UNIVERSIDADE FEDERAL DO PARANÁ
CURSO DE MATEMÁTICA

JULIO CESAR FAGUNDES
EDUARDO JERRI
MATEUS BALOTIN

RPG DE TEXTO EM PYTHON

Curitiba
2018

JULIO CESAR FAGUNDES GRR20185664
EDUARDO JERRI DA SILVA GRR20185689
MATEUS BALOTIN GRR20185656

RPG DE TEXTO EM PYTHON

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

Orientador: Prof. Jackson Antônio do Prado Lima

Curitiba, novembro de 2018

SUMÁRIO

1. INTRODUÇÃO

Este relatório é relativo ao trabalho realizado em Python sobre RPG de texto, realizado em trio.

RPG é a sigla para Role-Playing Game, que significa Jogo de interpretação de papel, ou seja, o jogador interpreta um personagem enquanto joga. O RPG é um jogo que acontece em tempo real, portanto o jogador toma decisões e essas decisões têm consequencias e podem desencadear uma série de eventos (ou não).

O RPG apareceu pela primeira oficialmente em 1974 com o jogo Dungeons and Dragons, também conhecido como D&D e desde então anda ganhando muito espaço no mundo dos jogos.

2. OBJETIVOS

2.1 OBJETIVO GERAL

O objetivo geral é obter um programa interativo com o usuário, onde ele possa tomar decisões em situações dadas pelo programa e dessas decisões terão consequências que irão definir o decorrer o RPG.

3 DESENVOLVIMENTO

3.1 Definição de Classes e funções

O programa conta com 12 classes e 4 funções (fora as definidas dentro de classes). A primeira classe é um dado, que será utilizado dentro das classes e funções com diversos fins.

```
1  from random import randint
13 class Dado:
14     #Cria a classe Dado, define a função rolar com os parâmetros x, lados e retorna um valor entre x e o número lados.
15     def rolar(x,lados):
16         return randint(x,lados)
17
18
```

Figura 1 – Classe Dado

A segunda classe é onde definimos Personagem, essa classe contém nome, saude e exp (pontos de experiência). Essa classe é mais uma classe de apoio, durante o programa ela é chamada dentro de outras classes utilizando super().

```
19 class Personagem:
20     def __init__(self,nome,saude,exp):
21         self.nome = nome
22         self.saude = saude
23         self.exp = exp
24     ''' Cria a classe Personagem, que será utilizada em mobs e no personagem principal. A classe personagem contém nome,
25     saude e pontos de experiência do mob ou jogador. def _init_ inicia toda vez que um objeto da classe é criado e os
26     parâmetros (self,nome,hp,exp).'''
27
```

Figura 2 – Classe Personagem

Logo a seguir definimos as classes com as quais o usuário poderá jogar, elas são Guerreiro e Mago. Utilizamos a função super() para essas classes herdarem as definições da classe Personagem, e __init__ para isso ocorrer ao iniciar o programa. Dentro das classes Mago e Guerreiro foram definidos também os parâmetros inicias de cada classe e as suas habilidades.

```
class Guerreiro(Personagem):
    #Aqui é definida a classe Guerreiro, que é umas das possíveis escolhas para o jogador.
    def __init__(self):
        super().__init__(nome =input("Você se tornou um guerreiro que não teme a morte, que nome você dirá à fada? "), saude=18, ex
    prof = "guerreiro"
    max_saude = 15
    nivel = 1
    dano_max = Dado.rolar(2,15)
    ganho_vida = 3
    nivel2 = 20 #experiência necessária para passar de nível
    ''' Aqui definimos a saude (pontos de vida) inicial Guerreiro como 15 por padrão. O dano inicial dele foi definido como uma
    rolagem de dado que
    varia de 2 a 15. O personagem começa no nível 1 e conforme seu nível nível aumenta também há um ganho de vida.'''


    def lutar(self):
        lutar()

    def bankai(self):

        heroi.saude -= 4
        if heroi.saude > 0:
            rollD = Dado.rolar(15,25)
            print("\nVocê usou bankai causando", rollD, "de dano")
            mob.saude -= rollD #vida que o monstro perde
            print("O", mob.nome, "esta com", mob.saude,"de vida")
        else:
            print("Você morreu!")


class Mago(Personagem):
    #Aqui é definida a classe Mago, a segunda escolha possível para o jogador.
    def __init__(self):
        super().__init__(nome =input("Você se tornou um mago que busca o conhecimento incessantemente, que nome você dirá à fada? ")
    prof = "mago"
    mana = 1
    max_mana = 1
    max_saude = 15
    nivel = 1
    dano_max = Dado.rolar(1,10)
    ganho_vida = 2
    nivel2 = 12

    '''
    Assim como no Guerreiro, aqui na Classe do Mago foram definidos os pontos de vida iniciais, o dano inicial e o nível que o
    jogador começa. Mas o ganho de vida por nível foi reduzido e a experiência para aumentar seu nível também. Outra diferença é
    que colocamos um parâmetro mana para o Mago poder utilizar magias (Será visto logo em seguida)
    '''

    def lutar(self):
        lutar()

    def missil_magico(self):
        if heroi.mana > 0:
            rollD = Dado.rolar(5,12)
            Mago.mana -= 1
            print("\nVocê usou Mísssil mágico causando", rollD, "de dano")
            mob.saude -= rollD #vida que o monstro perde
            print("O", mob.nome, "esta com", mob.saude,"de vida")
        else:
            print("Mana insuficiente")

