期末游戏项目报告

——翻牌记忆力训练游戏的探索与实现

项目开发人员

1、黄宇辰（学号：10185102253 张老师班级）

2、马应鹏（学号：10185102258 张老师班级）

引言摘要

1. 项目背景 ： 计算机导论期末项目
2. 项目基础 ： 基于python语言 、pygame库的游戏代码
3. 制作者的一段话：

作为python的初学者。哦，不，是编程的初学者，期末游戏项目的完成让我们由衷地感到骄傲与自豪，尽管我们的游戏并不是那么的完美，而且也有许许多多的新奇想法没来的及去实现。这款翻牌记忆训练小游戏，是我们倾注心血，一步一步完善出来的，每一个素材都经过我们自己的搜素雕饰，代码的细节部分，也让我们在完成项目时，费了不小的力气。但终归，我们完成了这个游戏！

1. 游戏项目介绍

游戏名称： 翻牌记忆力训练游戏（益智类）

游戏简介： 游戏界面中会出现4\*5的20张卡牌，20张卡牌包涵10对不同图案的卡牌（每 种卡 牌有 两张）。当每局游戏开始时候，每张卡牌会有短暂的翻面时间（每组0.7 秒），让大家记忆卡牌。随后玩家可以通过鼠标点击，将卡牌翻面，若卡牌图案相同， 则卡牌保持翻面状态，若卡牌图案不同，则卡盘变回盖伏状态。当玩家将所有卡牌翻 面时，玩家胜利。

游戏操作： 鼠标点击翻牌，按ESC键可以退出（或者点击右上角的‘×’图标）

游戏彩蛋： ①游戏与黄宇辰同学（制作人之一）喜爱的游戏进行曲《Hello》同时进行。

②当玩家连续点开两组、三组、四组、五组牌时，会分别听到愈加激动人心的双杀、 三杀、四杀、五杀的音效。

③同样，当玩家完成翻面的组数越来越多时，会出现更多的不同音效。

④当每局游戏开始时，会听到我们两位制作人认为最骚气的伊泽瑞尔游戏音效。

⑤当进行游戏下一局时，会听到卡牌大师崔斯特(卡牌的音效当然适合这个游戏)的游 戏音效“洗牌”。

⑥当玩家连续多次完成不了翻面的时候，会听到菲奥拉(游戏制作人黄宇辰曾经玩过 数百场的英雄)的嘲讽音效。

⑦当游戏结束时，游戏界面会恢复为彩色（此处致敬电影《后来的我们》当中男主见 清创作的游戏细节—当失去最爱的人时，游戏界面变为灰色）。

1. 基本架构

在主函数当中，如基于pygame库的其他游戏一样，总是会有设置界面、初始化库等基本操作，所以对此我们不进行过多赘述。我们对于游戏中大部分的音效部分都是在主函数当中进行完成，在完成音效部分时，我们设置了三个变量(已经翻牌的数目、连续翻牌的数目、翻牌的次数)以此去触发每一次音效。我们同样为游戏中的鼠标点击、翻牌失败等事件嵌套了音效。主函数当中，我们对于每一次翻牌事件都会确定此次翻牌是第一次翻牌还是第二次翻牌，并记录下牌面图案，以此来判断是否完成翻面。同时在每次翻面之后，判断本局游戏是否完成从而进入结束动画和下一局游戏。

我们的游戏主函数如下：

def main():  
 #在后面其他函数中也需要用到这两个变量，所以先标记全局  
 global FPSCLOCK, Background\_背景  
 #初始化pygame的库  
 pygame.init()  
 FPSCLOCK = pygame.time.Clock()  
 #设定游戏界面的大小  
 Background\_背景 = pygame.display.set\_mode((Width, Height),0,32)  
 #额外加的背景  
 Background\_背景图片 = pygame.image.load('Back2.jpg').convert()  
 Background\_背景.blit(Background\_背景图片, (0, 0))  
 ################################  
 pygame.mixer.music.load('背景音乐.ogg') # 背景音乐  
 pygame.mixer.music.set\_volume(0.05) # 设置音量  
 pygame.mixer.music.play(-1) # 播放音乐  
 sound\_begin = pygame.mixer.Sound('是时候表演真正的技术了.wav') # 播放开始！  
 sound\_begin.set\_volume(0.7)  
 sound\_begin.play(0, 0)  
 ################################  
 #先对鼠标的坐标位置进行一次赋值存储，以便后面对于鼠标的位置进行存储操作  
 mouse\_x\_position = 0  
 mouse\_y\_position = 0  
 #这里为我们的游戏界面去设置一个窗口标题 — “记忆训练小游戏”  
 pygame.display.set\_caption('宇宙最牛逼的记忆训练小游戏，彩蛋满满哦')  
 #这里是两个数据结构的函数，记下标记############################################################################  
 mainBoard = getRandomizedBoard()  
 revealedCards = generateRevealedCardsData(False)  
  
 firstSelection = None # 记录每次翻牌的两次记录中，第一次的翻牌坐标  
 Background\_背景.blit(Background\_背景图片,(0,0)) #这里重新刷新屏幕的背景  
 startGameAnimation(mainBoard) #开始的一个动画操作，给玩家一个残存的记忆印象，这个函数将在后面写出，记下标记#######################################  
 #########################################以下是计数器  
 n\_open\_num = 0 #已经翻多少次牌的计数  
 n\_dacheng\_num = 0 #已经翻出多少张牌  
 n\_lianxudachen = 0 #记录连续达成  
  
