Tabuleiro de números

Antonio Aparecido Medeiros Santana

[1] UEFS – Universidade Estadual de Feira de Santana-Departamento de Ciências Exatas
Av. Transnordestina, s/n, Novo Horizonte
Feira de Santana – BA, Brasil – 44036-900.

Resumo. *Em busca de novos integrantes do curso de Engenharia de Computação para a liga de jogos, o ramo do Instituto de Engenheiros Elétricos e Eletrônicos da Universidade Estadual de Feira de Santana, realizou um desafio aos estudantes interesados: a criação de um jogo de tabuleiro com regras e requisitos pré estabelecidos que devem ser seguidos. Foi com base nesse desafio que o programa presente neste relatório foi criado. Seu desenvolvimento se deu através da linguagem de programação Python no ambiente de desenvolvimento integrado Visual Studio Code. O presente jogo mostrou-se dentro de todos os requisitos e expectativas que lhe foram impostos e é um exemplo de trabalho concluído com sucesso.*

1. Introdução

Segundo [Amorim 2006], como citado em [Batista 2018], o primeiro jogo eletrônico da história surgiu em 1958, criado pelo físico Willy Higinbotham e de nome Tennis Programing. Tratava-se de um jogo simples que era jogado por meio de um osciloscópio. Da criação do primeiro jogo eletrônico até os dias atuais houve um salto enorme de tecnologia e desenvolvimento, no qual os jogos modernos estão cada vez mais realistas e complexos de serem desenvolvidos. Segundo a pesquisa da indústria brasileira de games, realizada pela Abragames[ABRAGAMES 2023], se consolidando como o maior mercado de games da América Latina e 5° maior em população online, o Brasil possui grande impacto na indústria mundial de jogos eletrônicos. Foi pensando nessas perspectivas que a Liga de Jogos do ramo do Instituto de Engenheiros Elétricos e Eletrônicos(IEEE) da Universidade Estadual de Feira de Santana(UEFS), decidiu lançar um desafio de desenvolvimento de um jogo eletrônico de tabuleiro para os ingressantes do curso de engenharia de computação da UEFS.

2. metodologia

O jogo foi desenvolvido na linguagem de programação Python 3.12 no ambiente de desenvolvimento integrado Visual Studio Code, tendo como sistema operacional o Windows 10-pro. O arquivo do código foi dividido em partes para maior otimização e fácil manutenção, são elas:

2.1. Código principal

No código principal, inicialmente o módulo de funções é importado, logo em seguida um variável de controle intitulada “continua” é criada recebendo o valor *True*. Após isso, um *loop while* é criado com a variável, enquanto verdade, o *loop* continuará a rodar. Dentro do *while* é onde o jogo acontece, a função “limpar” é executada, após isso a função “titulo” é

executada com o valor “BEM VINDO AO TABULEIRO DE NÚMEROS” seguida da “tocarSomMenus”, “pausar” e “limpar”. Depois dessa introdução do nome e som do jogo, a função “menuPrinc” é executada e a variável “choiceFirstMenu” recebe o valor da função “verificacao”. O fluxo do código segue para um bloco *if..elif...else* onde é avaliado o valor da variável “choiceFirstMenu”, se o valor for “1” o código executa as seguintes funções em sequência: “limpar”, “createMenus” com o parâmetro uma lista com as dificuldades dos jogos, um variável “dificuldade” recebe o retorna na função “verificacao”, “limpar”, “newGameMenu” e por fim “inciarJogo”, com os parâmetros dificuldade e o retorna da “verificacao”.

Caso “choiceFirstMenu” seja “2”, apenas “limpar” e “loadOldGame” são executadas. Se for “3” for o valor de “choiceFirstMenu”, “limpar” e “mostrarPlacar” serão executadas. Se for “4” apenas “limpar” e “howToPlay” são executadas. Caso seja “5” a variável continuar recebe *False* e um continue é executado, fazendo com que o código saia do *loop while*, onde a função “titulo” é executada com o parâmetro “VOCÊ SAIU”. Se “choiceFirstMenu” for outro valor dos citados acima, o fluxo segue para o else onde uma mensagem de erro é printada e o fluxo retorna para o início do *loop* e assim por diante.

