Para Entrega 2 — Modelagem Matemática e Funções Aplicadas ao Jogo da Memória

Introdução

Este projeto tem como objetivo apresentar as relações entre **Matemática** e **Desenvolvimento de Jogos Digitais**, a partir da análise de um **jogo da memória** desenvolvido na **Unity** com scripts em **C#**. Durante o funcionamento do jogo, são aplicados conceitos matemáticos como **funções lineares**, **interpolação**, **probabilidade**, **operações lógicas** e **contagem regressiva temporal**.

1. Estrutura Geral do Jogo

O jogo consiste em cartas dispostas em posições aleatórias.
O jogador deve clicar para virá-las e encontrar pares iguais.
O sistema reconhece acertos, destrói as cartas combinadas e controla o tempo restante da partida.
Além disso, há menus e gerenciamento de cenas para navegar entre as telas de jogo, opções e encerramento.

2. Modelagem Matemática nas Funcionalidades

2.1. Funções de Transformação (Rotação das Cartas)

Durante a animação de virar a carta, é utilizada uma função matemática que transforma o estado da rotação de 0° para 180° com o tempo.
Essa transformação é feita por meio da função:

```
Quaternion.Slerp(startRotation, endRotation, elapsedTime / flipDuration);
```

Modelagem Matemática:

$$R(t) = R_0 + (R_1 - R_0) \cdot \frac{t}{T}$$

R0 : ângulo inicial
R1 : ângulo final
T: tempo total de rotação
t: tempo decorrido

Essa é uma **função linear no tempo**, utilizada para criar uma animação suave e contínua.

2.2. Cálculo do Tempo e Função Decrescente

O script do cronômetro realiza a contagem regressiva com:

```
tempoRestante -= Time.deltaTime;
```

Matematicamente:

$$T(t) = T_0 - t$$

onde T0 é o tempo inicial e t é o tempo decorrido.
Quando T(t)=0, o jogo muda de cena automaticamente.
Além disso, o tempo é convertido para minutos e segundos usando funções de **piso e módulo**:

$$\text{minutos} = \lfloor T/60 \rfloor, \quad \text{segundos} = T \mod 60$$

2.3. Probabilidade e Aleatoriedade

As cartas são sorteadas aleatoriamente para garantir variedade no jogo:

```
rnd = Random.Range(0, cards.Length);
```

Matematicamente, cada carta tem a mesma **probabilidade uniforme** de ser escolhida:

$$P(X = i) = \frac{1}{n}$$

onde n é o número total de tipos de carta.
A estrutura garante que cada tipo apareça exatamente duas vezes, aplicando um controle lógico com contadores.

2.4. Lógica Booleana e Funções Condicionais

Em diversas partes do código, a lógica do jogo depende de comparações:

```
if (info1.id == info2.id)
```

Essa expressão representa uma **função booleana**:

$$f(a,b) = \begin{cases} 1, & \text{se } a = b \\ 0, & \text{caso contrário} \end{cases}$$

Essa função controla o fluxo do jogo, determinando se as cartas combinam ou devem ser viradas novamente.

3. Aplicações Diretas de Funções e Variáveis

Conceito Matemático	Aplicação no Código	Exemplo
Função linear	Controle do tempo e rotação	`elapsedTime / flipDuration`
Função decrescente	Cronômetro regressivo	`tempoRestante -= Time.deltaTime`
Função modular	Cálculo de segundos	`Mathf.FloorToInt(tempo % 60)`
Probabilidade uniforme	Escolha aleatória de cartas	`Random.Range(0, cards.Length)`

Função booleana	Comparação de pares	`if (id1 == id2)`
Interpolação geométrica	Animação 3D	`Quaternion.Slerp(...)`

4. Interpretação Matemática do Sistema

O jogo pode ser modelado como um **sistema dinâmico discreto**, onde o estado do jogo muda conforme o tempo e as ações do jogador:

$$S_{t+1} = f(S_t, \text{ação})$$

Essa relação descreve como cada evento (como virar uma carta) gera uma nova configuração do sistema, sendo uma aplicação direta de **modelagem matemática computacional**.

5. Conclusão

A Matemática é a base estrutural do jogo, mesmo que de forma implícita.
As **funções temporais**, **interpolação**, **lógica condicional** e **probabilidade** tornam o comportamento do jogo coerente, previsível e divertido.
Com isso, o projeto demonstra como a Matemática é aplicada na prática de forma essencial no desenvolvimento de jogos digitais.

6. Possíveis Extensões Matemáticas

- Criar **pontuação** com função inversa ao tempo gasto (ex: [Equação][Equação]
- Ajustar o **tempo inicial** com função exponencial por nível (ex: [Equação][Equação][Equação]n);
- Inserir estatísticas de desempenho do jogador e média de acertos por tentativa.

7. Scripts Utilizados

Os seguintes scripts foram utilizados na construção do jogo e demonstram os conceitos citados:

1. **Card.cs** – Define propriedades e comportamentos básicos das cartas.
2. **CardFlip.cs** – Controla a animação de virar as cartas e verificar combinações.
3. **CardInfo.cs** – Armazena o identificador numérico das cartas (pares).
4. **CardRandomizer.cs** – Posiciona e sorteia as cartas de forma aleatória.
5. **CronometroJogo.cs** – Gerencia o tempo da partida e a troca de cena quando o tempo acaba.
6. **GerenciadorDeCenas.cs** – Alterna entre cenas de menu, jogo e saída.
7. **GameManager.cs** – Controla a lógica central de combinação e estado das cartas.
8. **menu_principal_meneger.cs** – Gerencia os painéis do menu principal e das opções.
9. **Scrit_Jogar.cs** – Botão simples para iniciar o jogo.