

Objetivo do Projeto

Com este projeto o meu objetivo era criar um jogo de memória através de um baralho de carta, que fosse um pouco diferente da proposta do professor, tendo adicionado como *feature* o movimento vertical das cartas com uma velocidade relativa aos batimentos por minuto (BPM) de cada música referente a cada nível. Para isto transformei o BPM de cada musica em milissegundos e atribui esse valor para definir a velocidade a que as cartas se movem.

Abordagem ao Problema

Inicialmente comecei por tentar programar um menu principal, utilizando apenas um ficheiro, com todos os botões pretendidos, ou seja, os botões para cada nível e um botão de saída que encerra o jogo. Para isto criei uma classe *Button* que recebe todos os parâmetros necessários para a funcionalidade pretendida.

Após ter criado todos os botões comecei por programar o primeiro nível "4x3", utilizando um ficheiro separado do ficheiro do *main menu* que é chamado ao clicar no botão "4x3". Para isto criei a classe *Card* com todos os parâmetros que são necessários para aquilo que é o jogo em si.

Depois de feitas as classes, criei uma lista "manual" de todas as cartas de jogo que incluíam as suas posições, tamanhos, cor e um valor utilizado para a comparação entre cartas.

Com o *Game Board* criado inicializei o ciclo que mantém o jogo a correr, desenhando todas as cartas no ecrã, voltando cada carta com uma forma e uma cor, comparando-as com a escolha seguinte do jogador e atribuindo um valor de pontuação a mais caso o jogador acerte e removendo as cartas combinadas ou removendo pontos e voltando as cartas de novo para baixo.

Nesta fase inicial ainda não seria implementada a musica e o movimento vertical das cartas.

Com a evolução do projeto e com a sugestão/ajuda do professor transformei os vários ficheiros de cada nível num só ficheiro que é geral a todos os níveis recebendo 2 variáveis das quais o código está dependente, tendo terminado o projeto com apenas 3 ficheiros, o principal do qual é inicializado o programa, o do *main menu* e o ficheiro de nível.

Estrutura do Código

- *shuffle.py* (Ficheiro Principal)
 - Recebe todas as funções e inicia a função de nível consoante as variáveis dadas, que representam o numero de cartas na vertical, na horizontal e o espaço de tempo que a carta tem para se mover, em milissegundos.
 - Dá load há musica do *main menu* e inicia-a logo a seguir.

```

# imports pygame into our program
import pygame
import pygame.freetype
from Level_01 import levels
from main_menu import Main_Menu

# Class that makes the buttons and takes input from the mouse

def main():
    pygame.init()
    pygame.mixer.music.load('main_menu.ogg')
    pygame.mixer.music.play(-1)
    # Game loop, to keep the game running until window closed of "Exit" button pressed
    Level = Main_Menu(True)
    while (True):
        if (Level == 1):
            Level = levels(4, 3, 480)

        elif (Level == 2):
            Level = levels(4, 4, 454)

        elif (Level == 3):
            Level = levels(5, 4, 434)

        elif (Level == 4):
            Level = levels(6, 5, 370)

        elif (Level == 5):
            Level = levels(6, 6, 300)

main()

```

- Level_01.py (Ficheiro com o código do baralho e constitui 90% do jogo.)
 - Classe "Cards"
 - Recebe todos os parametros necessários para o jogo existir, sendo estes a posição (*tuplo(x,y)*), cor (*tuplo(R,G,B)*), tamanho *width*, tamanho *height*, match (*tuplo(forma, cor)*), variável que é usada para comparar as cartas de maneira a eliminá-las ou voltá-las para baixo de novo e uma variável que verifica se a carta está voltada para cima ou não.

```

class Card:
    def __init__(self, color, position, width, height, match):
        self.color = color
        self.position = position
        self.width = width
        self.height = height
        self.match = match
        self.selected = False

```

- Recebe uma função *draw()* que desenha a carta no ecrã os parametros passados na classe.

```
#Function to draw cards on screen while face down
def draw(self, screen, fill = 0):
    pygame.draw.rect(screen, self.color, (self.position[0],self.position[1],self.width,self.height), fill)
```

- Recebe uma função *isOver()* que verifica se o rato do jogador está por cima da carta.

```
#Function that checks if mouse is over the cards
def isOver(self, pos):
    if pos[0] > self.position[0] and pos[0] < self.position[0] + self.width:
        if pos[1] > self.position[1] and pos[1] < self.position[1] + self.height:
            return True
    return False
```

