JONAS DA SILVA MELO

Turma: 9025

PROJETO FINAL DE COMPUTAÇÃO I:

Jogo da Forca com Python

Trabalho final apresentado como requisito parcial de aprovação na disciplina de Computação I, ofertada ao curso de Engenharia Elétrica da Escola Politécnica da Universidade Federal do Rio de Janeiro, ministrada pelo professor doutor Giomar Oliver Sequeiros Olivera.

RIO DE JANEIRO

2022

1 INTRODUÇÃO

Esse projeto consiste na escrita de um código na linguagem de programação Python para simular o tradicional Jogo da Forca, onde o jogador tenta advinha uma palavra antes que seus erros completem a forma de um corpo humano na forca. O jogo, portanto, consistem em alguns elementos essenciais:

1. Palavra secreta: a essência do jogo é tentar adivinhar uma palavra que é oculta ao jogador.
2. Máscara: a palavra secreta é substituída por uma máscara, que tem como atributo apenas a quantidade de letras da palavra representadas por traços-baixos.
3. Chutes: para que a palavra seja descoberta, o jogador tem direito a arriscar as letras que compõe a palavra, uma por vez, que passam a ocupar sua posição na máscara, caso o chute esteja correto.
4. Cemitério: caso o chute esteja errado, a letra tentada passa a ocupar um espaço específico para que o jogador não venha tentá-la novamente.
5. Forca: da mesma forma, no caso de o chute estar incorreto, o erro é representado como uma parte do corpo que está pendurado na forca.
6. Vitória ou derrota: o jogador ganha o jogo ao acertar todas as letras que compõe a palavra, e perde o jogo ao ter todos os membros do corpo desenhados na forca, resultando no enforcamento do personagem.

Nos requisitos do projeto foi necessário representar esses elementos graficamente por meio módulo *turtle*, conhecido popularmente na linguagem Python por ser uma versão pré-instalada análoga aos gráficos de tartaruga introduzidos na década de 1960 por Cynthia Solomon e Seymour Papert na linguagem Logo, assim como uma forma acessível de aprender os fundamentos de Python (SOLOMON, PAPERT, 1976).

2 PROGRAMA

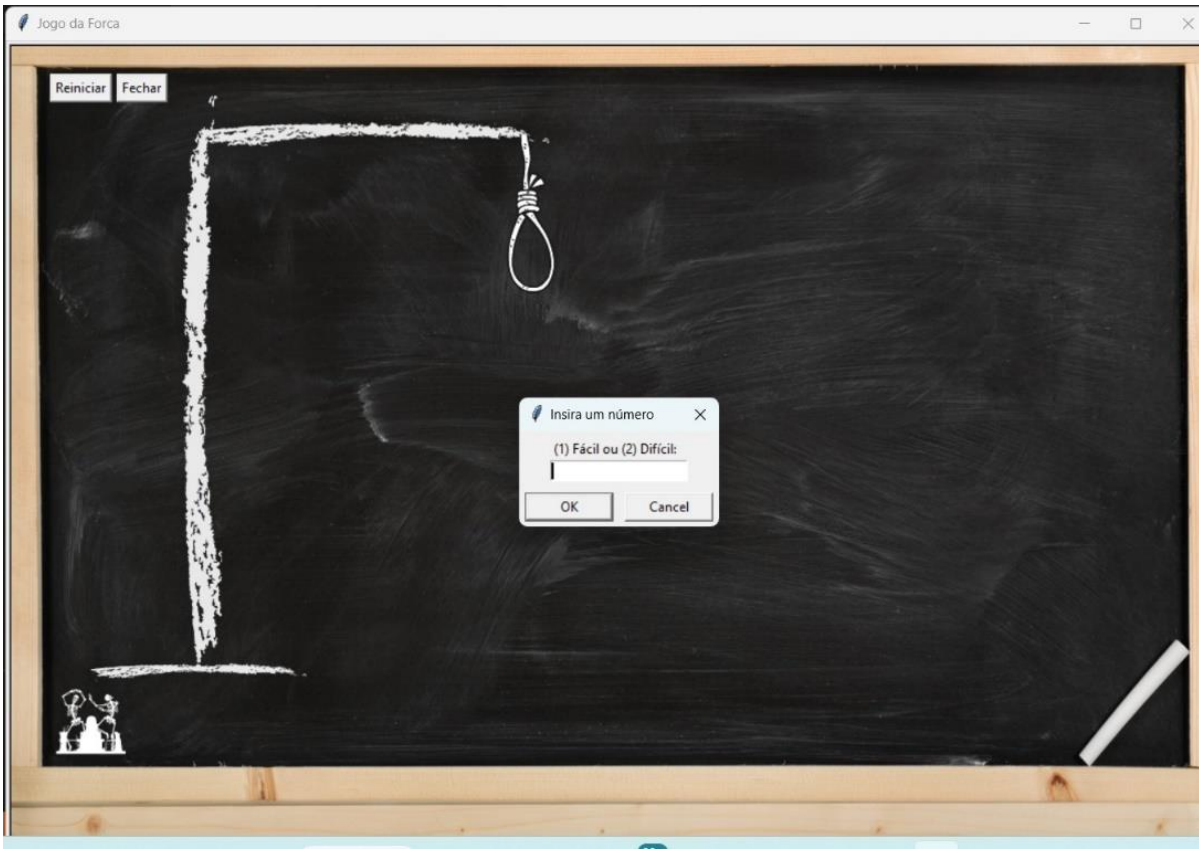
O programa foi desenvolvido por meio da prática de modularização e no padrão de arquitetura em camadas. Ou seja, os elementos do jogo foram traduzidos em funções separadas, cada qual com funcionalidades isoladas e dependentes apenas de variáveis globais, acessíveis apenas pela função principal – *main*.

