Joga da Vida - PBL

Yarlley Fernandes dos Santos[1]

[1]Bacharelado de Engenharia de Computação
Universidade Estadual de Feira de Santana (UEFS)
Av. Transnordestina, s/n - Feira de Santana, Novo Horizonte - BA, 44036-900

yarlleyfernandes9@gmail.com

Resumo. *Este trabalho apresenta a implementação do "Jogo da Vida", um autômato celular criado pelo matemático John Conway em 1970. O programa, desenvolvido em Python, simula a evolução de padrões celulares a partir de regras simples de sobrevivência, morte e reprodução em uma grade bidimensional. O objetivo principal é aplicar conceitos fundamentais de programação, como controle de fluxo e manipulação de matrizes, resultando em uma simulação funcional. Os resultados demonstram diferentes comportamentos de padrões a partir de configurações iniciais variadas.*

Palavras-chave: Autômato celular, Jogo da Vida, Simulação, Python.

1. Introdução

O Jogo da Vida, desenvolvido pelo matemático britânico John Conway em 1970, é um autômato celular que simula a evolução de células vivas em uma grade de células, de acordo com um conjunto de regras simples. O jogo se baseia em uma matriz bidimensional, onde cada célula pode estar "viva" ou "morta" e evolui de acordo com o número de vizinhos vivos ao redor dela. Este projeto visa implementar uma versão simplificada do Jogo da Vida em Python, utilizando uma matriz 10x10 para representar o tabuleiro e as células.

A motivação para resolver este problema reside na simulação de sistemas dinâmicos com regras simples que geram comportamentos complexos. A solução envolve a criação de uma matriz para representar o tabuleiro, a interação com o usuário para definir as células iniciais e a aplicação das regras de evolução das células, com atualização contínua da matriz até que todas as células morram.

2. Metodologia

2.1. Definição dos Requisitos

O programa foi desenvolvido para simular o Jogo da Vida utilizando uma matriz 10x10, onde as células começam mortas (representadas por `.`) e podem ser transformadas em vivas (representadas por `V`) conforme a interação do usuário. O usuário pode escolher o número inicial de células vivas e suas posições na matriz. O sistema, em seguida, executa a evolução das células de acordo com as regras do jogo, que são aplicadas a cada geração.

2.2. Descrição de Alto Nível do Algoritmo

O algoritmo segue a abordagem descrita anteriormente. O programa pode ser dividido nas seguintes partes principais:

1. **Criação da Matriz:** A função `criar_matriz_10x10` inicializa uma matriz 10x10, onde todas as células começam com o valor `.` (representando células mortas).
2. **Entrada do Usuário:** O usuário escolhe quantas células vivas ele deseja no tabuleiro e as localizações dessas células na matriz. As entradas são validadas para garantir que as escolhas estão dentro dos limites da matriz.
3. **Exibição da Matriz Inicial:** Após a configuração inicial das células vivas, a função `mostra_matriz_inicial` exibe a configuração inicial com as células vivas marcadas por `V`.
4. **Cálculo e Atualização da Matriz:** A cada iteração, o programa calcula o próximo estado de cada célula com base nos vizinhos. Para isso, ele conta os vizinhos vivos em torno de cada célula e aplica as regras:
    - Se uma célula viva tem menos de 2 vizinhos vivos, ela morre por subpopulação.
    - Se uma célula viva tem 2 ou 3 vizinhos vivos, ela sobrevive.
    - Se uma célula viva tem mais de 3 vizinhos vivos, ela morre por superpopulação.
    - Se uma célula morta tem exatamente 3 vizinhos vivos, ela se torna viva por reprodução.
5. **Exibição da Matriz Atualizada:** A cada geração, a função `mostra_matriz_atualizada` exibe a nova configuração da matriz com as células vivas e mortas. O jogo continua rodando até que todas as células estejam mortas, momento em que o programa exibe uma mensagem e encerra a execução automaticamente.
6. **Pausa e Limpeza:** Após a exibição, o programa faz uma pausa de 0,7 segundos para que o usuário possa visualizar a atualização e depois limpa o terminal para a próxima iteração. A simulação ocorre de forma contínua, sem possibilidade de interrupção durante a execução.

2.3. Ferramentas Utilizadas

- **Linguagem de Programação:** Python 3.13.2 64-bit
- **Sistema Operacional:** Windows 11, Versão 24H2
- **IDE:** VSCode
- **Bibliotecas:**
    - `os` - Para limpeza do terminal.
    - `time` - Para controlar a pausa entre as gerações.

3. Resultados e Discussões

3.1. Como Utilizar o Programa

Ao iniciar o programa, o usuário será solicitado a inserir o número de células vivas no tabuleiro inicial e suas posições (linha e coluna). O programa criará a matriz 10x10 preenchida inicialmente com células mortas (`.`), e as células vivas (`V`) serão posicionadas conforme a escolha do usuário. Após a configuração inicial, o programa começa a exibir a evolução do Jogo da Vida, mostrando as gerações sucessivas. A cada atualização, a nova configuração das células será exibida, e o programa continuará até que todas as células estejam mortas.

3.2. Dados de Entrada

O programa solicita:

- O número de células vivas que o usuário deseja no início do jogo.
- As posições dessas células no tabuleiro (linha e coluna).

  Cada célula escolhida é marcada com um `V` na matriz.

3.3. Dados de Saída

O programa exibe o estado da matriz a cada geração. Cada célula é representada por:

- `V` para células vivas.
- `.` para células mortas.

O jogo continua iterando e mostrando a nova configuração até que todas as células estejam mortas, momento em que o programa exibe uma mensagem e encerra a execução automaticamente.

3.4. Testes Realizados

O programa foi testado com diferentes números de células vivas e posições iniciais, garantindo que as células evoluíssem corretamente de acordo com as regras do jogo. Testes com um número muito pequeno de células vivas (como 1 ou 2 células) e com um número grande (até 80 células) foram realizados, e o programa se comportou conforme o esperado, exibindo a evolução correta da matriz.

3.5. Possíveis Erros

- Se o usuário inserir posições fora do intervalo (0-9), o programa solicita que a entrada seja refeita.
- O tratamento de entrada das linhas e colunas é feito utilizando o bloco `try: except:`, garantindo que o usuário insira valores válidos. Caso o valor inserido não seja um número inteiro ou esteja fora do intervalo permitido, o programa solicita que o usuário tente novamente.
- O programa assume que o usuário sempre escolhe um número válido de células vivas e localizações, então entradas inválidas em número ou formato de células vivas causam erros de execução.

4. Conclusão

O objetivo de implementar o **Jogo da Vida** foi cumprido com sucesso. O programa simula a evolução de células vivas e mortas em uma matriz 10x10 de acordo com as regras definidas por John Conway. Durante o desenvolvimento, a escolha das regras do jogo foi corretamente aplicada e os dados de entrada foram validados adequadamente para evitar erros.

Não foram implementadas funcionalidades para interromper a execução durante a simulação, o que limita a interatividade do programa enquanto ele está em execução. Uma possível melhoria seria a implementação de uma funcionalidade para o usuário parar a simulação quando desejar e poder escolher matriz de outros tamanhos. Além disso, otimizações no tempo de execução, como permitir uma visualização mais eficiente das gerações, poderiam ser exploradas.

Referências

PYTHON SOFTWARE FOUNDATION. *Python 3.13.2 Documentation*. Disponível em: `https://docs.python.org/3/`. Acesso em: 10 mar. 2025.

UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA. *Slides da disciplina EXA801 - Algoritmos e Programação I*. 2025.