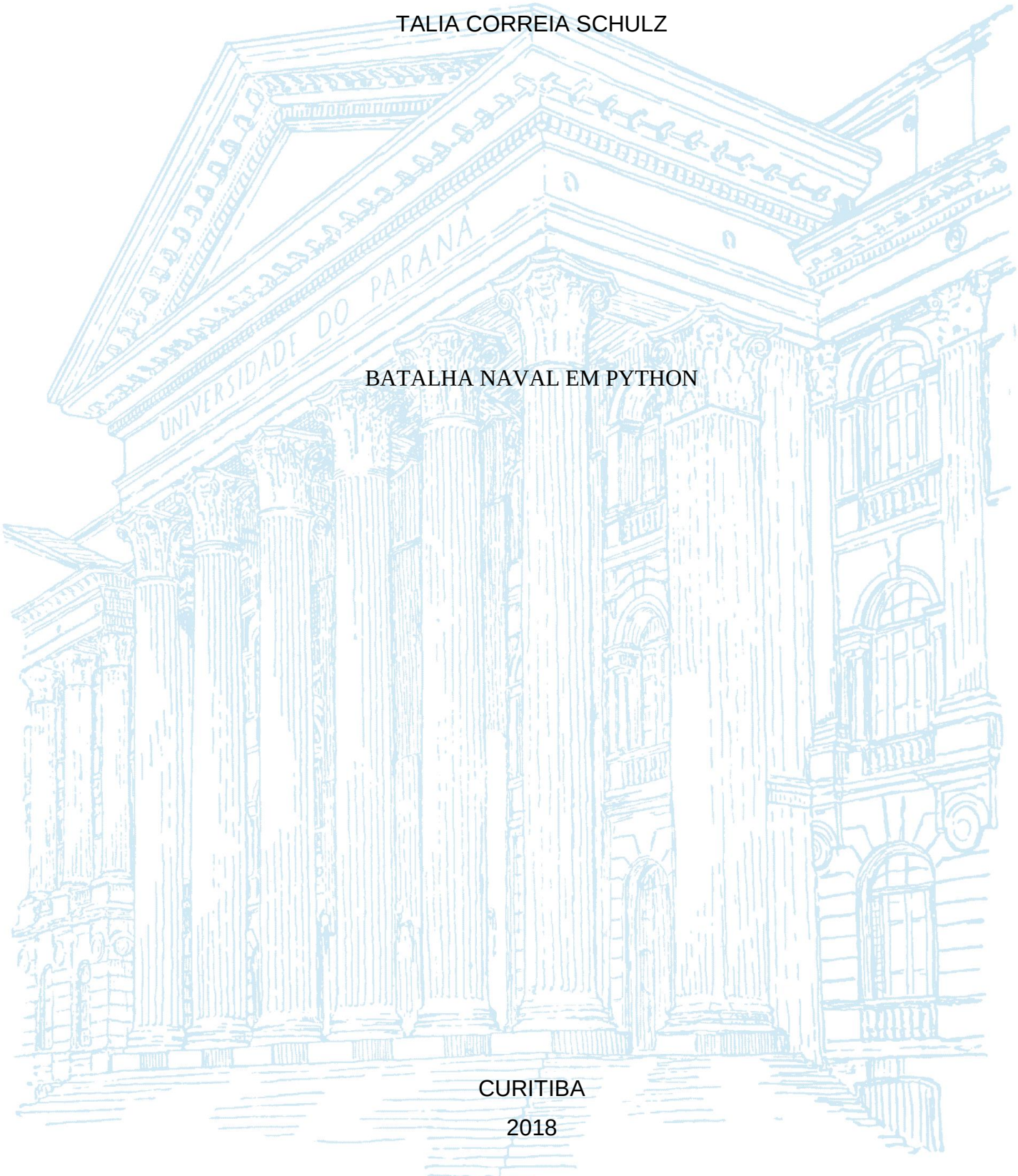
UNIVERSIDADE FEDERAL DO PARANÁ

AMANDA MACIEL DE OLIVEIRA

JOÃO VICTOR DA SILVA

TALIA CORREIA SCHULZ



BATALHA NAVAL EM PYTHON

CURITIBA

2018

AMANDA MACIEL DE OLIVEIRA
JOÃO VICTOR DA SILVA
TALIA CORREIA SCHULZ

BATALHA NAVAL EM PYTHON

Relatório apresentado à disciplina Fundamentos de Programação de Computadores do Curso de Graduação em Matemática da Universidade Federal do Paraná.

