|  |
| --- |
|  |
| **实验报告** |
| **课程名称： 计算机算法与程序设计**  **学院专业：学生姓名：**  **学　　号：**  **组员名字列表：**  **指导教师：戴波** |
| **日　　期：21 年 6月19日** |

摘 要

弹珠消除的一个经典游戏是打砖块游戏，这类游戏的基本规则是，玩家控制一个平板，有一个小球在平面内移动，碰到平板，砖块或者墙壁会出现反弹，如果小球掉到平板下，则游戏失败。这一类游戏在早期红白机上就已经出现。现在，PC机上也有各种版本的以打砖块为基本模型的游戏。

本游戏把砖块替换为了圆形弹珠，其他规则不变。本系统是基于python和Qt框架，采用面向对象技术实现了弹珠，弹球，平板等各个元素，通过弹珠的渲染，碰撞检测等优化了游戏的功能，并添加了生命值，关卡等额外元素，提升了游戏性。

**关键词：**弹珠消除；python；Qt；渲染；碰撞检测

目 录

第一章 绪 论 1

1.1 贪吃蛇游戏介绍 1

1.1.1 历史与创新 1

1.1.2 操作方式 1

1.2 引用模块列表 1

第二章 系统分析 3

2.1 功能分析 3

2.1.1 基本功能 3

2.1.2 拓展功能 3

第三章 详细设计及实现 5

3.1 地图选择界面 5

3.2 物品类 5

3.3 分数上传/下载 8

3.4 自定义地图 8

3.5 额外道具 9

3.6 蛇头与蛇身 10

第四章 测试 13

4.1 地图选择界面测试 13

4.2 移动测试 14

4.3 物体吞食测试 14

4.4 游戏结束测试 15

第一章 绪 论

1.1 贪吃蛇游戏介绍

1.1.1历史与创新

1976年，Gremlin平台推出了一款经典街机游戏Blockade。游戏中，两名玩家分别控制一个角色在屏幕上移动，所经之处砌起围栏。角色只能向左、右方向90度转弯，游戏目标保证让对方先撞上屏幕或围栏，就是两条每走一步都会长大的贪吃蛇比谁后完蛋，玩家要做的就是避免撞上障碍物和越来越长的身体。

Blockade 很受欢迎，类似的游戏先后出现在 Atari 2600、TRS-80、苹果 2 等早期游戏机、计算机上。但真正让这种游戏形式红遍全球的还是21年后随诺基亚手机走向世界的贪吃蛇游戏——Snake。

贪食蛇游戏操作简单，可玩性比较高。但由于过去的平台限制，蛇只能四相移动且游戏画面为像素风格。本作对其进行了创新——基于pygame，制作一个全向移动的新贪吃蛇。

在新贪吃蛇中，玩家可以控制蛇进行全向移动，使它更像现实生活中的蛇。与此同时，游戏中的道具也不再只有加血和去世两种苹果，添加了两种新道具：金苹果和遥控器。

游戏中共有四种道具：苹果，毒药，金苹果和遥控器。

吞食苹果后蛇身长度+1，并且增加1分数。

吞食金苹果后蛇的生命值+1.，并且增加5分数。

吞食遥控器后清除全图毒药，并且增加10分数。

吞食毒药后扣除1生命值。

游戏开始时蛇有1生命值。生命值扣光或蛇撞到边界则游戏结束

1.1.2操作方式

游戏开始后，通过键盘A/D键控制蛇转向。与物体碰撞时，蛇会自动吞食、

吞食不同物品有不同的效果。触及场地边界或吞食有毒食物和自身时，游戏结束。

1.2 引用模块列表

Tkinter

Configparser

Pygame

Websockets

第二章 系统分析

2.1 功能分析

2.1.1基本功能

1.蛇的全向移动

2.吞食物品

3.游戏结束判定

2.1.2拓展功能

1.额外物品

2.生命值功能

3.联机记录分数

4.地图选择

第三章 详细设计及实现

3.1地图选择界面

使用configparser库提供的功能读取./resource/下的所有文件并判断其是否为地图文件夹，若是则加入地图列表。然后用Tkinter 展示地图选择窗口。在Tkinter窗口内选择完地图开始游戏后进入game函数。

|  |
| --- |
| class foodSprite(pygame.sprite.Sprite):  def \_\_init\_\_(self):  pygame.sprite.Sprite.\_\_init\_\_(self)  self.image = pygame.image.load(imgpath+'food.png')  self.rect = self.image.get\_rect()  self.type = 0 |