 while True:  
 # 游戏主要程序的部分  
 mouseClicked = False #光标没有按下  
 Background\_背景.blit(Background\_背景图片,(0,0)) # 每次循环开始，先重新绘制游戏背景，去掩盖之前的画面  
 drawBoard(mainBoard, revealedCards) #同样调用后面的函数，记下标记################################################  
 for event in pygame.event.get(): # 捕捉游戏进程当中的每次事件  
 #这是一个关闭游戏的接口，当按下ESC键的时候或者点击右上角的插口的时候，这个游戏程序都会关闭  
 if event.type == QUIT or (event.type == KEYUP and event.key == K\_ESCAPE):  
 pygame.quit()  
 sys.exit()  
 #当光标移动时，记录光标移动的坐标  
 elif event.type == MOUSEMOTION:  
 mouse\_x\_position, mouse\_y\_position = event.pos  
 #当光标按下时，记录光标按下时的坐标  
 elif event.type == MOUSEBUTTONUP:  
 mouse\_x\_position, mouse\_y\_position = event.pos  
 mouseClicked = True #鼠标按下  
 # 这里调用了一个后来我们写的关于获取光标按下时所在卡牌的位置信息，记下标记###########################################################  
 Card\_x\_position, Card\_y\_position = getCardAtPixel(mouse\_x\_position, mouse\_y\_position)  
 if Card\_x\_position != None and Card\_y\_position != None:  
 # 这种情况下，光标在某张卡牌上面  
 if not revealedCards[Card\_x\_position][Card\_y\_position]: #如果光标在未翻过的牌面上，画出蓝框  
 drawHighlightCard(Card\_x\_position, Card\_y\_position)  
 if not revealedCards[Card\_x\_position][Card\_y\_position] and mouseClicked: #光标出现在未翻过的卡牌上，并且发生了翻牌动作  
 revealCardsAnimation(mainBoard, [(Card\_x\_position, Card\_y\_position)]) #进行绘出牌面图案的动作  
 revealedCards[Card\_x\_position][Card\_y\_position] = True #这个位置的牌变为翻开状态  
 if firstSelection == None: # 如果这是第一次的翻牌动作  
 firstSelection = (Card\_x\_position, Card\_y\_position) #记录第一次卡牌翻面时候的位置  
 #########################翻开卡牌声音  
 sound\_open = pygame.mixer.Sound('开牌声音1.ogg') # 播放开牌！  
 sound\_open.set\_volume(1)  
 sound\_open.play(0, 0)  
 pygame.time.wait(100)  
 else: # 第一次的翻牌位置已经被储存，进行第二次的翻牌  
 # 下面检查两次翻牌的牌面颜色图案是否一样  
 icon1shape, icon1color = getShapeAndColor(mainBoard, firstSelection[0], firstSelection[1]) #得到第一次翻的卡牌的颜色与形状  
 icon2shape, icon2color = getShapeAndColor(mainBoard, Card\_x\_position, Card\_y\_position) #得到第二次翻的卡牌的颜色与形状  
 if icon1shape != icon2shape or icon1color != icon2color:  
 n\_open\_num += 1 ###############################################  
  
 ###############记数 当超过双杀时终结  
 if n\_lianxudachen >= 2:  
 sound\_shutdown = pygame.mixer.Sound('终结.ogg') # 播放终结！  
 sound\_shutdown.set\_volume(0.5)  
 sound\_shutdown.play(0, 0)  
 if n\_open\_num %10 == 0:  
 sound\_tooslow = pygame.mixer.Sound('动作太慢了.ogg') # 播放终结！  
 sound\_tooslow.set\_volume(0.5)  
 sound\_tooslow.play(0, 0)  
 else:  
 pygame.time.wait(100)  
 sound\_shibai = pygame.mixer.Sound('翻牌失败.ogg') # 播放终结！  
 sound\_shibai.set\_volume(0.5)  
 sound\_shibai.play(0, 0)  
 n\_lianxudachen = 0 #####################终结连杀  
 # 如果两者卡牌的颜色和形状中有一个不一样  
 pygame.time.wait(1000) # 这里暂定1000微秒，即暂停一秒钟  
 coverCardsAnimation(mainBoard, [(firstSelection[0], firstSelection[1]), (Card\_x\_position, Card\_y\_position)]) #触发重新覆盖上这个牌面的动画  
 revealedCards[firstSelection[0]][firstSelection[1]] = False #这张牌重新变成没有翻面的状态  
 revealedCards[Card\_x\_position][Card\_y\_position] = False #第二张牌重新变成没有翻页的状态  
 elif hasWon(revealedCards): # 这里设置玩家胜利条件，当所有牌面都翻过来时候，玩家胜利  
 n\_open\_num += 1  
 n\_lianxudachen += 1  
 n\_dacheng\_num += 1  
 if n\_lianxudachen == 2:  
 sound\_2kill = pygame.mixer.Sound('双杀.ogg') # 播放双杀！  
 sound\_2kill.set\_volume(0.5)  
 sound\_2kill.play(0, 0)  
 if n\_lianxudachen == 3:  
 sound\_3kill = pygame.mixer.Sound('三杀.ogg') # 播放三杀！  
 sound\_3kill.set\_volume(0.5)  
 sound\_3kill.play(0, 0)  
 if n\_lianxudachen == 4:  
 sound\_4kill = pygame.mixer.Sound('四杀.ogg') # 播放四杀！  
 sound\_4kill.set\_volume(0.5)  
 sound\_4kill.play(0, 0)  
 if n\_lianxudachen == 5:  
 sound\_5kill = pygame.mixer.Sound('五杀.ogg') # 播放五杀！  
 sound\_5kill.set\_volume(0.5)  
 sound\_5kill.play(0, 0)  
 #为了处理最后一次翻牌音效，进行的功能性暂停  
 pygame.time.wait(1500)  
 ############################游戏结束更新num  
 n\_open\_num = 0  
 n\_dacheng\_num = -1  
 n\_lianxudachen = 0  
 gameWonAnimation(mainBoard) #进行游戏胜利动画  
 pygame.time.wait(2000) #等待2秒钟的时间  
 # 重新设置游戏版面  
 mainBoard = getRandomizedBoard() #随机重排牌面  
 revealedCards = generateRevealedCardsData(False) #将牌面全部变为反面  
 ##################可以在此处重新设置卡牌增多动作，但需要先了解下面数据结构的轮子，记下标记######################################################  
 # 这里将界面重新绘制，等待1秒钟时间  
 drawBoard(mainBoard, revealedCards)  
 pygame.display.update()  
 pygame.time.wait(1000)  
 # 重新开始游戏开局动画  
 startGameAnimation(mainBoard)  
 ###################################################################记数  
 if icon1shape == icon2shape and icon1color == icon2color:  
 n\_open\_num += 1  
 n\_lianxudachen += 1  
 n\_dacheng\_num += 1  
 if n\_lianxudachen < 2:  
 if n\_dacheng\_num == 1:  
 sound\_firstblood = pygame.mixer.Sound('一血.ogg') # 播放一血！  
 sound\_firstblood.set\_volume(0.5)  
 sound\_firstblood.play(0, 0)  
 if n\_dacheng\_num == 2:  
 sound\_3card = pygame.mixer.Sound('翻出三组牌.ogg') # 播放3牌！  
 sound\_3card.set\_volume(0.5)  
 sound\_3card.play(0, 0)  
 if n\_dacheng\_num == 3:  
 sound\_4card = pygame.mixer.Sound('翻出四组牌.ogg') # 播放4血！  
 sound\_4card.set\_volume(0.5)  
 sound\_4card.play(0, 0)  
 if n\_dacheng\_num == 4:  
 sound\_5card = pygame.mixer.Sound('翻出五组牌.ogg') # 播放5血！  
 sound\_5card.set\_volume(0.5)  
 sound\_5card.play(0, 0)  
 if n\_dacheng\_num == 5:  
 sound\_6card = pygame.mixer.Sound('翻出六组牌.ogg') # 播放6血！  
 sound\_6card.set\_volume(0.5)  
 sound\_6card.play(0, 0)  
 if n\_dacheng\_num == 6 :  
 sound\_7card = pygame.mixer.Sound('翻出七组牌.ogg') # 播放7血！  
 sound\_7card.set\_volume(0.5)  
 sound\_7card.play(0, 0)  
 if n\_dacheng\_num >= 7:  
 sound\_8card = pygame.mixer.Sound('超神.ogg') # 播放8血！  
 sound\_8card.set\_volume(0.5)  
 sound\_8card.play(0, 0)  
 n\_lianxudachen = 1  
 if n\_lianxudachen >= 2:  
 if n\_lianxudachen ==2:  
 sound\_2kill = pygame.mixer.Sound('双杀.ogg') # 播放双杀！  
 sound\_2kill.set\_volume(0.5)  
 sound\_2kill.play(0, 0)  
 if n\_lianxudachen ==3:  
 sound\_3kill = pygame.mixer.Sound('三杀.ogg') # 播放三杀！  
 sound\_3kill.set\_volume(0.5)  
 sound\_3kill.play(0, 0)  
 if n\_lianxudachen ==4:  
 sound\_4kill = pygame.mixer.Sound('四杀.ogg') # 播放四杀！  
 sound\_4kill.set\_volume(0.5)  
 sound\_4kill.play(0, 0)  
 if n\_lianxudachen == 5:  
 sound\_5kill = pygame.mixer.Sound('五杀.ogg') # 播放五杀！  
 sound\_5kill.set\_volume(0.5)  
 sound\_5kill.play(0, 0)  
 firstSelection = None # 这时候重新设置第一次翻牌的位置，为没有翻牌状态  
 # 时刻更新界面  
 pygame.display.update()  
 FPSCLOCK.tick(FPS)

