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 PRIMEIRA 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 PRIMEIRA 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 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 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 algumas funções essenciais para o funcionamento do projeto do jogo Inverkan, como a rotação do dia e da noite, a implementação de uma iluminação inteligente e a compleição de determinadas *quests*.

Enfim, o presente artigo tem como objetivo expor e organizar essas funções, as quais utilizam conceitos como vetores, matrizes, loops, expressões *if-else*, entre outros aprendidos na disciplina em questão.

2. DESENVOLVIMENTO

2.1 ROTAÇÃO DO DIA E DA NOITE

Em primeiro plano, o grupo Inverkan decidiu implementar uma função para gerenciar a rotação do Sol no jogo, para criar ciclos de dia e noite. A seguir, está o simples script referente à função descrita, aplicado à luz direcional que representa o Sol:

```
using UnityEngine;

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

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

2.2 COLETA DE LIXO

Ademais, uma função específica para gerenciar a coleta de lixo na cidade do jogo, ato que representa uma das *quests* que devem ser realizadas, foi criada. A seguir, está o script referente à função descrita:

```
using UnityEngine;

public class QuestColetaLixo : MonoBehaviour
{
    public GameObject[] lixos; // TAG
    public int numeroLixos;

    private string mensagemFinal =
      @"Parabéns! A coleta de lixo é essencial para respeitar o meio ambiente,
      evitar a disseminação de doenças e preservar a harmonia de uma comunidade.
      O descarte correto e a reciclagem reduzem a poluição e a pegada ambiental,
      contribuindo para a criação de um planeta mais sustentável. Pequenas ações
      fazem uma grande diferença para o futuro da nossa cidade e do planeta.";

    void Start()
    {
        numeroLixos = lixos.Length;
        ColetarLixo();
    }

    void ColetarLixo()
    {
        /*
          Verifica se ainda há lixos a serem coletados, parando quando eles acabam.
          No loop interno (for), verifica-se que ainda há l
          ixos e sua quantidade
        */
        do
        {
            for (int i = 0; i < lixos.Length; i++)
            {
                if (lixos[i] != null) // Se este lixo (i) ainda não foi coletado
                {
                    Destroy(lixos[i]);
                    lixos[i] = null;
                    numeroLixos--;
                    break;
                }
            }
        }
        while (numeroLixos > 0);

        if (numeroLixos == 0)
          Debug.Log(mensagemFinal);
    }
}
```

2.3 CONSTRUÇÃO DE ESTACIONAMENTO INTELIGENTE

Por fim, há a função responsável por efetuar uma das *quests* do jogo: de preencher um estacionamento com veículos elétricos, automatizados e conectados com as tecnologias da cidade. A seguir, está o script referente à função descrita:

```
using UnityEngine;

public class QuestEstacionamento : MonoBehaviour
{
    // Matriz bidimensional representando as vagas do estacionamento
    private GameObject[,] vagasEstacionamento;
    public GameObject carro;
    public GameObject onibus;
    public GameObject caminhao;

    // Botão de ativação da Quest
    public UnityEngine.UI.Button botaoPreencherEstacionamento;

    private string mensagemFinal =     private string mensagemFinal =
      @"Parabéns! O estacionamento inteligente que você implementou otimiza o
      uso do espaço, reduzindo o trânsito e o tempo de busca por vagas disponíveis.
      Com tecnologia de ponta, ele promove a sustentabilidade e a eficiência
      operacional. Além de proporcionar uma experiência mais prática e conveniente
      para os moradores da nossa cidade."

    void Start()
    {
        /*
          Cria a matriz que representa o estacionamento e o EventListener do botão
          ativador (que representa a quest)
        */
        vagasEstacionamento = new GameObject[6, 6];
        botaoPreencherEstacionamento.onClick.AddListener(PreencherEstacionamento);
    }

    void PreencherEstacionamento()
    {
        /*
          Preenche o estacionamento com veículos aleatórios, checando quais vagas
          estão vazias e selecionando o tipo de veículo com o uso de Random.Range()
        */
        for (int i = 0; i < 6; i++)
        {
            for (int j = 0; j < 6; j++)
            {
                if (vagasEstacionamento[i, j] == null)
```

```
                {
                    GameObject tipoVeiculo = null;
                    int veiculoEscolhido = Random.Range(0, 3);

                    switch (veiculoEscolhido)
                    {
                        case 0:
                            tipoVeiculo = carro;
                            break;
                        case 1:
                            tipoVeiculo = onibus;
                            break;
                        case 2:
                            tipoVeiculo = caminhao;
                            break;
                    }

                    Vector3 posicaoDaVaga = new Vector3(j*10, 0, i*10);
                    GameObject veiculo = Instantiate(
                      tipoVeiculo,
                      posicaoDaVaga,
                      Quaternion.identity
                    );
                    vagasEstacionamento[i, j] = veiculo;
                }
            }
        }

        Debug.Log(mensagemFinal);
    }
}
```

3. CONCLUSÃO

Em suma, foram implementadas, com base nos conteúdos 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 estes estão a rotação do dia e da noite, a iluminação inteligente, assim como as *quests* de coleta de lixo e a de construção de um 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.