Matrix Adivinhation

Gerson Ferreira dos Anjos Neto

Graduando em Engenharia da Computação - Universidade Estadual de Feira de Santana (UEFS) - Feira de Santana - Bahia - Brasil

gersonferreiradosanjosneto@gmail.com

Resumo. *Este relatório tem como objetivo descrever o desenvolvimento do jogo das "Somas Esquecidas" ou "Matrix Adivinhation". Inspirado na franquia de filmes "The Matrix", a aplicação se trata de um "Text-Based Game" baseado na manipulação de matrizes – preenchidas com números pseudo-aleatórios – cujas somas de suas linhas e colunas deverão ser adivinhadas pelos jogadores.*

1. Introdução

"*The Matrix*" é uma franquia de filmes dirigidos e roteirizados pelos irmãos Andy, Larry e Lana Wachowski, atualmente composta por 4 titulos: *The Matrix (1999)*, *Matrix Reloaded (2003)*, *Matrix Revolutions (2003)* e *Matrix Resurrections (2021).* O estilo *cyberpunk* (subgênero da ficção científica, caracterizada pelo avanço da tecnologia em detrimento da qualidade de vida) cativou a milhares de fãs ao redor do mundo, tornando a franquia um ícone dentro de sua categoria.

A narrativa é baseada em um mundo controlado pelas máquinas, onde a humanidade é mantida aprisionada em casulos e submetida a um estado de sono profundo, enquanto são utilizados como baterias para alimentar os autômatos. Os poucos que eram capazes de se libertar desse sono, uniam-se em uma rebelião contra a dominação das máquinas. Esse grupo era liderado pelo personagem Morpheus, sua companheira Trinity e o hacker Neo, o único capaz de libertar a humanidade dessa ilusão. Ao longo de sua batalha contra a simulação Neo e seu grupo terão de enfrentar diversos tipos de desafios, dentre eles o Agente Smith – representação da autoridade na Matrix, sua responsabilidade é manter a ordem e neutralizar as forças de resistência – que usará todas as suas capacidades para impedir-los [1].

O jogo das "Somas Esquecidas" ou "*Matrix Adivinhation*" foi inspirado nos desafios de lógica e raciocínio que são apresentados ao longo dos filmes, consiste de um *text-based game* (jogo baseado em texto) de tabuleiros que deve ser disputado entre dois jogadores, onde a cada rodada os jogadores escolhem uma linha ou coluna e tentam adivinhar a soma dessas.

Os primeiros *text games* (jogos de texto) surgiram entre os anos 70 e o final dos anos 80, precedendo a era dos gráficos 2D e 3D nos computadores pessoais, nestes os ambientes eram descritos nos mínimos detalhes e o jogador era convidado a interagir com o mesmo por meio de simples comandos que determinavam desde a sua movimentação à sua interação com determinados objetos ou criaturas. Um dos primeiros títulos conhecidos desse gênero foi o *Colossal Cave Adventure* desenvolvido por Will Crowther e Don Woods no ano de 1975 [2].

O desenvolvimento dessa aplicação foi feito utilizando a linguagem de programação *Python* em sua versão 3.8.10 dentro da *IDE* (Ambiente de Desenvolvimento Integrado) PyCharm 2022.1. Todo o desenvolvimento e testes foram realizados dentro do *OS* (Sistema Operacional) Ubuntu 20.04.4 LTS.

2. Metodologia

Para melhor compreendimento da construção de tal aplicação, as seções a seguir descreverão o processo de criação da mesma, desde os fundamentos teóricos utilizados à implementação do código.

2.1 Fundamentação Teórica

O *source code* (código fonte) foi construído visando uma boa legibilidade, manutenção e funcionamento do mesmo. Para tal, foram utilizadas as seguintes estruturas nativas da linguagem *Python*:

- Estruturas Condicionais: Os comandos *if, elif* e *else* são utilizados quando é necessário redirecionar o fluxo de execução do código, por meio de uma condição;

- Estruturas de Repetição: *While* e *For* são comando que permitem a repetição da execução de um bloco de código, seja por meio de uma condição ou pela interação de objetos;

- Funções: Permitem que um porção de código – que execute uma tarefa específica – seja reutilizada diversas vezes.