```

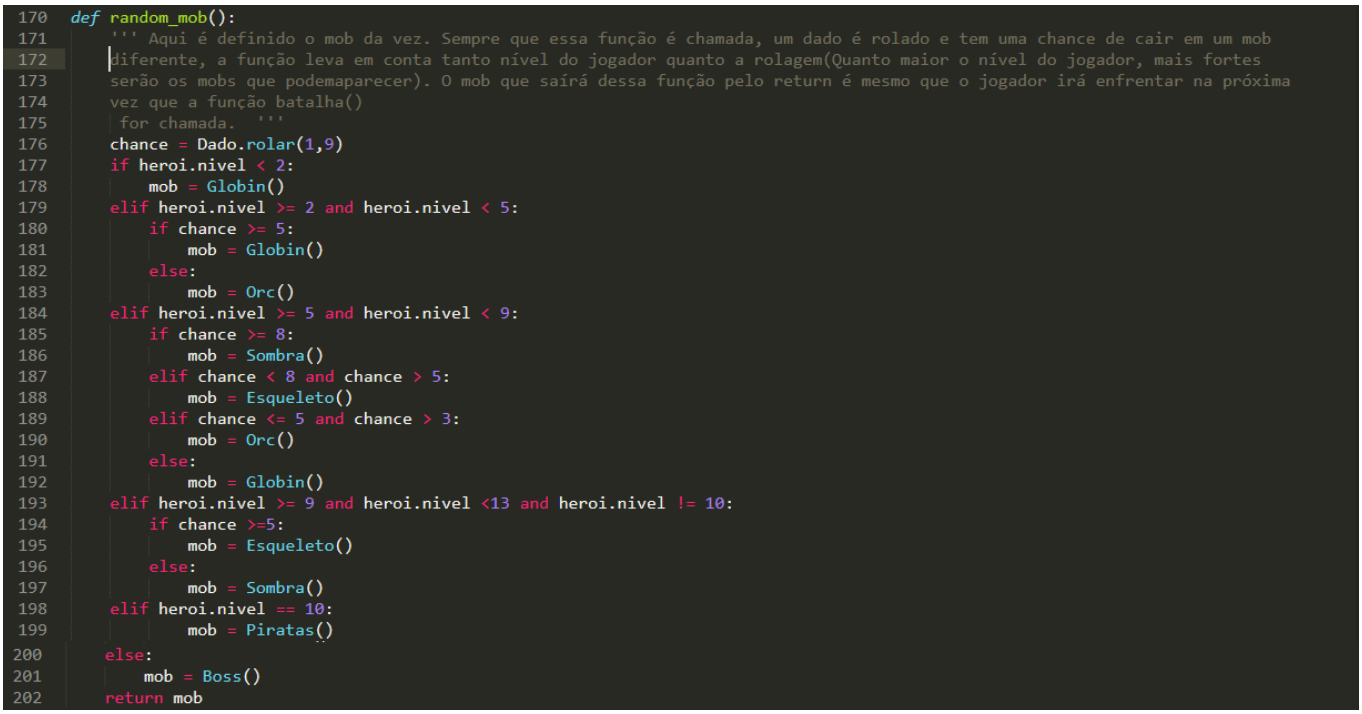
Figura 3 – Classes Guerreiro e Mago (3 e 4)