1. 核心算法技术
2. 像素坐标与方块坐标的转换的算法

这个算法将像素坐标与方块坐标之间进行转换，从而更好的去捕获我们的鼠标翻牌位置。

def Actural\_position(Card\_x\_position, Card\_y\_position):  
 # 这个轮子将方块坐标转化为像素坐标，供游戏本体识别  
 left = Card\_x\_position \* (Card\_size + Distance\_between\_card) + X\_margin\_横轴边缘像素  
 top = Card\_y\_position \* (Card\_size + Distance\_between\_card) + Y\_margin\_纵轴边缘像素  
 return (left, top)  
def getCardAtPixel(x, y): #这是一个统一像素坐标和方块坐标的轮子，将像素坐标转化为方块坐标  
 for Card\_x\_position in range(wide\_num):  
 for Card\_y\_position in range(high\_num):  
 left, top = Actural\_position(Card\_x\_position, Card\_y\_position) #这里获得光标下方块坐标的像素坐标值  
 CardRect = pygame.Rect(left, top, Card\_size, Card\_size) #这里获得每个牌的像素坐标  
 if CardRect.collidepoint(x, y): #如果光标下的像素坐标在某个牌下的像素坐标内部  
 return (Card\_x\_position, Card\_y\_position) #则返回那张牌的方块坐标  
 return (None, None) #若光标不在某个牌的像素坐标，那么返回，光标无选中牌面

1. 绘制牌面的算法

在这个算法当中，我们会从之前设置过的牌面形状元组和牌面颜色元组当中随机绘制出牌面。

def drawIcon(shape, color, Card\_x\_position, Card\_y\_position): #这是绘制牌面图案的轮子  
 quarter = int(Card\_size \* 0.25) # syntactic sugar  
 half = int(Card\_size \* 0.5) # syntactic sugar  
  
 left, top = Actural\_position(Card\_x\_position, Card\_y\_position) # 这里获得像素坐标  
 # 这里绘制每种牌面  
 if shape == DONUT:  
 pygame.draw.circle(Background\_背景, color, (left + half, top + half), half - 5)  
 pygame.draw.circle(Background\_背景, BGCOLOR, (left + half, top + half), quarter - 5)  
 elif shape == SQUARE:  
 pygame.draw.rect(Background\_背景, color, (left + quarter, top + quarter, Card\_size - half, Card\_size - half))  
 elif shape == DIAMOND:  
 pygame.draw.polygon(Background\_背景, color, ((left + half, top), (left + Card\_size - 1, top + half), (left + half, top + Card\_size - 1), (left, top + half)))  
 elif shape == LINES:  
 for i in range(0, Card\_size, 4):  
 pygame.draw.line(Background\_背景, color, (left, top + i), (left + i, top))  
 pygame.draw.line(Background\_背景, color, (left + i, top + Card\_size - 1), (left + Card\_size - 1, top + i))  
 elif shape == OVAL:  
 pygame.draw.ellipse(Background\_背景, color, (left, top + quarter, Card\_size, half))