Orientador: Prof. Jackson Antônio do Prado Lima

CURITIBA
2018

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O presente relatório descreve em detalhes, o programa realizado em Python, o qual visa desenvolver um dos mais populares jogos, Batalha Naval.

Python é uma linguagem de programação de alto nível, isto é, sua sintaxe se aproxima mais da nossa linguagem e se distancia mais do código da máquina. Foi criada por Guido Van Rossum em 1991 com base na linguagem ABC e C.

1.1 JUSTIFICATIVA

Por ser uma forma simples e divertida de treinar os conhecimentos sobre a linguagem Python durante o semestre e se aprofundar nas suas aplicações.

1.2 OBJETIVO

O objetivo é adaptar um jogo divertido e de simples compreensão (Batalha Naval) que tem como motivação simular uma batalha entre navios de guerras, para jogar é necessário um jogador. O jogador dispõe sua frota em uma matriz, o objetivo é atingir toda a frota adversária através de disparos, estes por sua vez, podem ser classificados como “Água”, se o disparo não atingir nenhuma embarcação, e “Fogo”, caso contrário.

2 DESENVOLVIMENTO

2.1 FUNÇÕES

2.1.1 Dispondo as Embarcações

Após a escolha do tamanho da matriz, as embarcações devem ser dispostas, tanto a do computador quanto a do jogador.

2.1.1.1 Do Jogador

Após a escolha do tamanho da matriz, o jogador deve dispor suas embarcações, e essas por sua vez devem ser colocadas de acordo com alguns critérios, e são eles:

- Não colocar as embarcações sobrepostas;
- Não colocar as embarcações fora da matriz;
- O tamanho do barco deve estar de acordo com as posições inseridas pelo jogador;
- Não colocar as embarcações na diagonal.

Por isso criamos a função P1, que recebe as coordenadas e caso não cumpram as condições citadas acima, solicita novas coordenadas, segue abaixo o código da função.

```
def P1(bb,bardic,lista_PP,mat_p,tamtriz,matriz_ADMP,posicao = 0):
    '''
    Tipo : Player
    Técnico: Pede para pessoa inserir as coordenadas de todos os barcos e os verifica
    In game: Coloca os barcos do Player
    '''
    dic = {5 : "A" ,4 : "N",3 : "D", 2 : "P"} # tamanho : tipo
    while posicao != len(bb): # repete até o todos os barcos serem colocados
        print("\n")
        #primeira coordenada do barco
        coor = input("Primeira coordenada do seu {} (some {} em alguma coordenada):".format(bb[posicao],bardic[bb[posicao]]-1))
        while verificada(coor) == False: #verifica se a coordenada foi digitado corretamente
            print("ERRO!\nDigite coordenadas válidas")
            coor = input("Primeira coordenada do seu {} (some {} em alguma coordenada):".format(bb[posicao],bardic[bb[posicao]]-1)
        try: #transforma a primeira coordenada em dois valores isolados
            i,j = coor.split(",")
        except:
            i,j = coor.split()
        #última coordenada do barco
        coor = input("Última coordenada do seu {} :".format(bb[posicao]))
        while verificada(coor) == False: #verifica se a coordenada foi digitado corretamente
            print("ERRO!\nDigite coordenadas válidas")
            coor = input("Última coordenada do seu {} :".format(bb[posicao]))
        try:#transforma a última coordenada em dois valores isolados
            x,y = coor.split(",")
        except:
            x,y = coor.split()
        i,j,x,y = int(i),int(j),int(x),int(y)
        #linguiça de verificações sobre as coordenadas i,j,x,y
        while (j == y and max((x-i),(i-x))+1 != bardic[bb[posicao]]) or (i == x and max((y-j),(j-y))+1 != bardic[bb[posicao]]) or
```