Da quinta à décima classe foram definidos os mobs(monstros) do jogo. As classes de mob também herdam a classe Personagem e tem parâmetros próprios de vida dano, além de (alguns mobs) terem habilidades únicas.

```
127 class Boss(Personagem):
128     ''' Classe Boss, herdando os parâmetros da classe Personagem, com super(). Definimos o dano base como uma rolagem de dados
129     entre 4 e 15 e a habilidade choque_do_trovao como uma rolagem entre 5 e 18, que serão as habilidades que o Boss poderá utilizar
130     em batalha.'''
131     def __init__(self):
132             super().__init__(nome = "Pikachu", saude = 40, exp = 40)
133     dano_max = Dado.rolar(4,15)
134     choque_do_trovao = Dado.rolar(5,18)
135 
136 class Globin(Personagem):
137     '''Classe Globin, herdando os parâmetros da classe Personagem, com super(). Definimos seu dano base como uma rolagem entre
138     1 e 6.'''
139     def __init__(self):
140             super().__init__(nome = "Globin", saude = 8, exp = 6)
141     dano_max = Dado.rolar(1,6)
142 
143 
144 class Orc(Personagem):
145     ''' Classe Orc, herdando os parâmetros da classe Personagem, com super(). Definimos seu dano base como uma rolagem entre
146     3 e 8.'''
147     def __init__(self):
148             super().__init__(nome = "Orc", saude = 10, exp = 8)
149     dano_max = Dado.rolar(3,8)
150 class Sombra(Personagem):
151     ''' Classe Sombra, herdando os parâmetros da classe Personagem, com super(). Definimos seu dano base como uma rolagem entre
152     2 e 10, e também uma magia com o dano sendo uma rolagem entre 5 e 12.'''
153     def __init__(self):
154             super().__init__(nome = "Sombra", saude = 20, exp = 15)
155     dano_max = Dado.rolar(2,10)
156     magia = Dado.rolar(5,12)
157 
158 class Esqueleto(Personagem):
159     ''' Classe Esqueleto, herdando os parâmetros da classe Personagem, com super(). Definimos seu dano base como uma rolagem entre
160      3 e 15'''
161     def __init__(self):
162             super().__init__(nome = "Esqueleto", saude = 20, exp = 15)
163     dano_max = Dado.rolar(3,15)
164 
165 class Piratas(Personagem):
166     ''' Classe Piratas, herdando os parâmetros da classe Personagem, com super(). Definimos seu dano base como uma rolagem entre
167      4 e 11'''
168     def __init__(self):
169             super().__init__(nome = 'Piratas', saude = 25, exp = 25)
170     dano_max = Dado.rolar(4,11)
```

Figura 4 – Classes de mobs

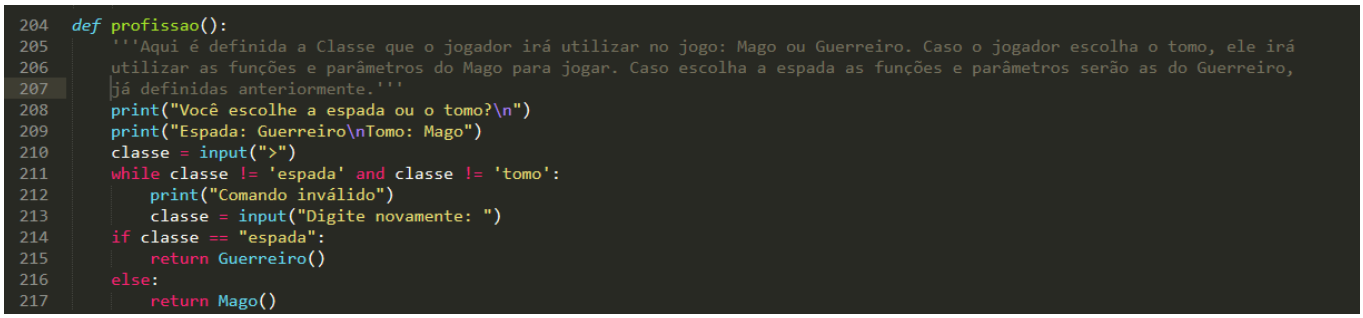
A primeira função (Fora das classes) é a random_mob, onde nós usamos o dado e condições para definir qual será o mob que irá sair da função.



```
def random_mob():
    ''' Aqui é definido o mob da vez. Sempre que essa função é chamada, um dado é rolado e tem uma chance de cair em um mob
    diferente, a função leva em conta tanto nível do jogador quanto a rolagem(Quanto maior o nível do jogador, mais fortes
    serão os mobs que podemaparecer). O mob que saírá dessa função pelo return é mesmo que o jogador irá enfrentar na próxima
    vez que a função batalha()
     for chamada.  '''
    chance = Dado.rolar(1,9)
    if heroi.nivel < 2:
        mob = Globin()
    elif heroi.nivel >= 2 and heroi.nivel < 5:
        if chance >= 5:
            mob = Globin()
        else:
            mob = Orc()
    elif heroi.nivel >= 5 and heroi.nivel < 9:
        if chance >= 8:
            mob = Sombra()
        elif chance < 8 and chance > 5:
            mob = Esqueleto()
        elif chance <= 5 and chance > 3:
            mob = Orc()
        else:
            mob = Globin()
    elif heroi.nivel >= 9 and heroi.nivel <13 and heroi.nivel != 10:
        if chance >=5:
            mob = Esqueleto()
        else:
            mob = Sombra()
    elif heroi.nivel == 10:
            mob = Piratas()
    else:
        mob = Boss()
    return mob
```