A função *main* (linhas 13 à 54) é responsável, portanto, de declarar a variáveis globais e chamar as outras funções que implementam o jogo. Iniciamos com as

configurações da interface gráfica do usuário (GUI). O *turtle* utiliza o pacote de ferramentas de GUI padrão do Python, o Tkinter. Funciona basicamente como um gráfico cartesiano que utiliza posições vetoriais para desenhar na tela. A função *iniciar_turtle* (linhas 56 à 79) é chamada primeiramente para realizar essa tarefa. E com auxílio de uma das classes de Tkinter, insere dois botões na tela, um para reiniciar o jogo, e outro para encerrar a execução do programa.

Para que fosse possível algumas das funcionalidades como atualizar o número de vidas e editar o corpo desenhado na forca, foi necessário o paradigma de orientação a objetos, utilizando-se da classe Turtle para criar objetos que seriam utilizados pelas funções do jogo. Após a criação desses objetos (linhas 19 à 21), é iniciado a interação com o usuário por meio da função *escolher_modalidade* (linhas 81 à 95), no qual o usuário insere o dígito “1” ou “2” para escolher entre a modalidade fácil e difícil. A Figura 1 **Erro! Fonte de referência não encontrada.**apresenta o que aparece inicialmente na tela.

Figura 1 – Tela inicial



Após a escolha da modalidade, a função *escolher_palavra* (linhas 97 à 112) utiliza a preferência do usuário para escolher pseudo-aleatoriamente a palavra secreta

a partir de um arquivo JSON onde é armazenado uma lista de palavras separadas em chaves que representam as modalidades “Fácil” e “Difícil”, onde se encontram objetos que representam grupos de palavras com valores do tipo *Array*. Para isso, a função faz uso dos módulos *json* e *random*, também pré-instalados no Python.