1. 结束动画的算法

如果说作为制作人之一去选择最终要的算法，那么结束动画必不可少，虽然它不像上面两个算法那样处于核心地位，但却实现了游戏中三分之一左右的部分。我们使用了一个for循环去循环出三个不同的连续动画。

def gameWonAnimation(board):  
 # 游戏结束动画的轮子######################################  
 coveRed\_红Cards = generateRevealedCardsData(True)  
 color1 = LIGHTBGCOLOR  
 color2 = BGCOLOR  
 Card\_cover\_卡牌背面图片 = pygame.image.load('卡牌背面.jpg').convert() # 记下标记##############################################  
  
 for i in range(0,3):  
 if i == 0:  
 Win\_pic = pygame.image.load('胜利图1.jpg').convert()  
 Background\_背景.blit(Win\_pic,(0,0))  
 drawBoard(board, coveRed\_红Cards)  
 sound\_ACE = pygame.mixer.Sound('团灭.ogg') #播放团灭！  
 sound\_ACE.set\_volume(0.5)  
 sound\_ACE.play(0, 0)  
 pygame.display.update()  
 pygame.time.wait(2000)  
 if i ==1 :  
 Win\_pic\_2 = pygame.image.load('胜利图2.jpg').convert()  
 Background\_背景.blit(Win\_pic\_2, (0, 0))  
 drawBoard(board, coveRed\_红Cards)  
 pygame.display.update()  
 sound\_xipai = pygame.mixer.Sound('洗牌.ogg') # 播放开始！  
 sound\_xipai.set\_volume(0.7)  
 sound\_xipai.play(0, 0)  
 pygame.time.wait(2000)  
 if i ==2 :  
 Win\_pic\_3 = pygame.image.load('胜利图3.jpg').convert()  
 Background\_背景.blit(Win\_pic\_3, (0, 0))  
 drawBoard(board, coveRed\_红Cards)  
 sound\_begin = pygame.mixer.Sound('是时候表演真正的技术了.wav') # 播放开始！  
 sound\_begin.set\_volume(0.7)  
 sound\_begin.play(0, 0)

1. 游戏制作历程

任务分工：

黄宇辰同学：代码实现、游戏创意、报告撰写、素材搜集

马应鹏同学：报告撰写、游戏创意、素材搜集

历程详谈：

坦白说，这款游戏并不是我们一开始就确定的游戏项目(原定项目是五子棋，由于过多人写五子棋的同类游戏，所以我们放弃了)，看完所有人提交的报告后，我们决定完成这款本没有同类项目的游戏。在项目制作当中，我们两个制作人的主要分工见上文。当我们的项目基本完成并递交了1.0版本后，有另外小组的同学(最终项目可能与我们相似)向我们询问了我们游戏的细节，在了解项目类型相似之后，我们果断进行游戏升级，将我们的游戏丰富了节奏性，更添加了更多的彩蛋。我们完善游戏之后，我们二人小组与她们交流了双方的游戏效果(那时对方可能并未完成游戏全部制作)，我们对于自己项目的游戏性与酷炫效果感到稍稍满意，并思考是否能够实现按钮的开始与结束，同时想尝试完成随着游戏进程的推进游戏难度增加等等，但毕竟时间有限，考试周也随之而来，我们没有如期丰富更多游戏内容，这也算是我们此次游戏项目的一份遗憾。

1. 不足处与将来可能工作
2. 可拓展性：主要从用户交互功能上去完善这款游戏：

①可以实现由玩家输入确定卡牌数量

②实现游戏难度的逐级递增

③写出游戏按钮，进行游戏开始与结束，让游戏界面不再那么突兀

④完成得分榜，游戏内置计时功能，在每一局完成之后，由用户输入玩家姓名， 记录每种难度下游戏完成的时间。

1. 部分缺陷：游戏封装后，程序会由高达230MB左右的硬盘占用，用户交互方面并没有做的足够 好。

附录：

参考文献与链接：

1. <https://blog.csdn.net/lqzdreamer/article/details/77917493?tdsourcetag=s_pctim_aiomsg>
2. 《**Python和Pygame游戏开发指南》 ————————————** [美] Al Sweigart 斯维加特

游戏操作说明：

鼠标左键点击翻牌，牌面一样翻牌成功，直到所有卡牌翻面，游戏胜利！

游戏源代码：

#第一步当然是导入这个库  
import random, pygame, sys  
from pygame.locals import \*  
  
FPS = 30 # 设置刷新的帧数  
Width = 640 # 这里设置界面宽度  
Height = 480 # 这里设置界面高度  
Speed\_of\_open = 30 # 在这里我们设置了翻牌的速度  
Card\_size = 50 # 单个牌的高与宽  
Distance\_between\_card = 10 # 每个牌之间的距离  
wide\_num = 5 # 这里我设置了横着有多少个牌  
high\_num = 4 # 竖着有多少个牌  
  
  
#这里我们进行异常处理，要保证我们牌的数量是偶数哦  
assert (wide\_num \* high\_num) % 2 == 0, '卡牌数量需要是偶数，此情况下，卡牌数量是单数，无法进行游戏'  
  
#这里我们计算一下边缘部分的像素，后面有用  
X\_margin\_横轴边缘像素 = int((Width - (wide\_num \* (Card\_size + Distance\_between\_card))) / 2)  
Y\_margin\_纵轴边缘像素 = int((Height - (high\_num \* (Card\_size + Distance\_between\_card))) / 2)  
  
#设置颜色 R G B  
Gray\_灰 = (100, 100, 100)  
NavyBlue\_海军蓝 = ( 60, 60, 100)  
White\_白 = (255, 255, 255)  
Red\_红 = (230, 0, 0)  
Green\_绿 = ( 0, 255, 0)  
Blue\_深蓝 = ( 0, 0, 255)  
Yellow\_黄 = (255, 255, 0)  
Orage\_橙 = (255, 128, 0)  
Purple\_粉 = (255, 0, 255)  
Gyan\_青 = ( 0, 255, 255)  
  