Figura 5 – Função random_mob

A segunda função é a profissão(). Nela o usuário escolhe se irá jogar utilizando o Mago ou o Guerreiro. O jogador entra com o texto ‘tomo’ ou ‘espada’ para mago ou guerreiro respectivamente, então por condicionais a função chama a classe correspondente.



```
def profissao():
    '''Aqui é definida a Classe que o jogador irá utilizar no jogo: Mago ou Guerreiro. Caso o jogador escolha o tomo, ele irá
    utilizar as funções e parâmetros do Mago para jogar. Caso escolha a espada as funções e parâmetros serão as do Guerreiro,
    já definidas anteriormente.'''
    print("Você escolhe a espada ou o tomo?\n")
    print("Espada: Guerreiro\nTomo: Mago")
    classe = input(">")
    while classe != 'espada' and classe != 'tomo':
        print("Comando inválido")
        classe = input("Digite novamente: ")
    if classe == "espada":
        return Guerreiro()
    else:
        return Mago()
```

Figura 6 – Função profissao

A décima primeira classe é a ataque(). Nessa classe é definido se o mob vai usar um ataque mais básico ou uma habilidade, a chance de acerto dos ataques tanto do mob quanto do jogador. Também é nessa função que o dano causado é descontado da vida do alvo. Para isso o dado e condicionais. Na definição do ataque do jogador um dado é rolado com valores entre 1 e 10, utilizando condicionais para definir que caso o valor seja maior que 5 então o jogador acertou o ataque e o programa imprimi uma mensagem, logo em seguida o dano é descontado da vida do mob. Caso contrário a função imprime a mensagem “Você errou!”.

```
219 class Ataque():
220     ''' Já definimos os parâmetros de dano e habilidades tanto do jogador quanto dos mobs. Agora falta dizer quando um mob vai
221     utilizar um ataque normal ou uma habilidade diferente e é isso que essa classe faz! Definimos a partir de rolagem de dados
222     se o mob irá utilizar um ataque mais básico ou a sua habilidade especial. Também foram utilizados dados para dizer se o ataque
223     acertou ou não, inclusive os ataques do jogador, e caso o ataque acerte ele será descontado da vida do alvo.
224         A função também conta com prints atualizando o jogador sobre acertos e erros de ataques, vida atual do alvo após ser
225     atingido e também informando quando o mob utiliza uma habilidade especial.'''
226 
227     def jogador():
228         roll = Dado.rolar(1,10)
229         if roll >= 5:
230             rollD = heroi.dano_max
231             print("\nVocê acertou com", rollD, "de dano")
232             mob.saude -= rollD
233             print("O", mob.nome, "esta com", mob.saude,"de vida")
234         else:
235             print("\nVoce errou!")
```

Figura 7 – Classe ataque, função jogador()

Ainda dentro da classe Ataque() temos a função mob(). Utilizando condicionais e rolagem de dados aqui é definido se o mob acertou o ataque e se ele usou um ataque normal ou uma habilidade. Logo em seguida, caso o mob tenha acertado o ataque, o dano causado é descontado da vida do jogador.

```
237     def mob(): # Dano dos monstros
238         roll = Dado.rolar(1,20)
239         if mob.nome == "Sombra":
240             if roll >= 10:
241                 rollD = mob.dano_max # dano aleatório
242                 print(mob.nome,"acertou com",rollD,"de dano")
243                 heroi.saude -= rollD # vida que o herói perde
244                 print(heroi.nome,"tem",heroi.saude,"de vida\n")
245             else:
246                 rollD = mob.magia
247                 print(mob.nome,"usou a habilidade sugar vida causando",rollD,"de dano")
248                 heroi.saude -= rollD
249                 print(heroi.nome,"tem",heroi.saude,"de vida\n")
250         if mob.nome == 'Boss':
251             if roll >= 12:
252                 rollD = mob.dano_max
253                 print(mob.nome,"acertou com",rollD,"de dano")
254                 heroi.saude -= rollD
255                 print(heroi.nome,"tem",heroi.saude,"de vida\n")
256             else:
257                 rollD = Boss.choque_do_trovao
258                 print(mob.nome,"usou coque_do_trovao",rollD,"de dano")
259                 heroi.saude -= rollD
260                 print(heroi.nome,"tem",heroi.saude,"de vida\n")
261         if roll >= 10 and mob.nome != 'Sombra' and mob.nome != 'Boss':
262             rollD = mob.dano_max
263             print("O",mob.nome,"acertou com",rollD,"de dano")
264             heroi.saude -= rollD
265             print(heroi.nome,"tem",heroi.saude,"de vida\n")
266         else:
267             print("O",mob.nome,"errou!\n")
```