Figura 1- Função P1 (Parte 1)

```
        print("ERRO!\nDigite coordenadas válidas")
        coor = input("Primeira coordenada do seu {} (some {} em alguma coordenada):".format(bb[posicao],bardic[bb[posicao]]-1))
        while verificada(coor) == False:
            print("ERRO!\nDigite coordenadas válidas")
            coor = input("Primeira coordenada do seu {} (some {} em alguma coordenada):".format(bb[posicao],bardic[bb[posicao]]
        try:
            i,j = coor.split(",")
        except:
            i,j = coor.split()
        coor = input("Última coordenada do seu {} :".format(bb[posicao]))
        while verificada(coor) == False:
            print("ERRO!\nDigite coordenadas válidas")
            coor = input("Última coordenada do seu {} :".format(bb[posicao]))
        try:
            x,y = coor.split(",")
        except:
            x,y = coor.split()
        i,j,x,y = int(i),int(j),int(x),int(y)
    lista_ijxy = [] #lista auxiliar para verificar se as coordenadas oferecidas podem ser efetuadas
    col,lin = min(y,j),min(x,i) #pega o menor valor das coordenadas na linha e na coluna
    if x == i: #se o barco estiver na horizontal
        while col <= max(y,j): #cresce a variável col até chegar no max(y,j)
            lista_ijxy.append([x,col]) #adiciona esses pontos na lista auxiliar
            col += 1
    elif y == j: #se o barco estiver na vertical
        while lin <= max(x,i): #cresce a variável lin até chegar no max(x,i)
            lista_ijxy.append([lin,y]) #adiciona esses pontos na lista auxiliar
            lin += 1
    #pega TODOS os pontos do suposto barco e verifica se ja não existe um barco naqueles pontos
    for ponto in lista_ijxy:
        if ponto in lista_PP:
            print("ERRO!\nDigite coordenadas válidas") #caso já exista um barco ali, chama a função novamente
            posicao = P1(bb,bardic,lista_PP,mat_p,tamtriz,matriz_ADMP,posicao)
            pivo = 1#impede que as coordenadas que deram errado do suposto barco, entrem na matriz adiminsitrativa
            break
        else:
            pivo = 0
    if pivo == 0: #se tudo deu certo, e for possível colocar o barco nas posições dejadas
        for e in lista_ijxy:
            i,j = e
            mat_p[i][j] = "{}".format(dic[bardic[bb[posicao]]]) #troca essas posições pela letra do barco
        #cria a linha na matriz administrativa referente ao barco recém colodo
        #Tipo do barco,tiros levados,tamanho do barco,lista com os seus pontos
        matriz_ADMP.append([dic[bardic[bb[posicao]]],0,bardic[bb[posicao]],lista_ijxy])
```

Figura 2- Função P1 (Parte 2)

2.1.1.2 Do Computador

As disposições das embarcações do computador são feitas pela função COM, a qual gera aleatoriamente as posições para os barcos do computador, garantindo que a quantidades destes seja restringida a partir de uma fórmula matemática baseada na dimensão da matriz anteriormente escolhida pelo jogador.

```
def COM(n,mat,lista_pontos,matriz_barcos):
    '''
    Tipo : COM
    Técnico: Gera a matriz com TODOS os barcos posicionados corretamente
    In game: Implícito
    '''
    num_barcos = floor((n*n)/64) #quantidade de barcos(nossa função)
    tamanhos = [5,4,3,2] #tamanhos dos tipos dos barcos em ordem
    dic = {5 : "A" ,4 : "N",3 : "D", 2 : "P"} #troca o tamanho do barco pela sua letra
    # os 2 for's pegam todos os tipos de barcos juntos (primeiro todos os A's depois os N' ...)
    for t in tamanhos:
        for a in range(num_barcos):
            i,j,x,y = coordenada(n,t) #gera as coordenadas do barco
            jogada(i,j,x,y,t,mat,dic[t],lista_pontos,n,matriz_barcos) #efetua a verficação dos pontos e gera a matriz dos barcos
```

Figura 3- Função COM

A função jogada, chamada dentro de COM e a função barcos são as que fazem a checagem dos pontos colocados, seguindo o mesmo critério definido para o Jogador.

```
def jogada(i,j,x,y,t,mat,carimbo,lista_pontos,n,matriz_barcos):
    '''
    Tipo: COM
    Técnico: Pega duas coordenadas (i,j) e (x,y) e preenche elas junto com todos os pontos que
    In game: Coloca UM barco entre duas coordenadas dadas(incluindo as próprias)
    '''
    i,j,x,y,lista_pontos = barcos(n,t,i,j,x,y,lista_pontos) #Verifica se os pontos são válidos
    lista_aux = []
    if i == x: #(Barco na horizontal)
        for b in range(t): # Troca as coordenadas entre (i,j) e (x,y) pelo tipo do barco (A,N,
            mat[i][j] = carimbo #Carimba o tipo do barco na Matriz de barcos do COM
            refil(i,j,carimbo,lista_aux) #Gera a lista que vai ser adicionada na Matriz admini
            j += 1 #Cresce as colunas
        matriz_barcos.append(lista_a) # Adiciona a lista criada no refil() na matriz administr
    elif j == y: #(Barco na vertical)
        for b in range(t): # Troca as coordenadas entre (i,j) e (x,y) pelo tipo do barco (A,N,
            mat[i][j] = carimbo #Carimba o tipo do barco na Matriz de barcos do COM
            refil(i,j,carimbo,lista_aux) #Gera a lista que vai ser adicionada na Matriz admini
            i += 1 #Cresce as linhas
        matriz_barcos.append(lista_a) # Adiciona a lista criada no refil() na matriz administr
#tamanho a matriz, tamanho do barco, coordenadas (x,y,j,i), lista com coordenadas de todos os
```

