《推箱子》游戏Python设计实验报告

1. 选题

【游戏题目】

推箱子

【游戏软件功能】

推箱子游戏中玩家通过控制键盘的上下左右键即通过与键盘操作界面的交互控制人物和箱子的移动。当人物靠近箱子并朝箱子位置移动的时候，箱子在其移动方向没有被围墙阻碍的情况下可以被移动。玩家可以通过空格键控制游戏结束。

玩家通关即将游戏中的箱子都送到规定的箱子目的地，玩家通关游戏结束。

玩家没有通关即为箱子推错位置，游戏中的箱子没有到达目的地，却无法再进行移动，箱子的两个方向（左方和上方，左方和下方，右方和上方，右方和下方）都是围墙或者箱子。

1. 方案

【设置Pushbox类】

Pushbox类包含了游戏实现的重要数据和函数，包括对于Pushbox类中的重要数据的初始化，实现通过键盘来控制人物移动，进而控制箱子移动，在移动过程中响应键盘操作进行屏幕显示图像的绘制，Pushbox是游戏构成的基础，通过Pushbox完成了游戏各个模块功能的整合实现。

Self.level

Self.w

Self.h

Self.man

Self.hint

Self.sbox

Self.destinations

Self.todes

初始化数据成员函数

移动操作实现函数

判断是否过关函数

画图函数

向上移动图像绘制函数

向下移动图像绘制函数

向左移动图像绘制函数

向右移动图像绘制函数

【地图迷宫的初始化】

通过字符列表对迷宫地图进行初始化，列表中字符的个数对应Pushbox类中游戏显示界面的宽度和高度对的乘积（self.w\*self.h）

墙

人物：‘@’

箱子：‘$’

地板空地：‘-’

目的地：‘。’

A：‘&’

B：‘#’

游戏迷宫

self.level =

list('---#&&&#####-------

---##.---###-------

---##$---###-------

--&&&----###&&-----

-##--$-$---#&&#----

&&&-&-&&-$-&&####&&

&---#-##-##&##--..#

#-$--$----------..&

&&&&&-###-#@##---.#

----#-----#&#&&&###

----##&&&&&#-------'）

1. 关键技术

【pygame函数库的使用】

1.通过pygame函数库的使用简化代码

#插入pygame函数库

import pygame,

from pygame.locals import \*

#通过pygame初始化游戏

 pygame.init()

#通过pygame初始化游戏音乐播放

 pygame.mixer.init()

#设置游戏界面对话窗口大小

screen = pygame.display.set\_mode((1068,620))

 # 加载图片内容

  skin = pygame.image.load(skinfilename)

 background=pygame.image.load("2.jpg")

  gameover=pygame.image.load("3.jpg")

#加载音乐并设置音乐相关参数

pygame.mixer.music.load('1.mp3')

  laugh=pygame.mixer.Sound("2.wav")

   pygame.mixer.music.play(-1, 9)

  pygame.mixer.music.set\_volume(0.25)

 laugh.set\_volume(0.35)

  screen.fill(skin.get\_at((0,0)))

#设置游戏显示界面窗口标题

  pygame.display.set\_caption('Minions love bananas.py')

 #创建游戏时钟

   clock = pygame.time.Clock()

  pygame.key.set\_repeat(200,50)

#窗口更新

  pygame.display.update()

【人物箱子移动】

1. 移动操作实现函数函数

在移动过程中根据判断人所在位置的前方（移动行进方向）(记为A)，人所在位置前方物体的前方(记为B)，实现人物的各种移动操作。

A为墙‘#’：人物‘@’不进行移动，即对于用列表表示的地图没有变化。

A为空地‘-’：人物‘@’进行移动，人物当前所处列表存储字符变为‘-’，A所处列表存储字符变为‘@’

A为箱子‘$’

B为空地‘-’：人物所处列表存储字符变为‘-’，A所处列表存储字符变为‘@’,B所处列表存储字符变为’$’

B为箱子或墙：不能移动，没有改变。

 #控制人物移动

def \_move(self, d):

  self.sbox = 0

  h = to\_offset(d, self.w)

        #画图像素尺寸高度有关

  h2 = 2 \* h

      if self.level[self.man + h] == '-' or self.level[self.man + h] == '.':

#人的位置前面是地板或者目的地

        # 进行移动

#人前面的位置变成人

            to\_man(self.level, self.man+h)

#人的位置变成地板

            to\_floor(self.level, self.man)

            self.man += h

    elif self.level[self.man + h] == '\*' or self.level[self.man + h] == '$':

#人的前面

        if self.level[self.man + h2] == '-' or self.level[self.man + h2] == '.':

#人前面的物体的前面

            #箱子离开目的地

              if self.level[self.man + h]=='\*':

               if self.level[self.man+h2]=='.'or self.level[self.man+h2]=='-':

                      self.todes=self.todes-1#抵达目的地的箱子数目减少

                      print(self.todes)

#人前面的物体的前面变成箱子

to\_box(self.level, self.man + h2)

#人的前面变化成人

            to\_man(self.level, self.man + h)

#人所在位置变成地

        to\_floor(self.level, self.man)#人所在位置变成地板

                #箱子抵达目的地

                if self.level[self.man+h2]=='\*':

                    self.todes=self.todes+1#抵达目的地的箱子数目增加

                    print(self.todes)

#人的位置增加

                self.man += h

【判断通关】

游戏通关即抵达目的地的箱子数目等于目的地数目，在Pushbox类中有判断通关（juge）函数，其功能主要与移动操作（\_move）函数结合实现，在move函数中，当人物所在位置前方列表存储字符为'$'（此位置记为A），人物所在位置前方A的前方(记为B)列表存储字符为'\*'（箱子到达目的地的字符表示符号）,到达目的地的箱子数目增加1,(此处存在特殊情况，即当箱子离开目的地再被推到目的地情况的处理:当箱子离开目的地，抵达目的地的箱子数目减1)

 #判断游戏是否胜利

    def juge(self):

         if self.todes==self.destinations:

            return True

【地图迷宫的初始化】

通过字符列表对迷宫地图进行初始化，列表中字符的个数对应Pushbox类中游戏显示界面的宽度和高度（self.w\*self.h）

1. 结果和效果

关卡



空格控制游戏结束



游戏通关



游戏要求玩家通过控制上下左右键，使得屏幕中的六个箱子全部抵达目的地，游戏完成了关卡的设计，当玩家将箱子全部移到目的地，会显示游戏结束界面，提醒玩家游戏结束，同时当玩家不想进行游戏或者认为不能通关时，可以通过空格退出游戏。

1. 总结和不足

Python是之前自己一直没有接触的编程语言，最大的难处就是入手，在Python游戏的设计中花了4天的时间了解Python的一些重要基本语法，学习使用Python的pygame函数库，学习配置Python编程环境，在做完以上内容后就要开始着手写自己的《推箱子》游戏，由于已经用两周时间完成了C语言的《推箱子》游戏的编写，在用Python编写时，思维的转化，算法的转化是一个难点，要摒弃之前用C语言写《推箱子》的算法思想，设计出新的应用于Python的算法。

游戏完成了最初的设想，却有很多待改进的地方，比如需要设置超过一关的更多关卡，需要设计更多的有趣的游戏功能，画面的细节同样需要更优化的处理，在游戏处于不同状态时音效的丰富化处理……自己需要在后续不断的学习中优化自己的游戏，进步空间还很大。