Figura 8 – Classe ataque, função mob()

A terceira função é a Subir_nivel(). Utilizando condicionais e laços de repetição essa função define os ganhos do personagem quando ele aumenta o seu nível. Quando o valor da experiência atual do herói for maior ou igual o necessário para subir de nível, ele recupera uma parcela dos pontos de vida perdidos, tem um ganho na vida máxima (Definido conforme a classe Mago ou Guerreiro) e, caso a sua classe seja Mago, recupera uma parcela da mana, além de aumentar o total de mana que o jogador pode ter.

```
269 def Subir_nivel():
270     ''' Definimos que na classe Guerreiro e Mago como padrão que o jogador deverá começar no nível 1. Agora é a parte em que
271     definimos as vantagens de subir de nível. Quando a experiência do jogador for maior ou igual a experiência necessária para
272     personagem. Caso esteja jogando com o Mago também haverá ganho de mana.'''
273 
274     if heroi.exp >= heroi.nivel2:
275         heroi.nivel += 1
276         heroi.nivel2 = heroi.nivel2
277         while heroi.saude <= (heroi.max_saude/2):
278             heroi.saude += 1
279         heroi.max_saude += heroi.ganho_vida
280         if heroi.prof == "mago":
281             heroi.max_mana += 1
282             while heroi.mana <= (heroi.max_mana/2):
283                 heroi.mana += 1
284             print("\nVoce subiu de nivel:\nNivel: {0} --> {1}\nVida atual: {2}| Vida máxima:{3} --> {4}\nMana atual: {5} | Mana
285         else:
286             print("\nVoce subiu de nivel:\nNivel: {0} --> {1}\nVida atual: {2} | Vida máxima: {3} --> {4}".format(heroi.nivel-1,
```

Figura 9 – Função Subir_nível

A décima segunda classe é a Options(). Aqui são definidas praticamente todas as ações que o jogador pode tomar durante o jogo (Fora de combate), portanto é uma classe bastante extensa. Ela será separada em tópicos para facilitar o entendimento. As funções contidas nessa classe são:

- **ajuda():** Imprime as ações que o jogador por tomar.
- **descansar():** Utilizando uma condicional essa função recupera 1 ponto de vida caso a vida do jogador seja menor do que a vida máxima. Mas também há uma chance de um mob encontrar o seu personagem e ataca-lo. Essa chance foi definida por uma rolagem de dados entre 1 e 5.
- **habilidades():** Utilizando condicionais, imprime as habilidades de batalha do jogador conforme a classe escolhida.
- **super_mode():** É uma função para auxiliar em testes e apresentação do trabalho, ela aumenta drasticamente a vida e os parametros de dano do jogador.
- **status():** Imprime a vida, mana (caso a classe escolhida seja Mago) e nível do jogador.
- **sair():** fecha o jogo.
- **explorar():** Essa função contém todos os caminhos possíveis a se explorar e a linha de história do RPG. Aqui foram definidos também eventos “obrigatórios”, quando e onde o jogador irá entrar em batalha.

Vamos começar com os eventos obrigatórios: dentro da função explorar foram definidos 3 eventos obrigatórios através de condicionais. Um deles ocorre ao atingir o nível 5, o segundo ao atingir o nível 10 e finalmente o último no nível 13. Este último em especial é quando o jogador encontra o último desafio do jogo que é o Boss final.

Os eventos não obrigatórios são definidos através de rolagem de dados, conforme a rolagem o usuário poderá se deparar com um simples print indicando que ele explorou algum lugar até árvores de decisões com diferentes eventos dependendo do ramo que o usuário tome.

```
288 class Options(Personagem):
289     '''Aqui é definida como o corpo do jogo por assim dizer. Por ser uma classe muito extensa os comentários serão divididos por
290      função para facilitar o entendimento.'''
291     def ajuda():
292         '''Essa função é um guia que imprime as ações que o jogador pode tomar.'''
293         print(Chaves_de_comandos)
294 
295     def descansar():
296         ''' Essa função serve para recuperar a vida do jogador, mas também há uma chance (definida per rolagem de dado) de um mob
297         encontrar o jogador e o atacar, neste caso o jogador somente entra em batalha e não chega a recuperar a vida.'''
298         chance = Dado.rolar(1,5)
299         if chance <= 3:
300             if heroi.saude < heroi.max_saude:
301                 print("Você tirou uma soneca")
302                 heroi.saude += 1
303                 print("Você tem {0}/{1} de vida".format(heroi.saude, heroi.max_saude))
304             else:
305                 print("Você está com a vida máxima")
306         else:
307             print("Você foi atacado por ", mob.nome)
308             batalha()
309 
```