- Refere um função *shape_draw()* que verifica se a carta está seleccionada e, através da variável *match[0]*, o valor da forma, define que forma vai ser desenhada na carta após esta ser seleccionada. Se *match[0]* for 0 desenha um círculo, se 1 desenha um quadrado, se 2 desenha um triângulo. Todo o código para cada forma é definido pelos valores da carta em si.

```
def shape_draw(self, screen):
    if (self.selected):
        if self.match[0] == 0: # Draws Circle if Shape value = 0
            pygame.draw.circle(screen, self.match[1], (self.position[0]+self.width//2,self.position[1]+self.height//2), self.width//4)
        elif self.match[0] == 1: # Draws Square if Shape value = 1
            pygame.draw.rect(screen, self.match[1], (self.position[0]+self.width//4,self.position[1]+self.height//3,self.width//2,self.height//3))
        else: # Draws Triangle if Shape value = 2
            pygame.draw.polygon(screen, self.match[1], [(self.position[0]+self.width//2,self.position[1]+self.height//3), (self.position[0]+self.width//4,self.position[1]+self.height//3), (self.position[0]+self.width//2,self.position[1]+self.height//3)])
```

○ Classe "Buttons"

- Recebe a classe "Buttons" definida no *main_menu.py*
- É utilizada apenas para colocar o botão "Exit" no ecrã de forma a voltar ao *main_menu*

○ Função *check_pair()* utilizada para comparar as duas cartas seleccionadas

```
#Function that compares both selected cards
def check_pair (c1, c2):
    if c1.match[0] == c2.match[0] and c1.match[1] == c2.match[1]:
        return True
    else:
        return False
```

- Função *levels(Cx,Cy,time)* que define o jogo e recebe as variáveis que definem o tamanho do *deck* de jogo e a variável *time* que define a velocidade a que as cartas mexem.
- No início da função, defino todos os valores típicos do pygame, como o tamanho da janela (res), a definir a área de jogo consoante a janela (screen) e a fonte (my font) utilizada nos botões e score.

```
#Defines window resolution
res = (1290, 712)

#Draws the window with previous parameters
screen = pygame.display.set_mode(res)

#Loads font to be used while running the program
my_font = pygame.freetype.Font("NotoSans-Regular.ttf", 23)
```

- De seguida começo por definir as possíveis coordenadas de cada carta, utilizando 3 listas que são iniciadas vazias, *Card_Position*, *pos_x*, *pos_y*. 1º Defino o tamanho total, em *y*, do tabuleiro de jogo (*Total_H*) e o espaço entre cada carta (*padding*), depois calculo o tamanho de cada carta para esse nível, utilizando as variáveis *Board*, passadas pela função, a *Total_H* e o *padding*. Com isto consigo calcular os restantes valores em torno de *x*. Finalmente junto a lista de posições em *x* e as posições em *y* na lista *Card_Position*.

```
# Creates array of X_positon and Y_Position, creates all possible card locations
Card_position = []
padding = 10
Total_H = 550
Board = (Cx, Cy)
Card_H = (Total_H - (Board[1]-1)*padding)//Board[1]
y = 50 - Card_H
pos_y = []
for i in range(Board[1]):
    y = y + Card_H + padding
    pos_y.append(y)

pos_x = []
Card_W = int(Card_H - Card_H//Board[1])
x = 360 - Card_W
for i in range(Board[0]):
    x = x + Card_W + padding
    pos_x.append(x)

for i in pos_x:
    for j in pos_y:
        Card_position.append((i,j))
```

- No próximo passo crio novamente 3 listas, a lista de cores usadas durante o jogo (*RGB*), a lista de formas a desenhar (*Shapes(1,2,3)*), e uma lista vazia onde vou juntar a lista de *RGB* e *Shapes* e vou dar *shuffle* de maneira a randomizar cada jogo.

```

# Array of all colors used for the shapes
RGB = [(0, 255, 255), (255, 0, 255), (0, 0, 255), (255,0,0), (0,255,0), (255,255,0)]

#Array of of numbers used in the function draw_shape to
# determine which shape will be drawned
Shapes = [0, 1, 2]

#Creates empty array of the deck for the level
Game_Deck = []

#Creates array of all cards for the specific level and shuffles it
Possible_Cards = []
for j in Shapes:
    for k in RGB:
        Possible_Cards.append((j,k))
random.shuffle(Possible_Cards)

```

- Finalmente crio o *Deck de Jogo* utilizando a lista vazia *Card_Sequence* onde junto metade do total de cartas de forma a criar os pares totais e dou *shuffle* no final antes de efetivamente criar a lista final com o total de cartas para o nível ativo, utilizando as variáveis criadas anteriormente, e termino, desenhando-as antes de iniciar o ciclo do jogo.

```

#Creates array of all pairs for the specific level and shuffles it
Card_Sequence = []
for i in range((Board[0]*Board[1]) // 2):
    Card_Sequence.append(Possible_Cards[i])
    Card_Sequence.append(Possible_Cards[i])
random.shuffle(Card_Sequence)

# Adds all cards to the Deck to be used with all the parameters created before
for i in range(len(Card_Sequence)):
    Game_Deck.append(Card((0,255,0), Card_position[i], Card_W, Card_H, Card_Sequence[i]))

#Shuffles the Deck before starting the game to assure different outcomes every game
random.shuffle(Game_Deck)

#Draws every card, face down, in the beginning of the game
for card in Game_Deck:
    card.draw(screen)