3.2 物品类

物品类继承了pygame的精灵类，自定义了type与image属性，初始化pygame.sprite.Sprite.rect对象。

创建对象后，我将同类对象加入到对应的pygame.SpriteGroup内，便于后续操作。

|  |
| --- |
| class foodSprite(pygame.sprite.Sprite):  def \_\_init\_\_(self):  pygame.sprite.Sprite.\_\_init\_\_(self)  self.image = pygame.image.load(imgpath+'food.png')  self.rect = self.image.get\_rect()  self.type = 0 |

3.3 分数上传/下载

游戏结束时，基于websockets模块短连接，轮询，从云端获取最高分纪录并和本地比较。如果本地刷新了游戏记录则上传。此处使用我租借的云服务器部署记录最高分服务。

|  |
| --- |
| async def getglobalmaxscore(websocket):  global score,globalscore  await websocket.send(str(score))  globalscore = await websocket.recv() async def refreshglobalmaxscore(websocket):  global score  await websocket.send(str(score))  async def main\_logic():  async with websockets.connect('ws://47.101.138.238:26535') as websocket:  await getglobalmaxscore(websocket)  await refreshglobalmaxscore(websocket) asyncio.get\_event\_loop().run\_until\_complete(main\_logic()) if int(score)<=int(globalscore):  messagebox.showinfo('Game Over','Your score is:'+str(score)+'\nGlobal highest score is:'+str(globalscore)) else:  messagebox.showinfo('Game Over','Your score is:'+str(score)+'\nFormer lobal highest score is:'+str(globalscore)+'.Congratulations!') |

3.4 自定义地图

 游戏有自定义地图功能。在./resource目录下新建一个文件夹，并且在文件夹中置入如下文件即可完成地图创建。

其中，config.ini的文件格式如图所示：

下列变量分别对应：场地长度，场地高度，蛇头初始横坐标，蛇头初始纵坐标，金苹果出现概率，遥控器出现概率，蛇头旋转加速度，蛇的移动速度。

|  |
| --- |
| [basic]  #注意，场地的大小应当是一种可以被正常显示的分辨率，例如1024x768，1366x768，1920x1080  field\_width=1366  field\_height=768  start\_x=0  start\_y=100  super\_food\_rate=15  helper\_rate=15  [player]  accr\_rotate=2  maxspeed=3 |

3.5 额外道具

额外道具基于3.2中提到的物品类。两种不存在于原版贪吃蛇的道具的type属性分别是1,2.

在蛇吞食道具时会检测它吞食的道具的type属性并进行不同的操作。

|  |
| --- |
| if attacker.type==1:#超级食物，加一次生命值  score+=5  health+=1 if attacker.type==2:#遥控器，引爆场上所有炸弹  score+=10  for i in poison\_group:  poison\_group.remove(i) |

3.6 蛇头与蛇身

 蛇头类和蛇身类分别为playerhead和playerbody。两者都有用于储存位置的rect子对象和angle属性（原理图如下），和用于储存图形的image属性。

蛇头还使用v，vsave0，vsave1三个属性用于实现移动逻辑。前文提到过蛇头用rect储存自身位置，但rect.centerx和rect.centery这两个表示自身坐标的变量是整型，如果直接对其进行操作，对于因为全向移动而产生浮点数向量位移的蛇会产生严重的bug。因此，vsave0和vsave1属性会将速度向量分解后的分向量加到自身上，并且将自身大于1的部分减去并加到rect上。这样一来，蛇头转向时就不会出现位置偏移。原理图如下：



蛇身类还有编号属性，用于储存其按距离蛇头远近排列的次序。每个游戏刻，将距离蛇头最远的蛇身移到当前蛇头的位置并将编号赋值1，再把蛇头移走，并把其他蛇身的编号+1.

