FUNDAÇÃO ESCOLA DE COMÉRCIO ÁLVARES PENTEADO – FECAP
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO MIRANDA DE SOUZA
SAID SALES DE SOUSA
SOFIA BOTECHIA HERNANDES
VITOR PAES KOLLE
VICTÓRIA DUARTE VIEIRA AZEVEDO

UM RELATÓRIO SOBRE A IMPLEMENTAÇÃO DA DISCIPLINA ALGORITMOS E LÓGICA DE PROGRAMAÇÃO NA SEGUNDA ENTREGA DO JOGO INVERKAN

SÃO PAULO
2025

FUNDAÇÃO ESCOLA DE COMÉRCIO ÁLVARES PENTEADO – FECAP
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO MIRANDA DE SOUZA
SAID SALES DE SOUSA
SOFIA BOTECHIA HERNANDES
VITOR PAES KOLLE
VICTÓRIA DUARTE VIEIRA AZEVEDO

UM RELATÓRIO SOBRE A IMPLEMENTAÇÃO DA DISCIPLINA ALGORITMOS E LÓGICA DE PROGRAMAÇÃO NA SEGUNDA ENTREGA DO JOGO INVERKAN

Trabalho apresentado à Fundação Escola de Comércio Álvares Penteado, São Paulo, durante o 1º semestre do Bacharelado em Ciência da Computação.

Orientador: Prof. Adriano Felix Valente

SÃO PAULO
2025

RESUMO

No projeto de jogos digitais do grupo Inverkan estão sendo incorporadas diversas funcionalidades para melhorar a experiência do jogador. Entre estas, com base em conhecimentos desenvolvidos na disciplina Algoritmos e Lógica de Programação do primeiro semestre do curso de Ciência da Computação, foram criadas diversas funções contendo algoritmos capazes de efetuar funcionalidades do jogo de forma eficiente e ágil. Sendo assim, o presente artigo expõe os *script*s de tais funções, tal como sua relação com a disciplina em questão e sua proposta dentro do jogo.

Palavras-chave: algoritmos; jogos digitais; lógica de programação.

ABSTRACT

In the Inverkan group's digital games project, several features are being incorporated to improve the player's experience. Among these, based on concepts developed in the Algorithms and Programming Logic discipline of the first semester of the Computer Science course, several functions were created containing algorithms capable of performing game functionalities efficiently and quickly. Therefore, this article presents the *scripts* of such functions, as well as their relationship with the subject, and their purpose within the game.

Keywords: algorithms; digital games; programming logic.

SUMÁRIO

1. INTRODUÇÃO

A disciplina de Algoritmos e Lógica de Programação é direcionada para o desenvolvimento de capacidades de pensamento crítico, de análise lógica e de programação usando a linguagem C#.

Com base nisso, o grupo Inverkan desenvolveu mais algumas funções essenciais para o funcionamento do projeto do jogo Inverkan, em adição àquelas implementadas na primeira entrega dele, tal como alterações nestas enviadas primordialmente. Assim, estas funções incluem a compleição de determinadas *quests* como a implementação de iluminação inteligente e de semáforos inteligentes, de coleta de lixo e de estacionamento inteligente.

Enfim, o presente artigo tem como objetivo expor e organizar essas funções, as quais utilizam conceitos como loops, expressões *if-else*, operadores lógicos e de comparação, matrizes, entre outros, aprendidos nas aulas da disciplina em questão.

2. DESENVOLVIMENTO

2.1 COLETA DE LIXOS

Mais adiante, foi implementada a função referente à *quest* de coleta de lixo, em que o jogador faz a coleta de resíduos espalhados pela cidade por meio da interação usando a tecla *e* e uma mensagem educativa é mostrada ao final.

