PROFESSOR:Luis Fernando dos Santos Pires	
CURSO:	
DISCIPLINA: Algoritmo e Lógica de Programação	
TURMA: 1 semestre matutino	DATA:10/2025
ALUNO(A): Arthur Henrique Dos Anjos, Arthur Daher Franceschelli, Fernanda Figueiredo, Ryan Santos, William Takuya Takeuchi Takaki	

1. Introdução a mecânica do jogo.

A disciplina de Algoritmo e Lógica de Programação é essencial para o desenvolvimento do pensamento computacional, permitindo a criação de soluções estruturadas por meio de códigos. O presente artigo tem como foco a criação do jogo Traffic Fury, um jogo do gênero carro/arcade, cuja mecânica envolve desviar de veículos em uma pista. O desenvolvimento desse jogo serviu como ferramenta de aprendizado e aplicação direta dos conteúdos abordados na disciplina. Desde a conceição das regras de jogo até a implementação do sistema de colisão e movimentação do carro, cada etapa exigiu a utilização de estruturas algorítmicas bem definidas e lógica condicional.

2. Fundamentação Teórica.

A disciplina de Algoritmo e Lógica de Programação é a base do desenvolvimento de qualquer sistema computacional. Entre os principais conceitos estão as estruturas de decisão (como if, else) e estruturas de repetição (como for, while), que são essenciais para definir o comportamento dinâmico de programas e jogos. Além disso, a lógica condicional é empregada para controlar eventos dentro do jogo, como o fim da partida em caso de colisão.

3. Metodologia e desenvolvimento.

O desenvolvimento do jogo *Traffic Fury* foi realizado utilizando a engine Unity, utilizando a linguagem de programação empregada C#, padrão na Unity, permitindo uma integração direta com os componentes visuais e lógicos do jogo. O projeto foi construído com base nos conhecimentos adquiridos na disciplina de Algoritmo e Lógica de Programação.

Movimentação do carro do jogador:

O primeiro script controla o deslocamento horizontal do carro com base na entrada do teclado (setas ou teclas A/D). A movimentação é feita com base em condições que verificam se o jogador está pressionando a direção esquerda ou direita:

```
public class Horizontal : MonoBehaviour
{
    // Start is called once before the first execution of Update after the MonoBehaviour is created
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetAxisRaw("Horizontal") > 0)
        {
            transform.Translate(0.01f, 0, 0);
        }
        else if(Input.GetAxisRaw("Horizontal")< 0)
        {
            transform.Translate(-0.01f, 0, 0);
        }
```

Essa lógica simples utiliza uma **estrutura condicional** (if/else if) para determinar a direção do movimento e aplica a movimentação ao objeto com transform.Translate(). Crucial para desviar dos obstáculos inimigos.

Implementação da pista infinita:

Para simular o movimento contínuo da estrada, foi criado um segundo script que move o cenário para trás, reposicionando-o quando ele atinge um certo ponto, criando a ilusão de uma pista infinita:

```
public class PistaInfinita : MonoBehaviour
{
    public float speed = 5f;
    public float comprimentoPista = 50f;

    void Update()
    {
        // Move para trás (igual ao seu código da grama)
        transform.Translate(Vector3.forward * -speed * Time.deltaTime);

        // Verifica se a pista já passou completamente
        if (transform.position.z <= -comprimentoPista)
        {
            // Reposiciona a pista
            ReposicionarPista();
        }
    }

    void ReposicionarPista()
    {
        // Move a pista de volta para a posição inicial
        Vector3 novaPosicao = transform.position;
        novaPosicao.z += comprimentoPista * 2f;
        transform.position = novaPosicao;
    }
```

Essa propriedade é uma demonstração clara do uso de condicionais, operações matemáticas e controle de vetores, que são empregados para assegurar que a performance do jogo permaneça estável, mesmo em longos períodos de jogo.

4. **O exemplo de código abaixa representa um sistema de matrizes e estruturas de repetição:**

```
for (int i = 0; i < 3; i++)
{
    Console.WriteLine(arr1[i]);
}
Console.WriteLine();

int[,] mat1 = {
    { 1, 2, 3 },     // Corrigido: vírgula em vez de fecha chaves
    { 4, 5, 6 },     // Corrigido: vírgula
    { 7, 8, 9 },     // Corrigido: vírgula
    { 10, 11, 12 }   // Corrigido: removido ponto e vírgula
}; // Adicionado: fecha chaves e ponto e vírgula

mat1[2, 1] = 20;

for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 3; j++)
    {
        mat1[i, j] *= 2;
        Console.WriteLine(mat1[i, j]); // Corrigido: era [i, 1], agora [i, j]
```

A utilização de matrizes e estruturas de repetição é fundamental nessas situações, pois permite o acesso rápido às colisões dos veículos adjacentes. A matriz opera como uma lista de localizações. Se temos conhecimento das coordenadas X e Y, isso te transporta rapidamente para o que se encontra naquela localização. Assim que a matriz identifica uma colisão, uma notificação de "você perdeu" será criada automaticamente, resultando na reinicialização do jogo. O ciclo checa a cada segundo todas as pistas para gerar novos desafios, nesse caso, o veículo adversário que se aproxima na direção oposta.

5. Conclusão.

A criação do jogo Traffic Fury mostrou, na prática, a importância dos conceitos de Algoritmo e Lógica de Programação na elaboração de jogos eletrônicos. Por meio da execução de recursos como a movimentação do personagem, um cenário sem fim e regras condicionais, foi viável utilizar diretamente o aprendizado teórico obtido nas aulas. O uso da plataforma Unity e a linguagem de programação C# ajudaram na familiarização com recursos comumente utilizados no setor tecnológico como um todo.