代码如下：

|  |
| --- |
| class playerhead(pygame.sprite.Sprite):  def \_\_init\_\_(self):  pygame.sprite.Sprite.\_\_init\_\_(self)  self.originimage = pygame.image.load(imgpath+'head.png')  self.angle=0  self.v=[0,0]  self.accr\_rotate=0  self.maxspeed=0  self.vsave0=0  self.vsave1=0  self.image=pygame.transform.rotate(self.originimage,-math.degrees(self.angle))  self.rect = self.originimage.get\_rect()  def isInBorder(self):  #边界检测  if (self.rect.left >= 0 and self.rect.right <= field\_width) and (self.rect.top >= 0 and self.rect.bottom <= field\_height):  return True  else:  return False  def move(self):  #限制角度范围，防止游戏过久后溢出  if abs(math.degrees(self.angle))>=360:  self.angle-=(math.pi\*2)\*self.angle/abs(self.angle)  #边界检测。在边界内就执行运动代码  if self.isInBorder():  a=math.cos(self.angle)  b=math.sin(self.angle)  #旋转。1.0001是为了防止出现除0错误  self.angle+=math.radians(move\_lr)  #旋转车体  self.image=pygame.transform.rotate(self.originimage,-math.degrees(self.angle))  self.rect = self.image.get\_rect(center=self.rect.center)  #用累积法储存速度，速度超过1就把整数部分加到坐标上，然后扣掉自己的整数部分  self.v=[self.maxspeed\*a,self.maxspeed\*b]  self.vsave0+=self.v[0]  self.vsave1+=self.v[1]  if abs(self.vsave0)>=1:  self.rect.centerx+=int(self.vsave0)  self.vsave0-=int(self.vsave0)  if abs(self.vsave1)>=1:  self.rect.centery+=int(self.vsave1)  self.vsave1-=int(self.vsave1) class playerbody(pygame.sprite.Sprite):  def \_\_init\_\_(self):  pygame.sprite.Sprite.\_\_init\_\_(self)  self.number=0  self.originimage = pygame.image.load(imgpath+'body.png')  self.angle=0  self.rect = self.originimage.get\_rect()  self.image=pygame.transform.rotate(self.originimage,-math.degrees(self.angle))  def update(self):  if self.number==len(body\_group.sprites())-1:  self.angle=body\_group.sprites()[0].angle  self.image=pygame.transform.rotate(self.originimage,-math.degrees(self.angle))  self.rect = self.image.get\_rect(center=self.rect.center)  self.rect.move\_ip(body\_group.sprites()[0].rect.centerx-self.rect.centerx,body\_group.sprites()[0].rect.centery-self.rect.centery)  self.number=1  else:  self.number+=1 |

 

（a） （b）

图3-9 生命值及关卡显示。(a)生命值显示；(b)关卡显示

碰撞检测与性能优化

这是整个游戏的核心功能。因为蛇头、蛇身和道具全部继承了pygame.sprite.Sprite类，所以我们可以使用自带的碰撞检测函数进行碰撞检测。

值得一提的是，在早期版本的游戏中，即使是在搭载了R7-4800H这样的强劲处理器的平台上运行，也会异常卡顿。这是因为最初版本的游戏，每tick蛇头移动时，所有的蛇身对象都会将自己移动到前一个蛇身的位置，进行大量的运算；而且，每tick进行碰撞检测时会检测场地上所有的物品是否与蛇头像素重叠。这样的性能开销是不可接受的，因此蛇身移动的算法修改成了前文所述的代码。此外，我计划用分区碰撞检测技术进行性能优化，但出于技术原因没有实现。

经过优化，现版本的游戏，资源占用虽然仍较高，但已不会妨碍游戏进行。

第四章 测试

4.1 地图选择界面测试

无问题，运行良好

4.2 移动测试

111

111

4.3 物体吞食测试

吞食四种道具均得到了应有的效果。







4.4 游戏结束测试

触及边界或者生命值归零时，游戏正常结束。

游戏结束后，从服务器获取全球最高分无异常。由于之前测试人员玩的分数太高，我打不过，此处的刷新记录截图比较旧。



* 综上，游戏可以正常运行，玩家游戏体验良好。