Figura 10 – Classe options, funções: ajuda e descansar

```
310     def habilidades():
311         ''' Imprime as habilidades de batalha do jogador conforme a classe escolhida para jogar.'''
312         if heroi.prof == "mago":
313             print('''
314 -Lutar: Desfere golpes corpo-a-corpo
315 -Missil mágico: Apesar do nome poderoso, o dano nao é lá grandes coisas
316 -Bola de fogo: Causa grande dano, porém o gasto de mana aumenta consideravelmente(nivel5)
317 -Gerar mana: Você recupera sua mana
318                 ''')
319         else:
320             print('''
321 -Lutar: Desfere golpes corpo-a-corpo
322 -Bankai: Usa a energia vital(saude) para desferir um poderoso golpe
323                 ''')
324 
325     def super_mode():
326         ''' Essa função foi definida para facilitar testes e apresentações do RPG. Ela aumenta substancialmente a vida e o dano
327         do jogador.'''
328         heroi.saude = 99
329         heroi.dano_max = Dado.rolar(99,99)
330 
331 
```

Figura 11 – Classe options, funções: habilidades e super_mode

```
333     def status():
334         ''' Imprime os status do jogador: Vida atual e máxima, mana (caso a classe escolhida seja Mago) e nível atual.'''
335         if heroi.prof == 'mago':
336             print("Vida: {0}/{1}".format(heroi.saude, heroi.max_saude))
337             print("Mana: {0}/{1}".format(heroi.mana, heroi.max_mana))
338             print("Nível: ", heroi.nivel)
339         else:
340             print("Vida: {0}/{1}".format(heroi.saude, heroi.max_saude))
341             print("Nível: ", heroi.nivel)
342 
343     def sair():
344         '''Sair do jogo'''
345         quit()
```

Figura 12 – Classe options, funções: status e sair

A função explorar por ser muito extensa terá apenas o anexo contendo uma amostra de como foi definida:

```
347     def explorar():
348         ''' Essa é a função mais extensa do programa. Aqui nós definimos todos os caminhos e eventos que o jogo oferece.
349         Quando o jogador decide explorar, há uma rolagem de dados que define o evento que irá ocorrer. O dado rola e conforme
350         o valor que cai (entre 1 e 15) temos desde eventos mais simples até eventos mais complexos que vão demandar de decisões
351         mais complicadas do usuário.
352             Alguns desses eventos têm como parâmetro para a ocontecer o nível do personagem. '''
353         lugar = Dado.rolar(1,15)
354         if heroi.nivel < 13:
355             if heroi.nivel == 5:
356                 print('''\nPassando por uma trilha na floresta, você se depara com uma cidade, quando se lembra que Valinor,
357 a cidade dos elfos, fica nessa trilha.\nLogo que você pisa na cidade um soldado elfo lhe recebe e diz que o rei estava a espera.
358 Após uma breve caminhada você chega à sala do trono e então o rei elfo lhe recebe e ofereceu um chá para lhe dar as novas.
359 Rei elfo: Chegou a informação de que o Pikachu devastou a sua cidade, lamento... Mas tenho boas novas,
360 descobri a sua localização e obtive informações cruciais para derrotarmos o monstro. O Pikachu está escondido no fim do vale
361 das trevas, atualmente com o seu nível seria impossível mata-lo mesmo com o item encantado da fada. Mas
362 se você aumentar a sua experiência em batalhas e alcançar o nível 13 você deverá ser capaz de salvar a humanidade! Como prova
363 de minha boa vontade lhe entrego este pergaminho de sabedoria, ele melhorará as suas habilidades em batalha.
364 \n -O acessor do Rei aparece na sala e o convoca para um reunião urgente, assim você segue jornada.''')
365                 heroi.nivel += 1
366             elif heroi.nivel == 10:
367                 print('''
368 Você chegou à cidade de Fiore, uma cidade bastante perigosa e habitada, principalmente, por humanos. Aqui piratas contrabandeiam
369 seus produtos roubados e os cidadãos vivem com medo e à mercê de bandidos. Quando você passa os piratas logo veem o item mágico
370 que a fada lhe deu. Você acelera o passo e segue o seu rumo.
371 \n...\n
372 Algum tempo depois, já próximo a saída da cidade, os piratas te emboscam e pedem o seu item mágico. Você se rende e entrega ou luta?
373 ~~Digite: "lutar" ou "entregar"
374  ''')
375                 comando = input(">")
376                 while comando != "lutar" and comando != "entregar":
377                     print("Comando inválido!")
378                     comando = input(">")
379                 if comando == 'entregar':
380                     heroi.saude = 0
381                     print("Os piratas fugiram rapidamente com o item e você não terá como recuperá-lo. Sem ele você não poderá derro
382                 else:
383                     batalha()
```