- Listas e Variáveis: São locais reservados na memória para o armazenamento de dados. Divergem entre si, de forma que a variável armazena apenas um elemento de um determinado tipo, enquanto a lista pode armazenar diversos elementos de variados tipos.

Python – criada por Guido Van Rossum em 1989 – é uma linguagem de programação de alto nível (*i.e.* com sintaxe simples, próxima da linguagem humana) onipresente, capaz de resolver os problemas dos mais simples aos mais complexos. Dentro da sua vasta biblioteca, além das funções antes mencionadas, o módulo *Random* também se fez extremamente útil no desenvolvimento do supracitado *software*, visto que dentre as suas inúmeras funcionalidades à capacidade de geração de números pseudo-aleatórios [3].

O processo de geração de tais números envolve algoritmos capazes de produzir resultados aparentemente aleatórios, visto que tais resultados são na verdade determinados por um valor inicial conhecido como *seed* (semente) ou *key* (chave). Tendo conhecimento de tal chave, é possível replicar facilmente tais resultados, por isso

o nome pseudo-aleatório (Pseudo, do grego *pseudes* significa “mentira” ou “falsidade”) [4].

2.2 Funcionamento Geral

Tendo esclarecido os conceitos teóricos utilizados para o desenvolvimento da aplicação, podemos partir para a lógica geral por trás do seu funcionamento. O jogo foi projetado para atender os seguintes requisitos:

- Os jogadores devem escolher entre disputar em 1 ou 2 tabuleiros;
- Os níveis de dificuldade influenciam diretamente na construção dos tabuleiros como mostrado na figura 1;

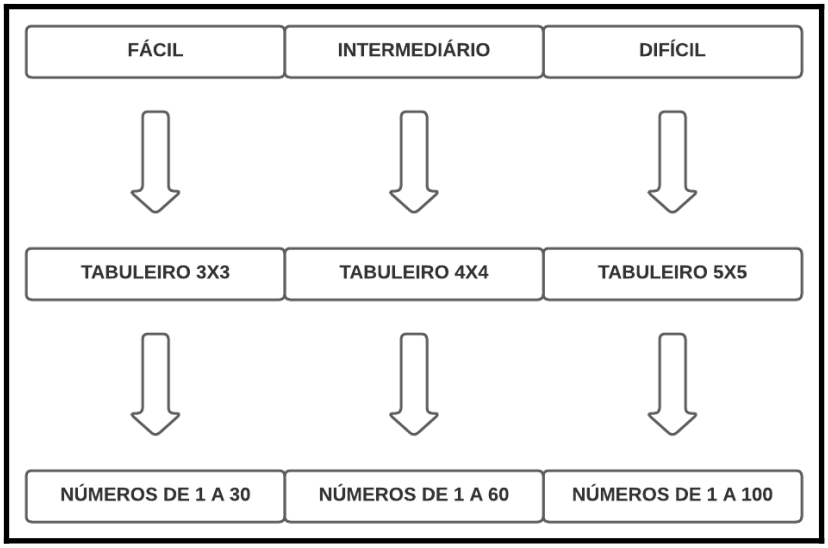

Figura 1. Funcionamento das dificuldades.

- As somas das linhas e colunas de cada tabuleiro devem permanecer ocultas, sendo respectivamente armazenadas ao fim de cada linha e abaixo de cada coluna;
- O critério de encerramento também deve ser escolhido pelos jogadores, podendo ser por número de rodadas (desde que ímpar) ou completando o(s) tabuleiro(s).