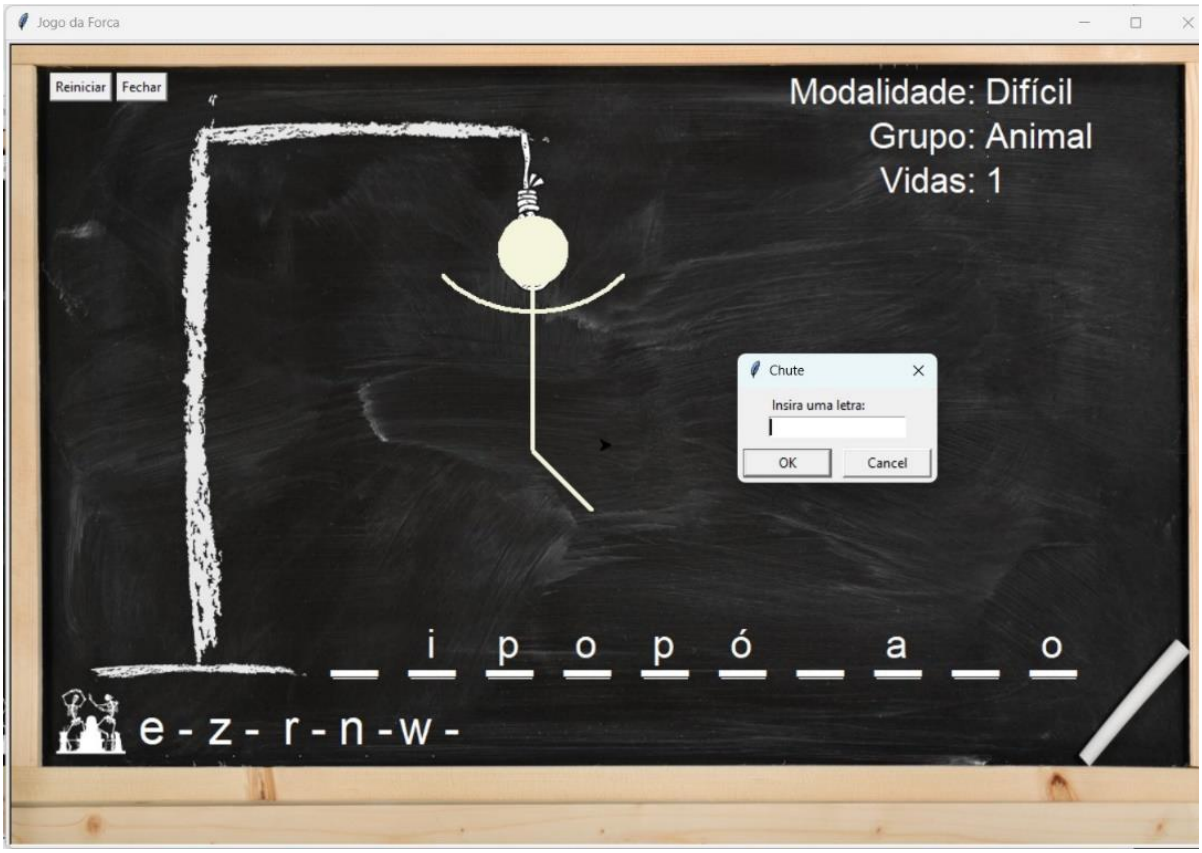
Com a palavra armazenada e a variável *mascara* declarada (linha 25 e 26), são chamadas as funções *atualizar_vidas* (linhas 114 à 125) e *escrever_mascara* (linhas 127 à 140) que finalizam as configurações iniciais da tela para que o jogo possa começar. A Figura 2 mostra como fica a tela.

Figura 2 – Começo do jogo

O jogo é representado no código por meio de um *loop*, ou seja, uma sequência de comandos que são seguidos repetidamente até que o jogador conclua o jogo, seja perdendo ou ganhando. Isso está contido na estrutura de repetição *while* (linhas 36 à 49), onde a função *atualizar_mascara* (linhas 151 à 179) solicitará a entrada de uma letra e verificará se a entrada está presente na palavra “sorteada”, escrevendo-a sobre os traços-baixos caso esteja correta, ou chamando a função *cemiterio* (linhas 287 à 295) para escrever a letra abaixo e a função *desenhar_boneco* (linhas 181 à 285) para

desenhar mais uma parte do corpo humano na forca caso esteja incorreta. Nessa etapa, a mostra como fica a tela do usuário.

Figura 3 - Após entradas do usuário



As condições para saída da estrutura de repetição *while* (linha 36) são a perda das seis vidas (quando a variável *n_vidas* é igual a zero) ou todas as letras da palavra foram adivinhadas (quando a variável *mascara* contém as mesmas letras da variável *palavra*). Conforme a segunda lei de Morgan, a negação da disjunção é equivalente a conjunção das negações. Portanto, usamos o operador lógico *and* para conjunção das negações das condições apresentadas. A função *perdeu_ou_ganhou* (linhas 297 à 309) avalia novamente as variáveis para informar o usuário da saída do loop. As e apresentam a últimas telas do jogo.

Após o encerramento do jogo, a tela permanece em *loop* (linha 54) até que o usuário decida encerrar a execução do programa fechando a janela ou clicando no botão Fechar.