Figura 4- Função Jogada

```
def barcos(n,t,i,j,x,y,lista_pontos):
    '''
    Tipo : COM
    Técnico: Verifica se os pontos dados e os pontos que estiverem entre eles, para ver s
    In game: Implícita
    '''
    aux,i1,j1 = 0,i,j # Para modificar i ou j sem alterar o valor original
    if i == x: #(Barco na horizontal)
        lista_aux = [] # adicinamos temporariamente os pontos até validação da regra(dois
        for elemento in range(t): # varia o J com o tamanho do barco
            if [i,j1] in lista_pontos: #confere se os pontos estão na lista que tem os por
                aux = 1 #if mais a frente
                break
            lista_aux.extend([[i,j1]]) #adiciona o ponto
            j1 += 1
        if aux == 1: # só cai aqui dentro se algum ponto já estiver preenchido (na lista
            #gera novas coordenadas para validação
            i,j,x,y = coordenada(n,t)
    #se algum ponto estiver,chama a funçao novamente com os novos pontos para efetuar a j
            i,j,x,y,lista_pontos = barcos(n,t,i,j,x,y,lista_pontos)
        else:
    #caso todos os pontos não estejam na lista de pontos dos barcos, adiciona os pontos v
            lista_pontos.extend(lista_aux)
    elif j == y:# (Barco na vertical)
        lista_aux = [] # adicinamos temporariamente os pontos até validação da regra(dois
        for elemento in range(t): # varia o I com o tamanho do barco
            if [i1,j] in lista_pontos: #confere se os pontos estão na lista que tem os por
                aux = 1 #if mais a frente
                break
            lista_aux.extend([[i1,j]])#adiciona o ponto
            i1 += 1
        if aux == 1: # só cai aqui dentro se algum ponto já estiver preenchido (na lista
            #gera novas coordenadas para validação
            i,j,x,y = coordenada(n,t)
    #se algum ponto estiver,chama a funçao novamente com os novos pontos para efetuar a j
            i,j,x,y,lista_pontos = barcos(n,t,i,j,x,y,lista_pontos)
        else:
    #caso todos os pontos não estejam na lista de pontos dos barcos, adiciona os pontos v
            lista_pontos.extend(lista_aux)
    return i,j,x,y,lista_pontos
```

Figura 5- Função Barcos

2.1.2 Tiros

Com os barcos dispostos, o jogo começa. Os tiros são feitos de forma alternada começando sempre pelo jogador.

2.1.2.1 Do Jogador

Os tiros realizados pelo jogador não podem acontecer fora da matriz e também não podem acontecer no mesmo lugar. A função denominada TiroP, (Figura 6), é a que verifica e gerencia. Caso alguma dessas exceções aconteça novas coordenadas devem ser solicitadas para o jogador.

Também dentro dessa função é feito uma verificação do modo como as coordenadas foram escritas, pois caso haja alguma incoerência, o programa mostra uma mensagem de erro e requisita novas coordenadas.