Para atender tais critérios de funcionamento e visando a modularidade do código, as funções a seguir foram criadas:

- *setDifficulty*: A função base do *source code*, verifica a dificuldade escolhida pelos jogadores e retorna um vetor que será utilizado pela maioria das outras funções;
- *generateNumbers*: Utiliza a função *randint* do módulo *Random* dentro de um laço *for* para gerar *n* números pseudo-aleatórios, sendo *n* definido pelo nível de dificuldade selecionado. O laço while é utilizado para evitar a repetição dos números. Retorna um vetor contendo os números que preencherão a matriz;

- *getRMatrix*: Inicializa uma matriz, aqui denominada "*rMatrix*" com tamanho *n + 1*, sendo *n* o tamanho pré-definido em relação a dificuldade selecionada. Por meio de um laço *for*, a matriz é preenchida com os números previamente gerados e um espaço e deixado para as somas (Figura 2);

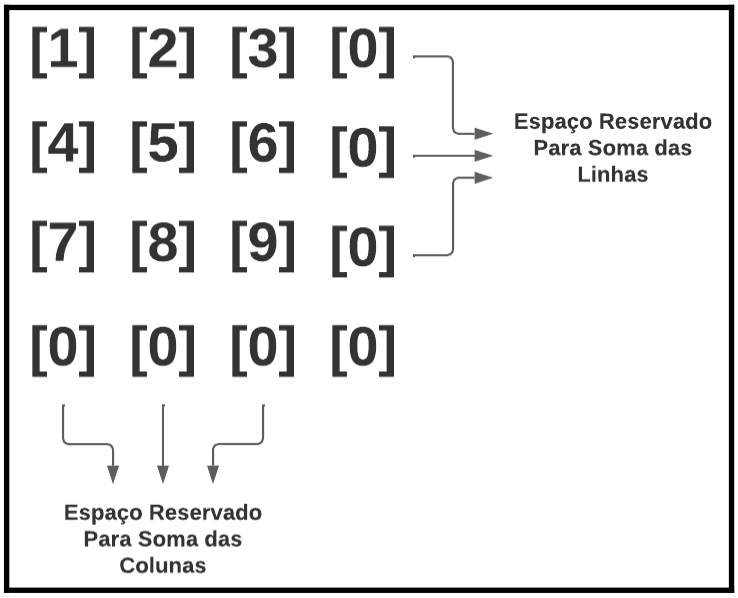

Figura 2. Exemplo de rMatrix.

- *calculateSum*: Percorre a "*rMatrix"* e realiza soma das suas linhas e colunas, armazenando-as respectivamente ao fim de cada linha e abaixo de cada coluna (Figura 3);

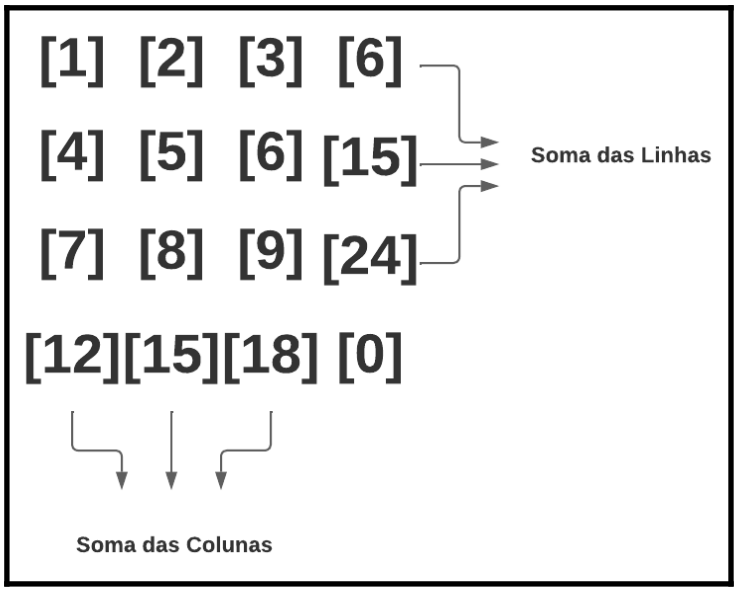

Figura 3. Exemplo de rMatrix com as somas das linhas e colunas.