```

- Antes de inicar o ciclo do jogo defino também todas as variáveis utilizadas durante o ciclo.

```
#Inititalize all variables used during the game
num_cards_selected = 0 #Used to see how many cards the player has clicked
score = 0 # Score at the beginning of the game
p_attempt = 0 # Used to take points from the players score based on player moves
pair_set = (len(Game_Deck)//2) # Total of pairs in the specific level

#variables used to load and play victory music during loop
end_song = False
load_song = True

MOVE_SIDE = time
move_side_event = pygame.USEREVENT + 1
pygame.time.set_timer(move_side_event, MOVE_SIDE)
```

- Inico o ciclo onde o jogo acontece, onde utilizo todas as funções e variáveis criadas de forma a produzir o jogo completo.

```
while (True):
    screen.fill((0,0,20))
    clock.tick(120)
    if load_song == False: ...

    # Gets position of the mouse during game
    pos = pygame.mouse.get_pos()

    # Detectes if mouse is pressed during game
    mb = pygame.mouse.get_pressed()

    # Cycle used to select cards and compare them
    for card in Game_Deck: ...
    for event in pygame.event.get(): ...

    # Detects if the player clicked the 'Exit' Button, stopping the music
    # and sending the player to the main menu
    if event.type == pygame.MOUSEBUTTONDOWN: ...

    # Draws the 'Exit' Button and changes it's color if the player hovers it
    for button in Button_Set: ...

    if pair_set == 0: ...

    # Prints the Score with the Value on screen
    my_font.render_to(screen, (20, 20), ('Score:' + str(score)), (255,255,0))

    # Flips everything that was buffered and draws it in the screen
    pygame.display.flip()
```

- *main.menu.py*

- Classe "Buttons"
 - Semelhante á classe "Cards" tem duas funções e é utilizada para definir todos os botões do menu principal.

```
class Button:
    def __init__(self, color, x,y,width,height, text=''):
        self.color = color
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.text = text
```

- Função *Main_Menu(bool)* onde se seleciona as variáveis que vão ser passadas à função *levels* e se dá load e play à música desse nível específico.

```
# Detects the click of the mouse on the buttons
if event.type == pygame.MOUSEBUTTONDOWN:
    for button in Button_Set:
        if button.isOver(pos):
            if button.text == 'EXIT':
                quit()

            elif button.text == '4x3':
                pygame.mixer.music.stop()
                pygame.mixer.music.load('level_01.ogg')
                pygame.mixer.music.play(-1)
                Level = 1
                return Level

            elif button.text == '4x4':
                pygame.mixer.music.stop()
                pygame.mixer.music.load('level_02.ogg')
                pygame.mixer.music.play(-1)
                Level = 2
                return Level

            elif button.text == '5x4':
                pygame.mixer.music.stop()
                pygame.mixer.music.load('level_03.ogg')
                pygame.mixer.music.play(-1)
                Level = 3
                return Level

            elif button.text == '6x5':
                pygame.mixer.music.stop()
                pygame.mixer.music.load('level_04.ogg')
                pygame.mixer.music.play(-1)
                Level = 4
                return Level

            elif button.text == '6x6':
                pygame.mixer.music.stop()
                pygame.mixer.music.load('level_05.ogg')
                pygame.mixer.music.play(-1)
                Level = 5
                return Level
```

Conclusão

Com este trabalho foi-me possível compreender muito melhor toda a matéria de python, pois foi possível aplicá-la a algo "real" e permitiu-me finalmente perceber a eficácia de "for" e ciclos, algo que sempre me causou dúvidas durante o semestre, tendo sido aqui onde tive mais dúvidas ao implementar um nível geral utilizando estes ciclos de forma a facilitar o aumento de cartas consoante o nível. Em suma gostei muito de realizar este projecto e apesar de não ter consigo fazer a minha ideia na totalidade e não ter conseguido implementar o *delay* depois de voltar duas cartas estou muito contente com o resultado final. **AVISO:**

Considere baixar o seu volume geral antes de iniciar o .exe pois não tive tempo de implementar um botão de *mute* ou um *slider* de volume.

Créditos e Referências

Todo o código foi realizado consultando toda a matéria leccionada pelo professor Diogo Andrade, disponível no moodle. Com especial ajuda dos meus colegas Daniel Fernandes, Marco Domingos e Pedro Miguel Marques, que me ajudaram a resolver certos problemas que se foram levantando ao longo do projecto. Classe de botões criada através da consulta deste vídeo: [https://www.youtube.com/watch?v=4_9twnEduFA]

Musicas(Youtube):

- "Main_Menu" - Duel of Fates 8 bits by GhostVoltage
- "4x3" - Shooting Stars 8 bits by 8 Bit Universe
- "4x4" - Star Wars Cantina Theme 8 bits by 8 Bit Universe
- "5x4" - Darude Sandstorm 8 bits by 8 Bit Universe
- "6x5" - Crazy Frog 8 bits by 8 Bit Universe
- "6x6" - We are number one 8 bits by 8 Bit Universe
- "Congratulations" - Pewdipie's Congratulations 8 bits by 8 Bit Universe