2.2. Módulos e funções

No arquivo “funcoes.py” várias funções e métodos são criados para auxiliar o programa principal. Mas antes disso uma série de bibliotecas que serão utilizadas são importadas, são elas: “os”, “tabulate”, “random”, “time”, “json” e “winsound”.

A primeira função criada é “titulo”, ela recebe uma *string* como parâmetro e printa essa *string* na formatação de um texto como visto na figura 1.

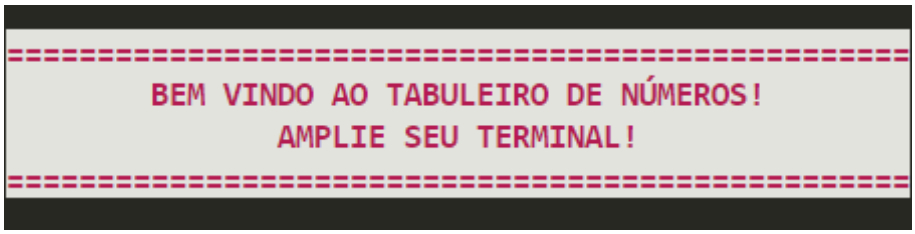

Figure 1. Funcaotitulo

A segunda função se chama “cores”, e recebe dois parâmetros, uma *string* e o nome da cor desejado, essa função retorna a *string* pintada com a cor que foi enviada. Já “formata”, recebe uma *string* igualmente e printa uma formatação para essa *string*. Na função erro, a uma mensagem de erro é mostrada no terminal com uma cor vermelha característica. As funções “tocarSomMenus”, “somFim” e “beep” vem a seguir e simplesmente emitem efeitos sonoros para fins de diferenciar os passos em que o jogo está. Adiante vem a função “limpar” que dá um clear no terminal e apaga as informações que estavam contidas nele, seguida pela “pausa” que espera um enter para continuar o fluxo de código.

Na função “howToPlay”, um pequeno texto ensinando as regras do jogo é printado no terminal. Logo após vem a criação da função “createMenus” que recebe um array e printa esse array em formato de menu. A função “menuPrincipal” em seguida printa os elementos presentes no menu principal. Agora a função “newGameMenu” é printa uma

lista de elementos presentes no menu de novo game, seguida pela função “verificacao” que recebe uma lista como parâmetro e enquanto a escolha feita pelo usuário não existir na lista a função não é retornada. A função “createTable” simplesmente recebe um tamanho e cria uma matriz baseada no valor recebido, em seguida temos a “formatedTable” que retorna a matriz formatada.

Agora, na definição dos jogadores e jogo de cada jogador é utilizada a função “sorteiaJogadorJogo”, que retorna um dicionário com o nome dos jogadores e o objetivo de cada um. Em seguida vem a função “colocaNaTabela”, que recebe a coordenada, jogadores, tabela e o valor, dentro dessa função o valor informado pelo usuário é colocado na tabela e pintado de acordo com o jogador que fez a jogada. As funções “tocarSomvictory” e “tocarSomEmpate” tocam os respectivos sons de vitória e empate, depois vem a função “iniciarJogo” que inicia realmente o jogo. Em seguida chega a função “cordenadas” que obtém uma coordenada definida pelo usuário. Seguindo vem “defineNum” que retorna uma lista com os números que podem ser utilizados pelo usuário, “pegaNum” funciona parecido porém ao invés de mostrar os números possíveis ele verifica se o número está dentro da faixa desejada. “carregaOldGame” pega o último jogo salvo e reinicia de onde ele parou, caso não exista ele retorna uma mensagem de erro. “usaPoder” recebe a linha escolhida pelo jogador e apaga todos os dados existentes. Agora chegamos na “verificaSeAcabou”, que percorre a tabela e caso ela esteja cheia para o jogo imediatamente, já “mostraPlacar” simplesmente mostra o placar existente no jogo. Para salvar o jogo utiliza-se a função “salvar” que salva o jogo em um arquivo Json que falaremos sobre ele em breve. “atualizaPlacar” pega o vencedor e joga na tabela de pontos, “sequenciaAscenDescen” verifica se os objetivos sequências ascendente e descendente foram cumpridos, o mesmo ocorre com a função “sequenciaParImp”. A função “verificaVitoria” recebe os valores das funções anteriores e se alguma for verdadeira retorna a vitória ao jogador. Por fim, temos a função “jogada”, que basicamente chama todas as funções citadas anteriormente dentro dela própria e funciona como um cérebro do programa.