Esta função inclui blocos *if-else* e operadores de comparação (“>” e “<”), ambos conceitos trabalhados em aula durante as aulas de Algoritmos e Lógica de Programação. A seguir, está o script referente à função descrita:

```
using UnityEngine;
using TMPro;
using System.Collections;

public class LixoManager : MonoBehaviour
{
    public static LixoManager Instance;

    [Header("Configuração da Grade")]
    public int linhas = 2;
    public int colunas = 5;

    [Header("Mensagem Final")]
    public TextMeshProUGUI mensagemTexto;
    [TextArea(3, 10)]
    public string mensagemFinal = "Você limpou toda a área! A cidade agradece pela su
a dedicação meio ambiente.";

    [Header("Efeito de digitação")]
    public float velocidadeDigitacao = 0.03f;

    private GameObject[,] matrizLixos;
    public GameObject painelDialogo;

    void Awake()
    {
```

```
        Instance = this;
    }

    void Start()
    {
        matrizLixos = new GameObject[linhas, colunas];
        GameObject[] lixosNaCena = GameObject.FindGameObjectsWithTag("Lixo");

        if (lixosNaCena.Length != linhas * colunas)
        {
            Debug.LogError("Número de lixos não corresponde ao tamanho da matriz defin
ida!");
            return;
        }

        int index = 0;
        for (int i = 0; i < linhas; i++)
        {
            for (int j = 0; j < colunas; j++)
            {
                matrizLixos[i, j] = lixosNaCena[index];
                index++;
            }
        }

        // Verificando se o campo de texto foi atribuído
        if (mensagemTexto != null)
            mensagemTexto.text = "";
        else
            Debug.LogError("Texto não foi atribuído ao campo mensagemTexto!");
    }

    public void ColetarLixo(GameObject lixoColetado)
    {
        for (int i = 0; i < linhas; i++)
        {
            for (int j = 0; j < colunas; j++)
            {
                if (matrizLixos[i, j] == lixoColetado)
                {
                    matrizLixos[i, j] = null;
                }
            }
        }

        if (TodosLixosColetados())
        {
            Debug.Log("Todos os lixos foram coletados!");
            StartCoroutine(DigitarMensagem());
        }
    }

    private bool TodosLixosColetados()
    {
```

```
        foreach (GameObject lixo in matrizLixos)
        {
            if (lixo != null)
                return false;
        }
        return true;
    }

    IEnumerator DigitarMensagem()
    {
        // Garantindo que a referência do texto não é nula
        if (mensagemTexto == null)
        {
            Debug.LogError("MensagemTexto (TextMeshProUGUI) não foi atribuída no Inspe
ctor!");
            yield break;
        }
        if (painelDialogo != null)
        {
            painelDialogo.SetActive(true);
        }

        // Limpar texto antes de começar a digitar
        mensagemTexto.text = "";

        // Iniciar o efeito de digitação
        foreach (char letra in mensagemFinal)
        {
            mensagemTexto.text += letra;
            yield return new WaitForSeconds(velocidadeDigitacao);
        }
    }
    void Update()
    {
        // Detecta quando a tecla E  pressionada
        if (Input.GetKeyDown(KeyCode.E))
        {
            // Verifica se o painel est ativo
            if (painelDialogo != null && painelDialogo.activeSelf)
            {
                // Desativa o painel para esconder o di logo
                painelDialogo.SetActive(false);
            }
        }
    }}
```

2.2 SEMÁFOROS INTELIGENTES

Ademais, alterou-se também a função responsável pela implementação dos semáforos inteligentes, a fim de torná-la mais assertiva e funcional. Nesse sentido, a nova versão inclui *loops foreach*, blocos *if-else* e operadores lógicos (“*&&*”). A seguir, está o script referente à função descrita:

```
using UnityEngine;
using TMPro;
using System.Collections;

public class QuestSemaforos : MonoBehaviour
{
    private bool isActive = false;

    public GameObject semaforosPai;
    public ContadorQuests contadorQuests;

    [Header("UI")]
    public GameObject panel;
    public TextMeshProUGUI mensagemCanvas;

    [Header("Texto da Mensagem")]
    [TextArea(3, 10)]
    public string textoCompleto = "Parabéns! A iluminação inteligente que você impleme
ntou melhora a segurança pública. Reduz o consumo de energia e os custos operacionai
s. Com sensores e automação, ela proporciona eficiência e sustentabilidade. Além diss
o, melhora a qualidade de vida da nossa cidade.";

    [Header("Configurações de Efeito")]
    public float velocidadeDigitacao = 0.03f;
    public float tempoEntreFrases = 1.2f;
    public float velocidadeApagar = 0.01f;
```

```
public bool escrevendo = false;

void Start()
{
    foreach (Transform filho in semaforosPai.transform)
    {
        filho.gameObject.SetActive(false);
    }

    panel.SetActive(false);
}

void Update()
{
    if ((Input.GetKey(KeyCode.E)) && (isActive == true) && (!escrevendo))
    {
        panel.SetActive(true);

        foreach (Transform filho in semaforosPai.transform)
        {
            filho.gameObject.SetActive(true);
        }

        StartCoroutine(DigitarFrasesSeparadas());
        contadorQuests.QuestRealizada();
    }

    if (Input.GetKeyDown(KeyCode.Space))
    {
        panel.SetActive(false);
        StopAllCoroutines();
        mensagemCanvas.text = "";
        escrevendo = false;
    }
}

IEnumerator DigitarFrasesSeparadas()
{
    escrevendo = true;

    string[] frases = textoCompleto.Split('.');

    foreach (string fraseBruta in frases)
    {
        string frase = fraseBruta.Trim();
        if (string.IsNullOrEmpty(frase)) continue;

        mensagemCanvas.text = "";

        foreach (char letra in frase + ".")
        {
            mensagemCanvas.text += letra;
            yield return new WaitForSeconds(velocidadeDigitacao);
        }

        yield return new WaitForSeconds(tempoEntreFrases);
```

```
            while (mensagemCanvas.text.Length > 0)
            {
                mensagemCanvas.text = mensagemCanvas.text.Substring(0, mensagemCanvas.
text.Length - 1);
                yield return new WaitForSeconds(velocidadeApagar);
            }
        }
        gameObject.SetActive(false);
    }


    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            isActive = true;
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            isActive = false;
        }
    }
}
```

