ALGORITMO E LÓGICA DE PROGRAMAÇÃO

1. Introdução

Neste projeto, foi desenvolvido um jogo da memória utilizando a linguagem C# dentro do ambiente Unity. O objetivo é aplicar conceitos de algoritmos e lógica de programação, como estruturas de decisão, repetição, vetores, matrizes e funções, conforme solicitado na atividade.

2. Estruturas Utilizadas no Código

- **Estruturas de Decisão**

O código utiliza estruturas condicionais (if, else if, else) diversas vezes, por exemplo:

Para verificar se o jogador clicou em uma carta: **if (Input.GetMouseButtonDown(0))**

Para verificar se uma carta já foi virada ou se há duas viradas: **if (cardToFlip == gameManager.cardFliped1 || cardToFlip == gameManager.cardFliped2)**

Esses comandos controlam o fluxo do jogo e garantem que o jogador não possa virar mais cartas do que o permitido.

- **Estruturas de Repetição**

A principal repetição usada é o laço while dentro das coroutines, responsável por animar a rotação da carta durante o tempo de virada:

```
while (elapsedTime < flipDuration)

{

   card.transform.rotation = Quaternion.Slerp(startRotation, endRotation, elapsedTime / flipDuration);

   elapsedTime += Time.deltaTime;

   yield return null;

}
```

Essa parte é executada até a animação de virar a carta ser concluída, repetindo o mesmo movimento gradualmente.

- **Vetores e Matrizes**

No seu código do CardFlip não há vetores diretamente, mas eles provavelmente estão no GameManager, que deve armazenar as cartas do jogo em um array (por exemplo, GameObject[] cards).

> O jogo utiliza vetores no script GameManager para armazenar todas as cartas em cena e facilitar a comparação entre elas.

- **Funções e Procedimentos**

O código tem várias funções (ou métodos), como:

- **Start() → inicializa o script e encontra o GameManager;**
- **Update() → detecta cliques do jogador e controla o fluxo principal do jogo;**
- **StartFlipBack() → chamada pelo GameManager para desvirar cartas erradas;**
- **PerformFlip() → executa a animação de rotação da carta (função central do script).**

Essas funções são chamadas conforme o jogo avança, organizando o código em partes independentes.

3. Objetivo do Código

O objetivo é permitir que o jogador clique nas cartas e as vire, utilizando animações controladas por corrotinas (IEnumerator) e condições lógicas.

Quando duas cartas são viradas, o GameManager verifica se formam um par e com isso as cartas iguais somem e caso contrário, elas são desviradas automaticamente para voltar a jogar.

4. Conclusão

O projeto demonstra a aplicação prática dos conceitos de algoritmos em um contexto real de desenvolvimento de jogos. Foram utilizadas estruturas de decisão, repetição, vetores e funções, de forma integrada e lógica, conforme os requisitos da disciplina.