Figura 13 – classe options, função explorar

A última função antes de o jogo de fato começar é a função batalha(). Essa função lê os comandos do jogador para a luta e em seguida chama as funções correspondentes. Através de condicionais são definidos turnos (primeiro a ação do jogador e em seguida a do mob) e caso alguém morra serão impressas as mensagens correspondentes. Caso o jogador mate o mob a experiência do mob é adicionada à do jogador e, caso tenha experiência suficiente o jogador pode subir de nível pela função Subir_nivel().

```
651 def batalha():
652     ''' Aqui é definido é definido como a batalha irá ocorrer. Essa função lê o comando que o usuário deu na batalha e o executa,
653     após executar o comando do usuário é o turno do do mob. Caso o mob morra a sua experiência é adicionada à experiência do herói
654     e então é chamada a função Subir_nivel(). Caso a vida do herói seja menor ou igual a zero a função imprime uma mensagem.'''
655     while heroi.saude > 0 and mob.saude > 0:
656         for comando_2, action in heroi.Comandos.items():
657             ''' lê o dicionário Comandos e atribui as variaveis comandos_2 e action seus valores respectivos.'''
658             print('Pressione {} para {}'.format(comando_2, action[0]))
659         comando_2 = input(">")
660         while comando_2 not in heroi.Comandos and comando_2 != 'status':
661             print("Comando não válido, digite novamente!")
662             comando_2 = input(">")
663 
664         if heroi.saude > 0:
665             if comando_2 == '1':
666                 Ataque.jogador()
667                 if mob.saude > 0:
668                     Ataque.mob()
669             elif comando_2  == 'm':
670                 heroi.missil_magico()
671                 if mob.saude > 0:
672                     Ataque.mob()
673             elif comando_2 == 'b':
674                 heroi.bankai()
675                 if mob.saude > 0:
676                     Ataque.mob()
677             elif comando_2 =='f':
678                 heroi.bola_de_fogo()
679                 if mob.saude > 0:
680                     Ataque.mob()
681             elif comando 2 =='status':
682                 Options.status()
683             else:
684                 heroi.gerar_mana()
685                 Ataque.mob()
686 
687 
688     if heroi.saude <= 0:
689         print(heroi.nome,'morreu!')
690     else:
691         if heroi.prof == "mago": # atualiza o nivel, e se upar mostra Vida, Mana e Xp
692             print('O',mob.nome,'morreu!')
693             heroi.exp += mob.exp
694             print("\nVida {0}/{1}\nMana {2}/{3}\nXP {4}".format(heroi.saude,heroi.max_saude,heroi.mana,heroi.max_mana,heroi.exp))
695             Subir_nivel()
696         else:
697             print('O',mob.nome,'morreu!') # Atualiza o nivel e se upar mostra Vida e Xp
698             heroi.exp += mob.exp
699             print("\nVida {0}/{1}\nXP {2}".format(heroi.saude,heroi.max_saude,heroi.exp))
700             Subir_nivel()
```

Figura 14 – Função batalha()

3.2 Chamando as funções e classes

Agora que já definimos todas as classes e funções, o jogo começa. A primeira mensagem impressa na tela é a introdução da história, logo em seguida são chamadas as funções profissão() para determinar a classe do jogador e a função random_mob() para gerar um mob.

```
640 print('''\nSubitamente você acorda em uma caverna e logo a frente avista uma fada. Esta que lhe retribui o olhar e diz:
641 -Finalmente você acordou! Sua vila foi atacada por um Pikachu e você foi o único sobrevivente. O Pikachu é uma criatura das
642 trevas que retorna a cada 1000 anos para atormentar a humanidade. A última vez que ele retornou os humanos foram quase extintos!\n
643 -Sinto em você um tremendo potencial, será que você lutaria contra o Pikachu para salvar a humanidade? Caso aceite eu terei o
644 prazer de lhe oferecer um de meus itens mágicos, err... Qual o seu nome mesmo??\n
645 \n==>Os itens que a fada disponibilizou são uma espada e um tomo, digite:
646     ''')
647
648 heroi = profissao()
649 mob = random_mob() # Utilizamos para facilitar de chamar essas classes posteriormente.
```