#这里为各种颜色进行变量名的美化  
BGCOLOR = NavyBlue\_海军蓝  
LIGHTBGCOLOR = Gray\_灰  
CardCOLOR = White\_白  
HIGHLIGHTCOLOR = Blue\_深蓝  
  
#这里设置了一些图形的样式的名称，提高代码的可读性  
DONUT = 'donut'  
SQUARE = 'square'  
DIAMOND = 'diamond'  
LINES = 'lines'  
OVAL = 'oval'  
  
#对图形样式和颜色设置元组（不再更改），以便后面进行随机取样  
ALLCOLORS = (Red\_红, Green\_绿, Blue\_深蓝, Yellow\_黄, Orage\_橙, Purple\_粉, Gyan\_青)  
ALLSHAPES = (DONUT, SQUARE, DIAMOND, LINES, OVAL)  
  
  
#处理 同种图形超过 2 个的异常情况  
assert len(ALLCOLORS) \* len(ALLSHAPES) \* 2 >= wide\_num \* high\_num, "图形组合数量不够，牌面个数过多了，应该调整卡牌数量"  
  
#下面是游戏的主程序开始  
  
def main():  
 #在后面其他函数中也需要用到这两个变量，所以先标记全局  
 global FPSCLOCK, Background\_背景  
  
  
 #初始化pygame的库  
 pygame.init()  
 FPSCLOCK = pygame.time.Clock()  
 #设定游戏界面的大小  
 Background\_背景 = pygame.display.set\_mode((Width, Height),0,32)  
  
 #额外加的背景  
 Background\_背景图片 = pygame.image.load('Back2.jpg').convert()  
 Background\_背景.blit(Background\_背景图片, (0, 0))  
 ################################  
 pygame.mixer.music.load('背景音乐.ogg') # 背景音乐  
 pygame.mixer.music.set\_volume(0.05) # 设置音量  
 pygame.mixer.music.play(-1) # 播放音乐  
 sound\_begin = pygame.mixer.Sound('是时候表演真正的技术了.wav') # 播放开始！  
 sound\_begin.set\_volume(0.7)  
 sound\_begin.play(0, 0)  
 ################################  
  
 #先对鼠标的坐标位置进行一次赋值存储，以便后面对于鼠标的位置进行存储操作  
 mouse\_x\_position = 0  
 mouse\_y\_position = 0  
 #这里为我们的游戏界面去设置一个窗口标题 — “记忆训练小游戏”  
 pygame.display.set\_caption('宇宙最牛逼的记忆训练小游戏，彩蛋满满哦')  
  
  
 #这里是两个数据结构的函数，记下标记############################################################################  
 mainBoard = getRandomizedBoard()  
 revealedCards = generateRevealedCardsData(False)  
  
 firstSelection = None # 记录每次翻牌的两次记录中，第一次的翻牌坐标  
 Background\_背景.blit(Background\_背景图片,(0,0)) #这里重新刷新屏幕的背景  
 startGameAnimation(mainBoard) #开始的一个动画操作，给玩家一个残存的记忆印象，这个函数将在后面写出，记下标记#######################################  
  
 #########################################以下是计数器  
 n\_open\_num = 0 #已经翻多少次牌的计数  
 n\_dacheng\_num = 0 #已经翻出多少张牌  
 n\_lianxudachen = 0 #记录连续达成  
  
  
  
 while True:  
  
 # 游戏主要程序的部分  
 mouseClicked = False #光标没有按下  
  
 Background\_背景.blit(Background\_背景图片,(0,0)) # 每次循环开始，先重新绘制游戏背景，去掩盖之前的画面  
 drawBoard(mainBoard, revealedCards) #同样调用后面的函数，记下标记################################################  
  
 for event in pygame.event.get(): # 捕捉游戏进程当中的每次事件  
  
  
 #这是一个关闭游戏的接口，当按下ESC键的时候或者点击右上角的插口的时候，这个游戏程序都会关闭  
 if event.type == QUIT or (event.type == KEYUP and event.key == K\_ESCAPE):  
 pygame.quit()  
 sys.exit()  
 #当光标移动时，记录光标移动的坐标  
 elif event.type == MOUSEMOTION:  
 mouse\_x\_position, mouse\_y\_position = event.pos  
 #当光标按下时，记录光标按下时的坐标  
 elif event.type == MOUSEBUTTONUP:  
 mouse\_x\_position, mouse\_y\_position = event.pos  
 mouseClicked = True #鼠标按下  
  
  
  
  
 # 这里调用了一个后来我们写的关于获取光标按下时所在卡牌的位置信息，记下标记###########################################################  
 Card\_x\_position, Card\_y\_position = getCardAtPixel(mouse\_x\_position, mouse\_y\_position)  
 if Card\_x\_position != None and Card\_y\_position != None:  
 # 这种情况下，光标在某张卡牌上面  
 if not revealedCards[Card\_x\_position][Card\_y\_position]: #如果光标在未翻过的牌面上，画出蓝框  
 drawHighlightCard(Card\_x\_position, Card\_y\_position)  
 if not revealedCards[Card\_x\_position][Card\_y\_position] and mouseClicked: #光标出现在未翻过的卡牌上，并且发生了翻牌动作  
  
  
 revealCardsAnimation(mainBoard, [(Card\_x\_position, Card\_y\_position)]) #进行绘出牌面图案的动作  
 revealedCards[Card\_x\_position][Card\_y\_position] = True #这个位置的牌变为翻开状态  
 if firstSelection == None: # 如果这是第一次的翻牌动作  
 firstSelection = (Card\_x\_position, Card\_y\_position) #记录第一次卡牌翻面时候的位置  
 #########################翻开卡牌声音  
 sound\_open = pygame.mixer.Sound('开牌声音1.ogg') # 播放开牌！  
 sound\_open.set\_volume(1)  
 sound\_open.play(0, 0)  
  
 pygame.time.wait(100)  
  
  
 else: # 第一次的翻牌位置已经被储存，进行第二次的翻牌  
 # 下面检查两次翻牌的牌面颜色图案是否一样  
 icon1shape, icon1color = getShapeAndColor(mainBoard, firstSelection[0], firstSelection[1]) #得到第一次翻的卡牌的颜色与形状  
 icon2shape, icon2color = getShapeAndColor(mainBoard, Card\_x\_position, Card\_y\_position) #得到第二次翻的卡牌的颜色与形状  
  
 if icon1shape != icon2shape or icon1color != icon2color:  
 n\_open\_num += 1 ###############################################  
  
