而我发现一个很严重的问题,每当我想到一个具有可行性的新功能时,我都会因为懒惰以及对bug的恐惧而一再拖延,直到确实找不到什么简单的可以加的功能后才会去考虑之前想到的功能,最后发现这个新功能其实并不难实现,只是我内心的恐惧在作祟.所以以后要是想到什么新功能,直接大胆加上去,当然也可以借助git开一个新的分支(不过我暂时还不是那么熟练),总之不要畏惧新的bug,只要投入足够时间,想要做好并不困难.

3.2 关于大作业的建议

1. 添加音频对Linux用户有点不太友好,好像要手写一个`find_SDL2_mixer.cmake`才能实现.
2. 图片文件夹中有几个图片可以实现动画,但文件中好像并没有对这几张图片的说明.
3. ~~以后讲Git最好能够将理论与实践结合起来,如果能直接演示或是让同学边看边练就好了.~~

3.3 结语

这次大作业相比其他作业算是相当有趣了(虽然刚接触时我有些震惊甚至怀疑自己是否能够完成),而我在本次作业中确实受益良多.最后的最后,感谢各位助教的付出与指导,热心的刘珺琪学长和陆晗学长给了我很多帮助,祝大作业越办越好.