- *getIMatrix*: Gera uma matriz, aqui denominada *"iMatrix"* de tamanho *n* sendo *n* o tamanho pré-definido em relação a dificuldade. Essa é preenchida com "X" é exibida para os jogadores durante a partida;

- *verifyBoardIsComplete*: É executada ao fim de cada rodada, sua função e percorre "*iMatrix"* verificando se a mesma já está completa (*i.e.* se não restam mais "X").

2.3 Entrada de Dados

Nesta seção será descrito o processo de recepção e tratamento das informações inseridas pelos usuários.

Pensando na melhor interação do usuário com o jogo, esse foi estruturado por meio de menus. Primeiramente, os jogadores deverão fazer 3 escolhas, sendo elas respectivamente: a quantidade de tabuleiros, o nível de dificuldade desejado e o critério de encerramento da partida. Essas informações são validadas – visando o bom funcionamento do código (*e.g.* o número de rodadas, quando definido, não pode ser ímpar) – e armazenadas.

Em seguida o nome dos jogadores é solicitado e a partida se inicia, enquanto o limite de rodadas não for atingido e/ou o tabuleiro estiver incompleto, cada usuário montará a sua jogada, escolhendo: linha ou coluna, o número da mesma e o seu palpite sobre a sua soma. Esses dados serão validados da seguinte forma:

- A linha ou coluna não pode estar completa, caso sim o jogador é convidado a escolher outra posição;

- O número da linha ou coluna deve respeitar o tamanho da matriz (*e.g.* para uma matriz 3X3 o número das linhas e colunas vai de 1 à 3);

- Por fim, o valor da soma deve ser um tipo válido (*i.e.* um número).

2.4 Manipulação dos Dados

Nesta seção será descrito o processo de manipulação das informações inseridas pelos usuários, tendo em vista que as mesmas já foram previamente tratadas nos processos descritos anteriormente.

Após a recepção de tais informações, o algoritmo verificará se houve acerto(s), respeitando os casos descritos a seguir:

- Caso 1: Ambos os jogadores acertam a soma, as linhas ou colunas escolhidas por ambos são reveladas e para cada casa é atribuído um ponto ao jogador. Vale ressaltar que em caso de intersecção das escolhas, os jogadores receberão a mesma quantidade de pontos (Figura 4);

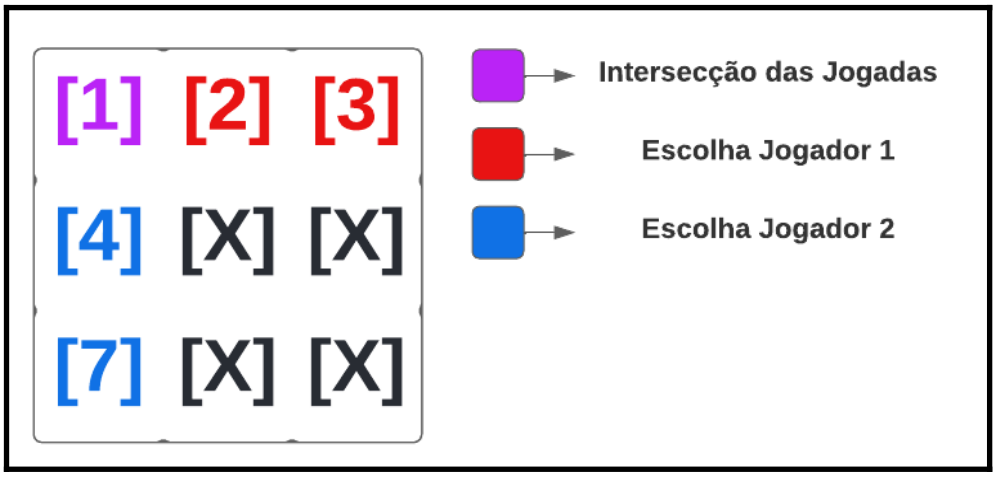

Figura 4. Exemplo de interseção de jogadas, neste caso ambos os jogadores recebem 3 pontos.