 ###############记数 当超过双杀时终结  
 if n\_lianxudachen >= 2:  
 sound\_shutdown = pygame.mixer.Sound('终结.ogg') # 播放终结！  
 sound\_shutdown.set\_volume(0.5)  
 sound\_shutdown.play(0, 0)  
 if n\_open\_num %10 == 0:  
 sound\_tooslow = pygame.mixer.Sound('动作太慢了.ogg') # 播放终结！  
 sound\_tooslow.set\_volume(0.5)  
 sound\_tooslow.play(0, 0)  
 else:  
 pygame.time.wait(100)  
 sound\_shibai = pygame.mixer.Sound('翻牌失败.ogg') # 播放终结！  
 sound\_shibai.set\_volume(0.5)  
 sound\_shibai.play(0, 0)  
  
 n\_lianxudachen = 0 #####################终结连杀  
  
 # 如果两者卡牌的颜色和形状中有一个不一样  
 pygame.time.wait(1000) # 这里暂定1000微秒，即暂停一秒钟  
 coverCardsAnimation(mainBoard, [(firstSelection[0], firstSelection[1]), (Card\_x\_position, Card\_y\_position)]) #触发重新覆盖上这个牌面的动画  
 revealedCards[firstSelection[0]][firstSelection[1]] = False #这张牌重新变成没有翻面的状态  
 revealedCards[Card\_x\_position][Card\_y\_position] = False #第二张牌重新变成没有翻页的状态  
  
  
 elif hasWon(revealedCards): # 这里设置玩家胜利条件，当所有牌面都翻过来时候，玩家胜利  
 n\_open\_num += 1  
 n\_lianxudachen += 1  
 n\_dacheng\_num += 1  
  
 if n\_lianxudachen == 2:  
 sound\_2kill = pygame.mixer.Sound('双杀.ogg') # 播放双杀！  
 sound\_2kill.set\_volume(0.5)  
 sound\_2kill.play(0, 0)  
 if n\_lianxudachen == 3:  
 sound\_3kill = pygame.mixer.Sound('三杀.ogg') # 播放三杀！  
 sound\_3kill.set\_volume(0.5)  
 sound\_3kill.play(0, 0)  
 if n\_lianxudachen == 4:  
 sound\_4kill = pygame.mixer.Sound('四杀.ogg') # 播放四杀！  
 sound\_4kill.set\_volume(0.5)  
 sound\_4kill.play(0, 0)  
 if n\_lianxudachen == 5:  
 sound\_5kill = pygame.mixer.Sound('五杀.ogg') # 播放五杀！  
 sound\_5kill.set\_volume(0.5)  
 sound\_5kill.play(0, 0)  
  
 #为了处理最后一次翻牌音效，进行的功能性暂停  
 pygame.time.wait(1500)  
 ############################游戏结束更新num  
  
 n\_open\_num = 0  
 n\_dacheng\_num = -1  
 n\_lianxudachen = 0  
  
 gameWonAnimation(mainBoard) #进行游戏胜利动画  
 pygame.time.wait(2000) #等待2秒钟的时间  
  
  
  
 # 重新设置游戏版面  
 mainBoard = getRandomizedBoard() #随机重排牌面  
 revealedCards = generateRevealedCardsData(False) #将牌面全部变为反面  
 ##################可以在此处重新设置卡牌增多动作，但需要先了解下面数据结构的轮子，记下标记######################################################  
  
 # 这里将界面重新绘制，等待1秒钟时间  
 drawBoard(mainBoard, revealedCards)  
 pygame.display.update()  
 pygame.time.wait(1000)  
  
 # 重新开始游戏开局动画  
 startGameAnimation(mainBoard)  
 ###################################################################记数  
 if icon1shape == icon2shape and icon1color == icon2color:  
 n\_open\_num += 1  
 n\_lianxudachen += 1  
 n\_dacheng\_num += 1  
 if n\_lianxudachen < 2:  
 if n\_dacheng\_num == 1:  
 sound\_firstblood = pygame.mixer.Sound('一血.ogg') # 播放一血！  
 sound\_firstblood.set\_volume(0.5)  
 sound\_firstblood.play(0, 0)  
 if n\_dacheng\_num == 2:  
 sound\_3card = pygame.mixer.Sound('翻出三组牌.ogg') # 播放3牌！  
 sound\_3card.set\_volume(0.5)  
 sound\_3card.play(0, 0)  
 if n\_dacheng\_num == 3:  
 sound\_4card = pygame.mixer.Sound('翻出四组牌.ogg') # 播放4血！  
 sound\_4card.set\_volume(0.5)  
 sound\_4card.play(0, 0)  
 if n\_dacheng\_num == 4:  
 sound\_5card = pygame.mixer.Sound('翻出五组牌.ogg') # 播放5血！  
 sound\_5card.set\_volume(0.5)  
 sound\_5card.play(0, 0)  
 if n\_dacheng\_num == 5:  
 sound\_6card = pygame.mixer.Sound('翻出六组牌.ogg') # 播放6血！  
 sound\_6card.set\_volume(0.5)  
 sound\_6card.play(0, 0)  
 if n\_dacheng\_num == 6 :  
 sound\_7card = pygame.mixer.Sound('翻出七组牌.ogg') # 播放7血！  
 sound\_7card.set\_volume(0.5)  
 sound\_7card.play(0, 0)  
 if n\_dacheng\_num >= 7:  
 sound\_8card = pygame.mixer.Sound('超神.ogg') # 播放8血！  
 sound\_8card.set\_volume(0.5)  
 sound\_8card.play(0, 0)  
 n\_lianxudachen = 1  
 if n\_lianxudachen >= 2:  
 if n\_lianxudachen ==2:  
 sound\_2kill = pygame.mixer.Sound('双杀.ogg') # 播放双杀！  
 sound\_2kill.set\_volume(0.5)  
 sound\_2kill.play(0, 0)  
 if n\_lianxudachen ==3:  
 sound\_3kill = pygame.mixer.Sound('三杀.ogg') # 播放三杀！  
 sound\_3kill.set\_volume(0.5)  
 sound\_3kill.play(0, 0)  
 if n\_lianxudachen ==4:  
 sound\_4kill = pygame.mixer.Sound('四杀.ogg') # 播放四杀！  
 sound\_4kill.set\_volume(0.5)  
 sound\_4kill.play(0, 0)  
 if n\_lianxudachen == 5:  
 sound\_5kill = pygame.mixer.Sound('五杀.ogg') # 播放五杀！  
 sound\_5kill.set\_volume(0.5)  
 sound\_5kill.play(0, 0)  
  