2.3 ROTAÇÃO DIA E NOITE

Por fim, a função de estacionamento inteligente foi também alterada a fim de maximizar a jogabilidade e a experiência do usuário. Sob este viés, foram implementados os conceitos de booleanas, expressões *if-else*, e *loops foreach*.

A seguir, está o script referente à função descrita:

```
using UnityEngine;

public class RotacaoDiaNoite : MonoBehaviour
{
    Vector3 rotacao = Vector3.zero;

    public Light[] postesLuz;             // Luzes dos postes (Point Light)
    public MeshRenderer[] emissores;      // Esferas emissoras (com material emissivo)

    public float anguloNoite = 90f;       // Sol no topo = dia (ajustado)
    public float anguloDia = 270f;        // Sol embaixo = noite

    public Color corEmission = Color.white;
    public float intensidadeEmission = 2f;

    public bool controlePostesAtivo = false;
    public GameObject postesPai;

    void Update()
    {
        rotacao.x = 2 * Time.deltaTime;
        transform.Rotate(rotacao, Space.World);

        float rotacaoSol = transform.rotation.eulerAngles.x;
```

```
        if (controlePostesAtivo) // Só funciona se a quest foi ativada
        {
            bool ligar = !(rotacaoSol > anguloNoite && rotacaoSol < anguloDia);
            ControlarPostesLuz(ligar);
        }
    }
    void ControlarPostesLuz(bool ligar)
    {
        foreach (Light luz in postesLuz)
        {
            luz.enabled = ligar;
        }

        foreach (MeshRenderer emissor in emissores)
        {
            if (ligar)
            {
                emissor.material.EnableKeyword("_EMISSION");
                emissor.material.SetColor("_EmissionColor", corEmission * intensidadeE
mission);
            }
            else
            {
                emissor.material.DisableKeyword("_EMISSION");
            }
        }
    }
}
```

3. CONCLUSÃO

Em suma, foram implementadas, com base nos conteúdos recentemente abordados na disciplina de Algoritmos e Lógica de Programação do primeiro semestre do curso de Ciência da Computação (como matrizes, vetores e conceitos de programação em C#), diversas funcionalidades essenciais no jogo digital Inverkan, proporcionando uma jogabilidade satisfatória e eficiente.

Sendo que, entre estas novas funcionalidades, estão as *quests* de implementação de iluminação inteligente e de semáforos inteligentes, de coleta de lixo e de estacionamento inteligente.

4. REFERÊNCIAS

UNITY TECHNOLOGIES. **Classe Light – Referência de Script**. Unity Documentation, [s.d.]. Disponível em: https://docs.unity3d.com/ScriptReference/Light.html. Acesso em: 23 mar. 2025.

UNITY TECHNOLOGIES. **Transform.eulerAngles – Referência de Script**. Unity Documentation, [s.d.]. Disponível em: https://docs.unity3d.com/ScriptReference/Transform-eulerAngles.html. Acesso em: 25 mar. 2025.

UNITY TECHNOLOGIES. **GameObject – Referência de Script**. Unity Documentation, [s.d.]. Disponível em: https://docs.unity3d.com/ScriptReference/GameObject.html. Acesso em: 23 mar. 2025.

UNITY TECHNOLOGIES. **Random.Range – Referência de Script**. Unity Documentation, [s.d.]. Disponível em: https://docs.unity3d.com/ScriptReference/Random.Range.html. Acesso em: 21 mar. 2025.

UNITY TECHNOLOGIES. **Object.Instantiate – Referência de Script**. Unity Documentation, [s.d.]. Disponível em: https://docs.unity3d.com/ScriptReference/Object.Instantiate.html. Acesso em: 29 mar. 2025.

UNITY TECHNOLOGIES. ***Physics Section***. Disponível em: https://docs.unity3d.com/Manual/PhysicsSection.html. Acesso em: 30 mar. 2025.

UNITY TECHNOLOGIES. **Scripting básico de C# no Unity – Unity Learn**. Unity Learn, [s.d.]. Disponível em: https://learn.unity.com/project/beginner-gameplay-scripting. Acesso em: 20 mar. 2025.