Além disso, a função verifica se os tiros dados acertaram ou não alguma embarcação adversária, contabilizando os acertos do jogador, e modifica a matriz do jogo, para em caso de acerto ele trocar, “~” por “X” e em caso de erro trocar por “O”.

```
def tiroP(mati_paratiro,tamtriz,lista_PP,lista_tirop,matriz_ADMCOM,P_pontos):
    '''
    Tipo : Player
    Técnico: Pede pra pessoa atirar em uma coordenada e verifica se esse tiro é válido
    In game: Função que cuida dos tiros do player
    '''
    coor = input("Atire: ")
    while verificada(coor) == False: #verifica se a coordenada foi digitado corretamente
        print("ERRO!\nDigite coordenadas válidas")
        coor = input("Atire: ")
    try:
        i,j = coor.split(",")
    except:
        i,j = coor.split()
    i,j = int(i),int(j)
    # verifica se (i,j) está dentro do campo de batalha e se a pessoa já atirou nesse mesmo local
    while (i not in range(1,tamtriz+1)) or (j not in range(1,tamtriz+1)) or ([i,j] in lista_tirop): #+1 para ch
        print("Coordedas inválidas")
        coor = input("Atire: ")
        while verificada(coor) == False: #verifica se a coordenada foi digitado corretamente
            print("ERRO!\nDigite coordenadas válidas")
            coor = input("Atire: ")
        try:
            i,j = coor.split(",")
        except:
            i,j = coor.split()
        i,j = int(i),int(j)
    lista_tirop.append([i,j]) #com o tiro validado, adiciona na lista com os tiros do player
```

Figura 6- Função TiroP

Dentro da função tiro existe uma outra função a Contiro, (Figura 7), essa por sua vez, só é chamada caso algum barco do adversário seja atingido. Ou seja, ela

verifica qual dos barcos sofreu o ataque e os contabiliza, modificando a segunda coluna da matriz administrativa do jogador, onde é feita a contagem dos tiros levados de cada embarcação.

```
def contiros(i,j,matriz_ADMP):
    '''
    Tipo : Geral
    Técnico: Verifica se algum ponto de algum barco levou o tiro (i,j), contabiliza
    In game: Implícita
    '''
    linha = 0 #define qual barco vai ser escolhido
    for f in matriz_ADMP: #pega as linhas da matriz administrativa do player
        for u in matriz_ADMP[linha][3]: #pega a lista que contem todos os pontos do
            if [i,j] == u: # verifica se o tiro (i,j) acertou alguma posição desse b
                matriz_ADMP[linha][1] += 1 # se sim marca um tiro levado
        linha += 1 # varia o barco
```

Figura 7- Função Contiro

2.1.2.2 Do Computador

Os tiros do Computador são realizados de forma aleatória, assim como nos tiros do jogador, os que são realizados pelo Computador devem seguir algumas condições, tais como: não atirar no mesmo lugar e não atirar fora da matriz. Caso ocorra alguma dessas exceções devem ser geradas novas coordenadas.

Esse controle é feito pela função Tiro (Figura 8), essa função também gerencia os acertos e erros, isto é, quando o computador gerar aleatoriamente os tiros, essa função vai verificar se acertou um barco da frota adversária, se sim, nas respectivas coordenadas troca, "~" por "X" e adiciona um ponto na pontuação do computador, ou "~" por "O", caso contrário.

```
def tiro(matiro,n,lista_pontos,lista_tiro,matriz_ADMP,PCOM_pontos): #COM
    x = randint(1,n) # começa em 1 pra não pegar a linha/coluna dos números
    y = randint(1,n) # começa em 1 pra não pegar a linha/coluna dos números
    while [x,y] in lista_tiro: #verifica se o COM já atirou naquele lugar
      x = randint(1,n)
      y = randint(1,n)
    if [x,y] in lista_pontos: #Se acertou um barco
      matiro[x][y] = "X" #troca ~ por X
      PCOM_pontos += 1 #Pontos do COM
      lista_tiro.append([x,y]) # Adiciona o tiro no histórico
      contiros(x,y,matriz_ADMP) # Valida esse tiro dentro da matriz administrativa do Player
    else: #Se errou o barco
      matiro[x][y] = "O" #troca ~ por O
      lista_tiro.append([x,y]) # Adiciona o tiro no histórico
    return PCOM_pontos
```

Figura 8- Função Tiro

Novamente quando o Computador acerta algum barco adversário, dentro da função Tiro é chamada a função Contiro (Figura 7), descrita no item anterior.

2.1.3 Derrubando as Embarcações

Após um tiro ser dado, tanto pelo computador quanto pelo jogador, é necessário conferir se esse tiro acertou algum barco adversário, se sim é preciso verificar se o barco atingido foi derrubado ou não.