Figura 15 – Introdução da história

Em seguida são impressos nome, vida, exp e mana(caso a classe escolhida tenha sido Mago) do jogador. E por fim são chamadas as funções do jogo, o jogador entra com uma ação e por condicionais o programa chama a função ou classe correspondente, dando prosseguimento ao jogo.

```
702 if heroi.prof == "mago":
703     print("\nNome: {0}, Vida: {1}, Mana: {2}, XP: {3}\n".format(heroi.nome, heroi.saude,heroi.mana,heroi.exp))
704 else:
705     print("\nNome: {0}, Vida: {1}, XP: {2}\n".format(heroi.nome, heroi.saude,heroi.exp))
706 print("Digite ajuda para ver os comandos: ")
707
708 count = 0
709 while heroi.saude > 0 and heroi.nivel < 14:
710     ''' Aqui é onde o programa chama as funções/classes. Até agora nós definimos como o jogo deve ocorrer, aqui é onde o usuário
711     decide qual função da opções vai ser chamada para o jogo poder seguir.'''
712     acao = input("\nDigite a sua ação: ")
713     mob = random_mob()
714     resultado = count%3  # limpa a tela de comando conforme certas ações sao executadas
715     count += 0.5
716     if resultado == 1:
717         os.system('cls' if os.name == 'nt' else 'clear') # limpa a tela
718     if acao == 'explorar':
719         Options.explorar()
720         count += 1
721     elif acao == 'descansar':
722         Options.descansar()
723         count += 0.5
724     elif acao == 'ajuda':
725         Options.ajuda()
726     elif acao == 'status':
727         Options.status()
728     elif acao == 'super_mode':
729         Options.super_mode()
730     elif acao == 'ganhar_xp':
731         Options.ganhar_xp()
732     elif acao == 'habilidades':
733         Options.habilidades()
734     elif acao == 'sair':
735         Options.sair()
736     else:
737         print("Ação inválida")
738
```

Figura 16 – Chamando as funções

Através de um laço de repetição, caso o personagem do usuário morra ele sempre terá a opção de jogar novamente. Dentro do laço são chamadas todas as funções e classes definidas assim como na figura acima.

```
739 if heroi.saude <= 0:
740     print("Rest in peace\n")
741     escolha = input("Deseja continuar?\nSim ou Não: ") # Pede se o jogador quer continuar ou não, se sim volta para o jogo.
742     while escolha != "sim" and escolha != "não":
743         print("Comando inválido") # Repete ate validar o comando
744         escolha = input("Digite um comando válido: ")
745     while escolha == "sim":
746         os.system('cls' if os.name == 'nt' else 'clear') # limpa a tela
747         heroi = profissao()
748         mob = random_mob()
749 
750         if heroi.prof == "mago":
751             print("\nNome:{0}, HP:{1}, Mana:{2} XP:{3}\n".format(heroi.nome, heroi.saude,heroi.mana,heroi.exp))
752         else:
753             print("\nNome:{0}, HP:{1},XP:{2}\n".format(heroi.nome, heroi.saude,heroi.exp))
754         print("Digite ajuda para ver os comandos")
755 
```

Figura 17 – Laço de repetição

O último desafio do jogador é enfrentar o Pikachu, caso o usuário perca ele entra no laço acima, caso ganhe uma mensagem de conclusão é impressa.

```
347     def explorar():
348         ''' Essa é a função mais extensa do programa. Aqui nós definimos todos os caminhos e eventos que o jogo oferece.
349         Quando o jogador decide explorar, há uma rolagem de dados que define o evento que irá ocorrer. O dado rola e conforme
350         o valor que cai (entre 1 e 15) temos desde eventos mais simples até eventos mais complexos que vão demandar de decisões
351         mais complicadas do usuário.
352             Alguns desses eventos têm como parâmetro para a ocontecer o nível do personagem. '''
353         lugar = Dado.rolar(1,15)
354         if heroi.nivel < 13:
636         else:
637             print("Um Pikachu selvagem apareceu!\n")
638             batalha()
794     print('''
795         Parabéns! Você salvou a humanidade por mais 1000 anos, agora todos reconhecem {0} como um grande herói e seu nome será lembrado para sempre como
796         um dos grandes campeões a derrotar o poderoso Pikachu. Essa história teve um final feliz, esperamos que nas próximas reencarnações de Pikachu
797         ajam heróis tão destemidos quanto você para proteger a humanidade, pois estando preparados ou não,  Pikachu continuará com seu insaciável desejo por
798         sangue...
799         '''.format(heroi.nome))
```

Figura 18 – Conclusão

4 CONCLUSÃO

Foi um tabalho bastante complicado, pois exigia conhecimento de algumas funções que não tivemos oportunidade de ver em sala de aula, mas por termos escolhido um tema o qual é de interesse de todos os integrantes do grupo nós nos esforçamos e no fim conseguimos. Apesar do grande tempo gasto no trabalho, como foi um tópico que todos gostavam foi algo prazeroso de se fazer, apesar de alguns momentos estressantes. Mas embora o prazo menor do que gostaríamos, nós aprendemos muito e estamos satisfeitos com o resultado deste trabalho.