  
  
 firstSelection = None # 这时候重新设置第一次翻牌的位置，为没有翻牌状态  
  
  
 # 时刻更新界面  
 pygame.display.update()  
 FPSCLOCK.tick(FPS)  
  
#以下是所有的轮子，手动分割线—————————————————————————————————————————————  
  
def generateRevealedCardsData(val): #这个轮子用于画出游戏界面，是一个列表式的数据结构，同样记下标记#####################  
  
 revealedCards = []  
 for i in range(wide\_num):  
 revealedCards.append([val] \* high\_num)  
 return revealedCards  
  
  
def getRandomizedBoard(): #这个轮子同样用于画出游戏界面，随机绘制出不同的图案  
 # 一开始，图标的存储列表当然是空的，这里利用了列表的可变性，每次利用完一个图案，后面可以进行删除  
 icons = []  
 for color in ALLCOLORS: #这里用一个嵌套的for循环去获取随机的颜色和图案，这里产出了所有的图案颜色组合  
 for shape in ALLSHAPES:  
 icons.append( (shape, color) )  
  
 random.shuffle(icons) # 随机打乱每个图标在列表中的顺序  
 numIconsUsed = int(wide\_num \* high\_num / 2) # 统计出总共需要多少个图案  
 icons = icons[:numIconsUsed] \* 2 # 这里更新图案列表，将图案列表中的前numIconsUsed个图案\*2，获得所有需要的图案  
 random.shuffle(icons) #这里再次将列表中所有图案的顺序打乱  
  
 # 这里绘制出所有牌面组合出来的样式，并且填充进入图案  
 board = [] #整个版面一开始，当然也是一个空的列表  
 for x in range(wide\_num):  
 column = []  
 for y in range(high\_num):  
 column.append(icons[0])  
 del icons[0] # 每次删除完图案列表里刚刚填入的那个图案  
 board.append(column)  
 return board  
 #上述画完坐标轴如下  
 # 0 1 2 3 4 5  
 # 0  
 # 1  
 # 2  
 # 3  
 # 4  
  
def List1D\_into\_list2D(groupSize, theList):  
 # 将一维列表分解成二维列表的一个简单轮子  
 result = []  
 for i in range(0, len(theList), groupSize):  
 result.append(theList[i:i + groupSize])  
 return result  
  
  
def Actural\_position(Card\_x\_position, Card\_y\_position):  
 # 这个轮子将方块坐标转化为像素坐标，供游戏本体识别  
 left = Card\_x\_position \* (Card\_size + Distance\_between\_card) + X\_margin\_横轴边缘像素  
 top = Card\_y\_position \* (Card\_size + Distance\_between\_card) + Y\_margin\_纵轴边缘像素  
 return (left, top)  
  
  
def getCardAtPixel(x, y): #这是一个统一像素坐标和方块坐标的轮子，将像素坐标转化为方块坐标  
 for Card\_x\_position in range(wide\_num):  
 for Card\_y\_position in range(high\_num):  
 left, top = Actural\_position(Card\_x\_position, Card\_y\_position) #这里获得光标下方块坐标的像素坐标值  
 CardRect = pygame.Rect(left, top, Card\_size, Card\_size) #这里获得每个牌的像素坐标  
 if CardRect.collidepoint(x, y): #如果光标下的像素坐标在某个牌下的像素坐标内部  
 return (Card\_x\_position, Card\_y\_position) #则返回那张牌的方块坐标  
 return (None, None) #若光标不在某个牌的像素坐标，那么返回，光标无选中牌面  
  
  
def drawIcon(shape, color, Card\_x\_position, Card\_y\_position): #这是绘制牌面图案的轮子  
 quarter = int(Card\_size \* 0.25) # syntactic sugar  
 half = int(Card\_size \* 0.5) # syntactic sugar  
  
 left, top = Actural\_position(Card\_x\_position, Card\_y\_position) # 这里获得像素坐标  
 # 这里绘制每种牌面  
 if shape == DONUT:  
 pygame.draw.circle(Background\_背景, color, (left + half, top + half), half - 5)  
 pygame.draw.circle(Background\_背景, BGCOLOR, (left + half, top + half), quarter - 5)  
 elif shape == SQUARE:  
 pygame.draw.rect(Background\_背景, color, (left + quarter, top + quarter, Card\_size - half, Card\_size - half))  
 elif shape == DIAMOND:  
 pygame.draw.polygon(Background\_背景, color, ((left + half, top), (left + Card\_size - 1, top + half), (left + half, top + Card\_size - 1), (left, top + half)))  
 elif shape == LINES:  
 for i in range(0, Card\_size, 4):  
 pygame.draw.line(Background\_背景, color, (left, top + i), (left + i, top))  
 pygame.draw.line(Background\_背景, color, (left + i, top + Card\_size - 1), (left + Card\_size - 1, top + i))  
 elif shape == OVAL:  
 pygame.draw.ellipse(Background\_背景, color, (left, top + quarter, Card\_size, half))  
  