A função derrubada (Figura 9) faz esse trabalho. Ela utiliza a matriz administrativa de quem levou o tiro para fazer a verificação. Dentro da matriz a segunda coluna é responsável por armazenar quantos tiros o barco levou. Este é o motivo da criação da matriz administrativa.

```
def derrubada(mati_paratiro,matriz_barcos):
    '''
    Tipo : Geral
    Técnico: Verifica se o barco foi derrubado
    In game: Se o barco foi derrubado, troca todos os X do barco na matriz por sua respectiva letra
    '''
    linha = 0 #define qual barco vai ser escolhido
    for h in matriz_barcos: #pega as linhas da matriz administrativa do COM
        if matriz_barcos[linha][1] == matriz_barcos[linha][2]: #se os tiros levados forem iguais ao tamanho do
            for c in matriz_barcos[linha][3]: # pega todos os pontos do barco
                i,j = c
                mati_paratiro[i][j] = matriz_barcos[linha][0] #troca o X na posição i,j pela letra do barco
        linha += 1 #varia o barco
```

Figura 9- Função Derrubada

2.1.4 Funções Display

Para a saída do jogo são usadas as funções matriz (Figura 10), imprime (Figura 11) e imprime lado (Figura 12), estas, são responsáveis por: criar as matrizes (de ordem pré-definida pelo jogador) e formatar para que as matrizes saiam de forma apresentável e com aspecto de tabuleiro para uma melhor jogabilidade.

```
def matriz(n): #tamanho da matriz
    '''
    Tipo : Geral
    Técnico: cria uma matriz de ordem N
    In game: implícito
    '''
    matriz = []
    for i in range(n):
        matriz.append(n*["~"])
    return matriz
```

Figura 10-Função Matriz

```python
def imprime(mat,n): #printa mat de tamanho n
    '''
    Tipo : Geral
    Técnico: Imprime bonitinho uma matriz de ordem N
    In game: Mostra a matriz na tela
    '''
    for l in range(n):
        print(n*"----")
        for c in range(n):
            print("|{:^3}".format(mat[l][c]),end = "") #print bonitinho (:^3 força os prints a te
        print("|")
    print(n*"----")
```

Figura 11- Função Imprime

```python
def imprime_lado(mat1,mat2,n): #printa mat1 na esquerda, mat2 na direita, as duas de tamanho n
    '''
    Tipo : Geral
    Técnico: Imprime bonitinho duas matrizes de ordem N
    In game: Mostra as duas matrizes na tela, uma do lado da outra
    '''
    text = ""
    for l in range(n):
        text += (n*"----")+"     "+(n*"----")+"\n"
        for c1 in range(n):
            text += "|{:^3}".format(mat1[l][c1])
        text += "|    "
        for c2 in range(n):
            text += "|{:^3}".format(mat2[l][c2])
        text += "|    \n"
    text += (n*"----")+"     "+(n*"----")+"\n"
    print(text)
```

Figura 12- Função Imprime Lado

2.1.5 Função de Verificação

A função verificada faz a filtragem das jogadas do player para garantir que estas não sejam escritas de maneiras inválidas, como por caracteres ou sem espaços. Esta função é chamada dentro de outras, como por exemplo na função P1 e na função TiroP.

```python
def verificada(coor):
    '''
    Tipo : Player
    Técnico: Verifica se a coordenada foi escrita corretamente "i,j" ou "i j" e se i e j são números
    In game: Retorna true se a coordenada foi válida e False caso contrário
    '''
    try:
        i,j = coor.split(",") #tenta separar por vírgula
    except:
        try:
            i,j = coor.split() #tenta separar por espaço
        except:
            return False
    return (i.isdigit() and j.isdigit()) #após separar por virgula OU por espaço, verifica se são números
```

Figura 13- Função Verificada

2.2 PROGRAMA PRINCIPAL

Ao dar início à execução do jogo, o jogador se depara com um chart de opções, no qual pode decidir jogar, ler mais sobre o jogo ou sair. A última escolha dá fim ao código por meio do comando break. Todos esses comandos foram feitos por meio de laços de repetição e estruturas de controle condicionais.