- Caso 2: Um dos jogadores acerta soma, a linha ou coluna escolhida por aquele jogador é revelada e os pontos são atribuídos para cada casa revelada;
- Caso 3: Nenhum dos jogadores acertaram, neste caso existem 2 possibilidades:
  - O valor escolhido é menor do que a soma correta → A casa com o menor valor na posição escolhida é revelada;
  - O valor escolhido é maior do que a soma correta → A casa com o maior valor na posição escolhida é revelada.

Por fim, vale ressaltar que as jogadas são computadas ao mesmo tempo, não antepondo nenhum dos jogadores, mas, somente no Caso 2, apenas o jogador vencedor da rodada tem seus pontos e casas avaliados.

2.5 Saída de Dados

Enfim, nesta seção será descrita a forma como os resultados supracitados são apresentados aos jogadores.

Ao fim de cada rodada são exibidos para os jogadores: os tabuleiros atualizados com as casas que foram reveladas, o histórico de jogadas e o placar parcial. Caso tenha sido escolhida a opção de 2 tabuleiros, o histórico de cada jogador é exibido separadamente.

Mesmo que tenha sido definido um número de rodadas, o algoritmo realiza ao fim de cada rodada uma verificação (por meio da função *verifyBoardIsComplete)* dos tabuleiros, caso estejam completos a partida é encerrada e o jogador com mais pontos é anunciado como o vencedor.

3. Resultados e Discussões

Após descrito o funcionamento do *software*, será descrito nesta seção um breve manual de uso para o mesmo.

Para garantir o bom andamento do jogo, os seguintes passos deverão ser executados (Figura 9):

I. Ao iniciar o jogo, será solicitado nesta ordem: a quantidade de tabuleiros, a dificuldade é o critério de encerramento. As escolhas deverão respeitar as opções exibidas na tela. Logo após, os jogadores deverão inserir seus nomes;

II. A cada rodada, o jogador indicado na tela deve escolher uma linha ou coluna e o valor que acha ser equivalente a soma da mesma. Não será aceita jogada em uma linha ou coluna que já esteja completa;

III. Após ambos fazerem suas escolhas, os resultados serão computados. O tabuleiro é atualizado e exibido, juntamente com o histórico de jogadas e o placar parcial.

As etapas II e III se repetirão nessa ordem durante a partida, respeitando o critério de encerramento escolhido. Ao final, o jogador com o maior número de pontos (*i.e.* o que revelou o maior número de casas) vence.

4. Conclusão

O processo de desenvolvimento do *software* atendeu, com sucesso, a todos os critérios pré-estabelecidos. O conceito de *text game* apresentado traz para o mesmo uma unicidade cativante, entretanto, futuras implementações poderiam evoluir o projeto utilizando:

- Uma interface mais interativa com elementos 2D ou 3D;
- Um ambiente que melhor represente a inspiração do jogo (a franquia *"The Matrix");*
- A adição de efeitos sonoros e animações que estimulem os jogadores.

5. Referências

[1] Carolina Marcelo (2022) "Filme The Matrix: resumo, análise e explicação", https://www.culturagenial.com/filme-the-matrix/, Maio de 2022.

[2] Leigh Alexander (2014) "The joy of text – the fall and rise of interactive fiction", https://www.theguardian.com/technology/2014/oct/22/interactive-fiction-awards-games, Maio de 2022.

[3] Swaroop C. H. (2008) “A Byte of Python”, Version 2.1.

[4] Alexander Arobelidze (2020) “Random Number Generator: How Do Computers Generate Random Numbers?”, https://www.freecodecamp.org/news/random-number-generator/, Maio de 2022.