Algoritmos e Lógica de Programação: aplicação da linguagem C# em Beyond Mars

Nomes dos integrantes:

Ana Clara

Gabriel Vazquez Mamede Diniz

Luiz Miguel de Toledo

SUMÁRIO

1. **Uso de função**

```
void processarCeu()
{
    // Gira o Sol de -90 a 270 graus ao longo do "dia"
    float rotacaoX = Mathf.Lerp(-90, 270, segundos / 86400f);
    luzDirecional.rotation = Quaternion.Euler(rotacaoX, 0, 0);
}
```

Processamento de céu;

```
void AtualizarTaxaHabitavel()
{
    int ativos = 0;

    // Conta quantos objetos estão ativos
    foreach (GameObject obj in objetosParaAtivar)
        if (obj.activeSelf)
            ativos++;

    // Calcula a porcentagem de habitabilidade
    float porcentagem = (float)ativos / objetosParaAtivar.Count * 100f;
    taxaHabitavel.value = porcentagem;

    if (porcentagem >= 100f)
    {
        Debug.Log("Marte está totalmente habitável!");
    }
```

Função que atualiza a taxa habitável;

```
1 referência
public void AddIron(int amount)
{
    ironCount += amount;
    UpdateIronUI();
}
2 referências
void UpdateIronUI() => ironText.text = ironCount.ToString();
```

Função que adiciona ferro e outros minérios;

```
void ActivateNextPanel()
{
    if (currentIndex >= solarPanels.Count)
        return;

    if (HasRequiredResources())
    {
        DeductResources();
        solarPanels[currentIndex].SetActive(true); // Liga o painel
        currentIndex++;
        StartCoroutine(ShowMessage("Painel criado com sucesso!", 1f));
    }
    else
    {
        StartCoroutine(ShowMessage("Recursos insuficientes!", 1f));
    }
}
```

Função que ativa o próximo painel

```
public void FecharTodos()
{
    mainCanvas.SetActive(false);
    historiaCanvas.SetActive(false);
    constructorCanvas.SetActive(false);
    Cursor.visible = false;
    Cursor.lockState = CursorLockMode.Locked;
    Time.timeScale = 1f;
}
```

Função para fechar todos os canvas;