Em seguida são feitas as verificações de posição, tiros, etc. como já descrito anteriormente, porém de maneira interativa com o jogador.

```
if __name__ == '__main__':
    verificador = 0
    while True: #continua rodando o jogo até a pessoa escolher sair
        menu = int(input("BATALHA NAVAL\n1) Jogar\n2) Sobre o jogo\n3) Sair\n=>"))
        print("\n")
        if menu == 3:
            print("Tchau Marujo, até a próxima.")
            break
        elif menu == 2:
            print('''Bem vindo Marujo!\nAntes de navegar, seguem algumas regras que regem o jogo:''')
            print("Para jogar BATALHA NAVAL,você terá a opção de desafiar o computador.\nFeita a esco
            print("sua frota será composta de:\nX PORTA-AVIÕES(tamanho 5)\nX NAVIOS-TANQUE(tamanho 4)
            print("obs: a quantidade de barcos varia de acordo com o tamanho do campo.\n\nSeus barcos
            print("\nSeu objetivo é afundar toda a frota inimiga tentando advinhar, a partir de jogad
            print("Portanto apresse-se para salvar sua frota! Boa Sorte, Marujo!\n")
            verificador = int(input("1) Voltar \n2) Sair\n=> "))
            print("\n")
            while verificador not in (1,2):
                verificador = int(input("1) Voltar \n2) Sair\n=> "))
                print("\n")
            if verificador == 2:
                print("Tchau Marujo, até a próxima.")
                break
        elif menu == 1:
            op = int(input("1) P vs COM\n2) Voltar\n=>"))
            print("\n")
            if op == 1:
                P1_pontos = PCOM_pontos = 0 # Conta quantos pontos o jogador fez
                #dicionário útil mais pra frente dentro das funções
                bardic = {"Porta-aviões" : 5,"Navios-tanque" : 4,"Destóier" : 3, "Pesca" : 2}
                #pega o tamanho da matriz
                tamtriz = int(input("Defina o tamanho do campo de batalha (Apenas UM número): "))
                if tamtriz > 20: #hehe
                    print("Oi Jackson")
                while tamtriz < 8 : #tamanho mínimo é 8
                    print("Tamanho inválido")
                    tamtriz = int(input("Defina o tamanho do campo de batalha: "))
                p1 = input("Marujo, insira seu nome: ") #Nome do marujo
                print("\n")
                quant_barcos = floor((tamtriz**2)/64) #pega a quantidade de cada tipo de barco
                mat_p1 = matriz(tamtriz+1) #cria a matriz de jogo (+1 por causa da linha e coluna adi
                matriz_ADMP1 = [] # matriz administrativa do P1
                aux_mat = matriz(tamtriz) #COM vai colocar os barcos aqui, depois colocar essa matriz
                aux_mat_ParaTiroCOM = matriz(tamtriz) #matriz para o P1 atirar
                aux_mat_ParaTiroP1 = matriz(tamtriz) #matriz para o COM atirar
                coluna = linha = 0
                for s in range(tamtriz+1): #adiciono uma linha e uma coluna a mais para printar bonit
                    mat_p1[0][coluna] = coluna
                    coluna += 1
                    mat_p1[linha][0] = linha
                    linha += 1
                imprime(mat_p1,tamtriz+1) #mostro a matriz do player para poder adicionar os barcos
                ba = quant_barcos*["Porta-aviões"]
                bn = quant_barcos*["Navios-tanque"]
                bd = quant_barcos*["Destóier"]
                bp = quant_barcos*["Pesca"]
                bb = []
```