2.3. Arquivo Json

O arquivo Json tem uma única função dentro do programa, receber os valores do placar e salvar algum jogo, ele não possui nenhuma função ou método especial, simplesmente é uma lista com dois dicionários, “placar” e “jogo-salvo”.

3. Resultado e discurssões

Agora faremos um pequeno manual de uso do programa. Para utilizá-ló, primeiramente é preciso fazer a instalação de bibliotecas do *Python* que são necessárias para o seu funcionamento, são elas: os, tabulate, random, time, json e winsound. Depois de fazer todas as instalações necessárias com o *pip install*, você já pode rodar o programa.

A primeira tela será uma mensagem de bem vindo, para seguir basta clicar enter. Agora você está no menu principal onde tem algumas opções: novo jogo, carregar jogo antigo, placar, aprender a jogar e sair. Você deve selecionar o número referente a escolha que você deseja. Vamos supor que sua escolha seja [1], no próximo menu você deverá escolher a dificuldade do jogo, após essa escolha você também deve escolher se o jogo terá poderes especiais ou não. O nome do jogador 1 será solicitado e logo após o nome do

jogador 2. Agora um sorteio é feito e os objetivos são distribuídos, porém, antes de exibi-los uma mensagem informando que o adversário não poderá ver seu objetivo é mostrada. Agora um sorteio para ver qual jogador vai começar é feito e o resultado é exibido, assim o jogo começa. Na parte do jogo um título informa de qual jogador é a vez, se você escolheu um jogo com poderes especiais, nesse momento é perguntado se você deseja utiliza-lo, para utilizar é necessário digitar "power" e escolher uma linha, o jogador deve escolher uma coordenada como uma batalha naval e informar qual o número que ele deseja inserir no local, porém cuidado, os números do inimigo influenciam no seu jogo. O jogo termina quando um dos objetivos forem cumpridos ou um empate ocorre.

Os game até então não apresenta erros, vários testes foram feitos como: digitar números inválidos, strings vazias, números negativos, strings em locais que deveriam ser números, porém o programa inibi o usuário de cometer quaisquer destes erros, pois na sua criação blocos de try... except foram utilizados para tal, se um erro é detectado o bloco except é acionado e pede ao usuário que informe os dados corretamente.

Entretanto, possíveis melhorias podem ser realizadas no programa, uma delas é fazer a verificação de vitória horizontalmente, pois, atualmente o game só considera vitória de você completar o objetivo horizontalmente, e na diagonal, independente do sentido. Outra possível melhoria seria utilizar bibliotecas de interface gráfica para desenvolver o jogo, pois nessas interfaces o ambiente gamificado pode trazer uma sensação de maior bem estar ao usuário.

4. Conclusão

Logo, percebe-se que todos os requisitos do produto foram entregues, o jogo realiza todos os objetivos, cria diferentes tipos de tamanho, sorteia a ordem dos jogadores e dos jogos.

Sendo assim, o objetivo do jogo se mostrou eficiente e fiel ao que foi pedido, e em futuras atualizações o game pode receber uma diversidade maior de tamanho de tabelas, mais jogadores jogando de uma vez, interface gráfica, efeitos sonoros melhorados, já que era proibido utilizar-se de arquivos extras. Assim, possíveis melhorias podem e devem ser feitas no futuro.

References

ABRAGAMES (2023). Pesquisa da indústria brasileira de games. *disponível em:https://www.abragames.org/pesquisa-da-industria-brasileira-de-games.html, acessado no dia 08/07/24 às 13:43.*

Batista, Mônica de Lourdes Souza, e. a. (2018). *Um estudo sobre a história dos jogos eletrônicos.* Revista Eletrônica da Faculdade Metodista Granbery-http://re. granbery. edu. br-ISSN (2018), 1 edition.