Figura 4 - Tela quando o jogador perde o jogo

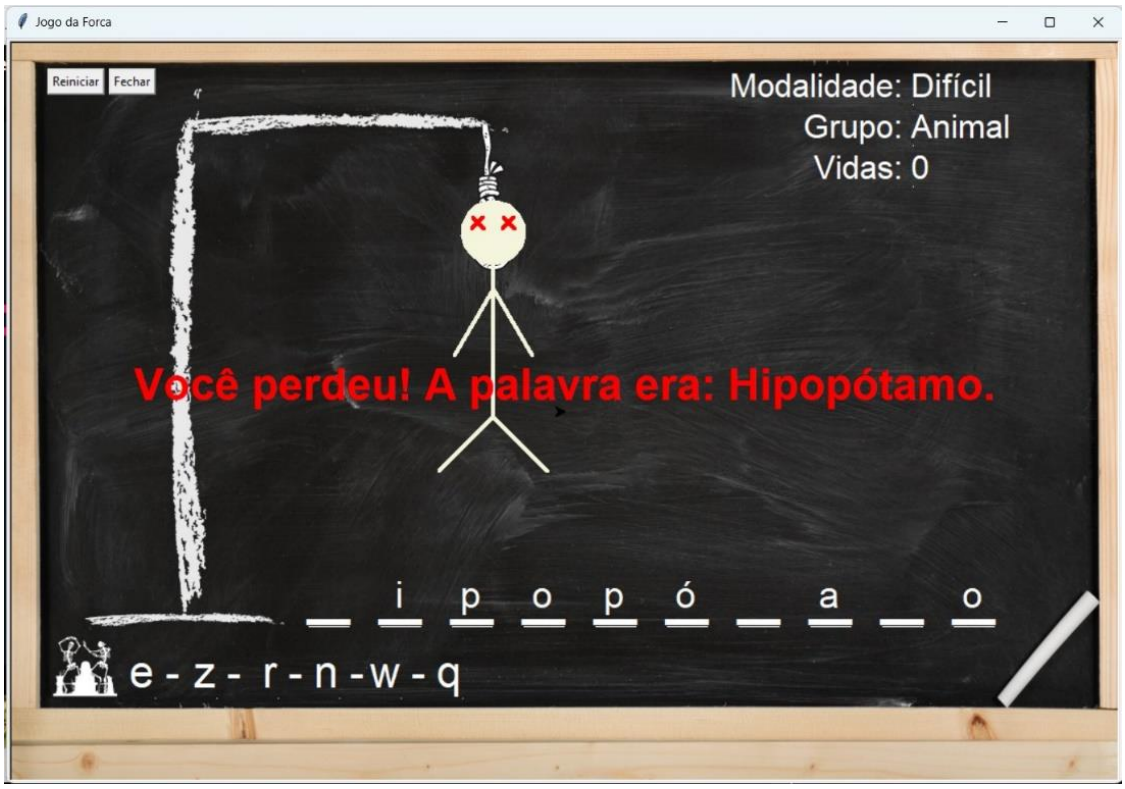
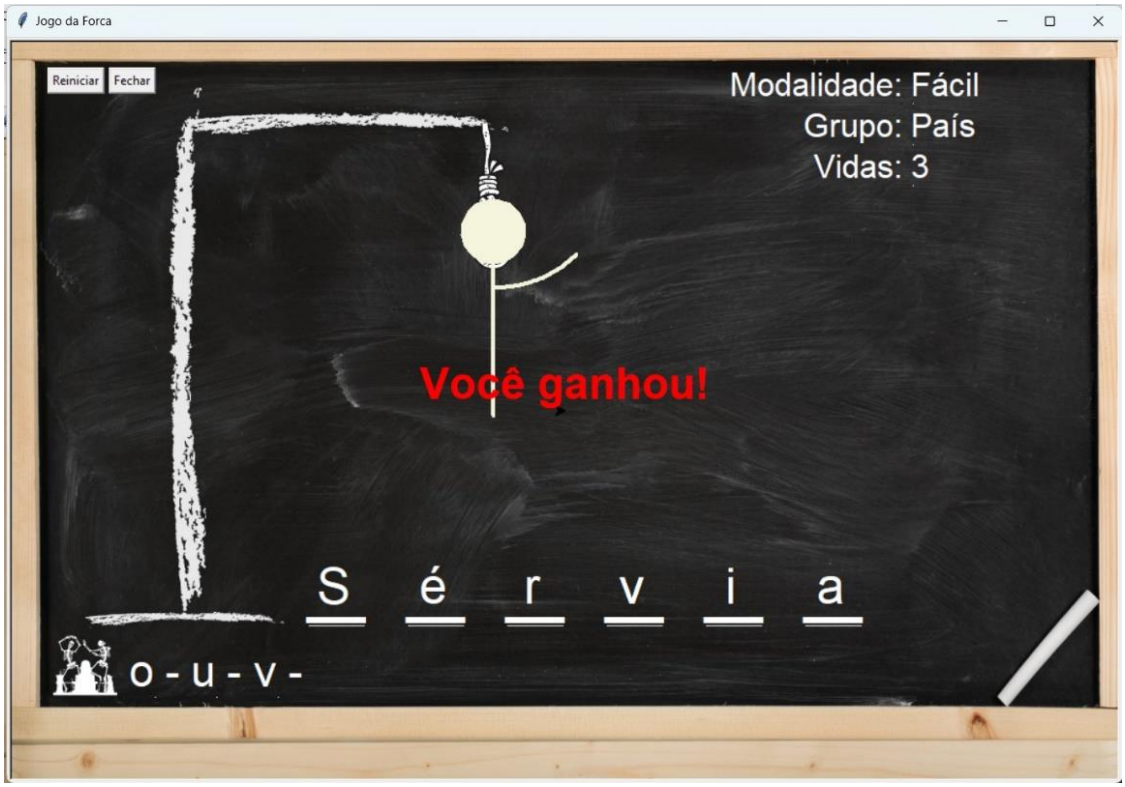