Figura 14- Código Principal (Parte 1)

```
            ba = quant_barcos*["Porta-aviões"]
            bn = quant_barcos*["Navios-tanque"]
            bd = quant_barcos*["Destóier"]
            bp = quant_barcos*["Pesca"]
            bb = []
            #crio a lista com todos os barcos em ordem decrescente
            bb.extend(ba),bb.extend(bn),bb.extend(bd),bb.extend(bp)
            lista_PP1 = [] #lista pontos de p1, barcos dele
            lista_COM = [] #lista pontos do COM, barcos dele
            lista_tiroCOM = [] #lista dos tiros do COM
            lista_tiroP1 = [] #lista dos tiros do P1
            matriz_ADMCOM = [] #matriz administrativa do COM
            posicao = 0 #util denrto de P1
            posicao = P1(bb,bardic,lista_PP1,mat_p1,tamtriz,matriz_ADMP1) #player coloca seus ba
            Pontos_total = 0
            for ki in range(quant_barcos*4):# Conta quantas posições foram preenchidas (Máximo d
                Pontos_total += matriz_ADMP1[ki][2]
            COM(tamtriz,aux_mat,lista_COM,matriz_ADMCOM) #computador coloca seus barcos
            #adiciona uma linha e uma coluna em todas as coordenadas dos barcos (por causa do im
            for fi in range(len(lista_COM)):
                i,j = lista_COM[fi]
                lista_COM[fi] = [i+1,j+1]
            mat = matriz(tamtriz+1) #matriz com os barcos do COM
            mat_ParaTiroP1 = matriz(tamtriz+1) #matriz para P1 atirar
            mat_ParaTiroCOM = matriz(tamtriz+1) #matriz para COM atirar
            linha = coluna = 0
            for xi in range(tamtriz):
              for yi in range(tamtriz):
                mat[xi+1][yi+1] = aux_mat[xi][yi]
                mat_ParaTiroP1[xi+1][yi+1] = aux_mat_ParaTiroP1[xi][yi]
                mat_ParaTiroCOM[xi+1][yi+1] = aux_mat_ParaTiroCOM[xi][yi]
            for si in range(tamtriz+1): #adiciono uma linha e uma coluna para printar bonitinho
                mat[0][coluna] = coluna
                mat_ParaTiroP1[0][coluna] = coluna
                mat_ParaTiroCOM[0][coluna] = coluna
                coluna += 1
                mat[linha][0] = linha
                mat_ParaTiroP1[linha][0] = linha
                mat_ParaTiroCOM[linha][0] = linha
                linha += 1
            verificador2 = 0
            limpa_tela()
            while verificador2 == 0: #Mata mata acontece aqui dentro
                imprime_lado(mat_ParaTiroP1,mat_ParaTiroCOM,tamtriz+1)#imprime os dois campos de batalha
                #P1 atira
                P1_pontos = tiroP(mat_ParaTiroP1,tamtriz,lista_COM,lista_tiroP1,matriz_ADMCOM,P1_pontos)
                derrubada(mat_ParaTiroP1,matriz_ADMCOM) #verifica se algum barco do COM foi derrubado
                #COM atira
                PCOM_pontos = tiro(mat_ParaTiroCOM,tamtriz,lista_PP1,lista_tiroCOM,matriz_ADMP1,PCOM_pontos)
                derrubada(mat_ParaTiroCOM,matriz_ADMP1) #verifica se algum barco do P1 foi derrubado
                #imprime_lado(mat_ParaTiroP1,mat_ParaTiroCOM,tamtriz+1)
                #verifica se alguém ganhou
                sleep(0.3)
                limpa_tela()
                if P1_pontos == Pontos_total or PCOM_pontos == Pontos_total:
                    if P1_pontos == Pontos_total:
                        print("PARABÉNS marujo {}\nVocê conseguiu impedir a frota inimiga do ataque.\nAté a p
                    else:
                        print("VOCÊ PERDEU\nSua frota foi totalmente massacrada.")
                    verificador2 = 1
            fim_de_game = input("Deseja jogar novamente(S/N)? ") #termina/recomeça o jogo
            if fim_de_game == "S" or fim_de_game == "s":
                verificador = 1
            elif fim_de_game == "N" or fim_de_game == "n":
                print("Adeus Marujo\nFoi bom navegar com você.")
                break #quebra o while grandão
sleep(1)
limpa_tela()
```

Figura 15- Programa Principal (parte 2)

3 CONSIDERAÇÕES FINAIS

O trabalho foi uma ferramenta fundamental para o desenvolvimento do grupo na vida acadêmica. Após percorrer desafios como lidar com prazos e múltiplas tarefas, podemos concluir que a experiência como um todo nos preparou para futuras atividades semelhantes, além de ter sido uma excelente forma de testar e aplicar os conhecimentos adquiridos durante o curso de Python.

REFERÊNCIAS

[1] BIBLIOTECA OS PYTHON. Disponível em: <https://pt.stackoverflow.com/questions/170573/biblioteca-os-python>. Acessado em: 18/11/2018

[2] BIBLIOTECA TIME PYTHON. Disponível em: <https://www.tutorialspoint.com/python/time_sleep.htm>. Acessado em: 18/11/2018.

[3] DELGADO, A. L. N. *CI-240 Fundamentos de Programação*. Disponível em: <https://pt.wikiversity.org/wiki/Introdu%C3%A7%C3%A3o_%C3%A0s_Linguagens_de_Programa%C3%A7%C3%A3o/Python>. Acessado em: 23/11/2018.

[4] LINGUAGEM DE ALTO NÍVEL. Disponível em: <https://woliveiras.com.br/posts/o-que-e-linguagem-de-programacao-de-alto-nivel/>. Acessado em: 23/11/2018.

[5] LINGUAGEM DE PROGRAMAÇÃO PYTHON. Disponível em: <https://pt.wikiversity.org/wiki/Introdu%C3%A7%C3%A3o_%C3%A0s_Linguagens_de_Programa%C3%A7%C3%A3o/Python>. Acessado em: 23/11/2018.