  
def getShapeAndColor(board, Card\_x\_position, Card\_y\_position):  
 # board[x][y][0] 储存牌面的形状  
 # board[x][y][1] 储存牌面的颜色  
 return board[Card\_x\_position][Card\_y\_position][0], board[Card\_x\_position][Card\_y\_position][1] #返回一个牌面内容的颜色与形状  
  
  
def drawCardCovers(board, Cards, coverage):  
 # Draws Cards being coveRed\_红/revealed. "Cards" is a list  
 # of two-item lists, which have the x & y spot of the Card.  
 for Card in Cards:  
 left, top = Actural\_position(Card[0], Card[1])  
 pygame.draw.rect(Background\_背景, BGCOLOR, (left, top, Card\_size, Card\_size)) #自己修改记下标记#######################################  
 shape, color = getShapeAndColor(board, Card[0], Card[1])  
 drawIcon(shape, color, Card[0], Card[1])  
 if coverage > 0: # only draw the cover if there is an coverage Background\_背景.blit(Background\_背景图片, (0, 0))  
 Card\_cover\_卡牌背面图片 = pygame.image.load('卡牌背面.jpg').convert() #记下标记##############################################  
 Background\_背景.blit(Card\_cover\_卡牌背面图片,(left, top))  
  
 pygame.display.update()  
 FPSCLOCK.tick(FPS)  
  
  
def revealCardsAnimation(board, CardsToReveal):  
 # Do the "Card reveal" animation.  
 for coverage in range(Card\_size, (-Speed\_of\_open) - 1, -Speed\_of\_open):  
 drawCardCovers(board, CardsToReveal, coverage)  
  
  
def coverCardsAnimation(board, CardsToCover):  
 # 翻回动画  
 for coverage in range(0, Card\_size + Speed\_of\_open, Speed\_of\_open):  
 drawCardCovers(board, CardsToCover, coverage)  
  
  
def drawBoard(board, revealed):  
 # 画出所有牌面组成的整体  
 for Card\_x\_position in range(wide\_num):  
 for Card\_y\_position in range(high\_num):  
 left, top = Actural\_position(Card\_x\_position, Card\_y\_position)  
 if not revealed[Card\_x\_position][Card\_y\_position]:  
 # 记下标记#########################################################################################  
 Card\_cover\_卡牌背面图片 = pygame.image.load('卡牌背面.jpg').convert() # 记下标记##############################################  
 Background\_背景.blit(Card\_cover\_卡牌背面图片, (left, top))  
 else:  
 # Draw the (revealed) icon.  
 shape, color = getShapeAndColor(board, Card\_x\_position, Card\_y\_position)  
 drawIcon(shape, color, Card\_x\_position, Card\_y\_position)  
  
  
def drawHighlightCard(Card\_x\_position, Card\_y\_position):  
 left, top = Actural\_position(Card\_x\_position, Card\_y\_position)  
 pygame.draw.rect(Background\_背景, HIGHLIGHTCOLOR, (left - 5, top - 5, Card\_size + 10, Card\_size + 10), 4)  
  
  
def startGameAnimation(board):  
 # Randomly reveal the Cards 8 at a time.  
 coveRed\_红Cards = generateRevealedCardsData(False)  
 Cards = []  
 for x in range(wide\_num):  
 for y in range(high\_num):  
 Cards.append( (x, y) )  
 random.shuffle(Cards)  
 CardGroups = List1D\_into\_list2D(8, Cards)  
  
 drawBoard(board, coveRed\_红Cards)  
 for CardGroup in CardGroups:  
 revealCardsAnimation(board, CardGroup)  
 pygame.time.wait(700)  
 coverCardsAnimation(board, CardGroup)  
  
  
def gameWonAnimation(board):  
 # 游戏结束动画的轮子######################################  
 coveRed\_红Cards = generateRevealedCardsData(True)  
 color1 = LIGHTBGCOLOR  
 color2 = BGCOLOR  
 Card\_cover\_卡牌背面图片 = pygame.image.load('卡牌背面.jpg').convert() # 记下标记##############################################  
  
  
 for i in range(0,3):  
 if i == 0:  
 Win\_pic = pygame.image.load('胜利图1.jpg').convert()  
 Background\_背景.blit(Win\_pic,(0,0))  
 drawBoard(board, coveRed\_红Cards)  
 sound\_ACE = pygame.mixer.Sound('团灭.ogg') #播放团灭！  
 sound\_ACE.set\_volume(0.5)  
 sound\_ACE.play(0, 0)  
 pygame.display.update()  
 pygame.time.wait(2000)  
 if i ==1 :  
 Win\_pic\_2 = pygame.image.load('胜利图2.jpg').convert()  
 Background\_背景.blit(Win\_pic\_2, (0, 0))  
 drawBoard(board, coveRed\_红Cards)  
 pygame.display.update()  
 sound\_xipai = pygame.mixer.Sound('洗牌.ogg') # 播放开始！  
 sound\_xipai.set\_volume(0.7)  
 sound\_xipai.play(0, 0)  
 pygame.time.wait(2000)  
  
 if i ==2 :  
 Win\_pic\_3 = pygame.image.load('胜利图3.jpg').convert()  
 Background\_背景.blit(Win\_pic\_3, (0, 0))  
 drawBoard(board, coveRed\_红Cards)  
 sound\_begin = pygame.mixer.Sound('是时候表演真正的技术了.wav') # 播放开始！  
 sound\_begin.set\_volume(0.7)  
 sound\_begin.play(0, 0)  
  
  
 #################  
 pygame.display.update()  
 pygame.time.wait(2000)  
  
  
def hasWon(revealedCards):  
 #检查胜利条件的轮子  
 for i in revealedCards:  
 if False in i:  
 return False # 如果有没有翻开的牌面，那么游戏未完成  
 return True  
#####################################################################  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()