Figura 5 - Tela quando o jogador ganha o jogo



3 DESAFIOS

Os problemas encontrados durante a escrita do código são elencados abaixo:

1. Normalização das entradas do usuário para ignorar acentos e cedilha;
2. Usar o método *clear* para redesenhar o boneco enforcado;
3. Posicionamento e tamanho dos traços-baixos e letras conforme extensão da palavra;
4. Armazenamento das palavras em banco de dados.

O primeiro problema encontrado foi solucionado a partir do estudo no módulo *unicodedata*, pré-instalado no Python, assim como o estudo de conceitos e práticas de Unicode na linguagem de programação Python (LUTZ, 2013; RAMALHO, 2015).

O segundo problema foi solucionado introduzindo princípios do paradigma de orientação a objetos, onde os desenhos do Turtle foram atribuídos à uma variável que posteriormente poderia ser redesenhada após o uso do método *clear*.

O terceiro problema encontrado foi resolvido utilizando funções de primeiro grau, onde o coeficiente linear é o tamanho máximo condizente com o tamanho da tela, e o coeficiente angular é um fator razoável para que a extensão da palavra seja bem representada na tela.

O último problema era a presença de uma extensa lista de palavras no código, o que o deixava mais poluído e difícil de ler. A solução foi externar o banco de palavras para um arquivo JSON. Para tanto, foi utilizado o módulo pré-instalado do Python para trabalhar com esse tipo de arquivo.

4 CONCLUSÃO

A experiência de construir o programa permitiu encontrar desafios que só seriam superados a partir de um aprofundamento no conhecimento da linguagem de programação. Apesar de haver diversas funcionalidades que foram inseridas a partir do aperfeiçoamento do código, há algumas considerações que podem ser feitas. O arquivo JSON apresentou alguns problemas para codificar *strings* em UTF-8, o que poderia ser contornado possivelmente pela adoção de outro formado como XML ou CSV. Outra questão é que a função *cemiterio* poderia ser, ao invés de uma função, um bloco *else* na estrutura de seleção entre as linhas 162 e 176.

REFERÊNCIAS

LUTZ, M. **Learning Python**. 5. ed. Beijing, O'Reilly, 2013. Disponível em: https://www.safaribooksonline.com/library/view/learning-python-5th/9781449355722/.

RAMALHO, L. **Fluent Python: Clear, Concise, and Effective Programming**. [S.l.], O'Reilly Media, 2015. Disponível em: https://books.google.co.in/books?id=bIZHCgAAQBAJ.

SOLOMON, C. J., PAPERT, S. "A Case Study of a Young Child Doing Turtle Graphics in LOGO". 1976. **Anais** [...] New York, NY, USA, Association for Computing Machinery, 1976. p. 1049–1056. DOI: 10.1145/1499799.1499945. Disponível em: https://doi.org/10.1145/1499799.1499945.