Algoritmo e Lógica de Programação

Aplicações de funções no jogo INGLORIOUS

Função de alternar a luz

```
public void AlternarLuz()
{
    foreach (Light luz in lights)
    {
        if (luz != null && !lightSlider.isRecharging)
        {
            // Inverte o estado da luz
            luz.enabled = !luz.enabled;

            // Alterna o controle da luz no mapa
            controleMapa.luzLigada = luz.enabled;
        }
        else if (lightSlider.isRecharging)
        {
            luz.enabled = false;
            controleMapa.luzLigada = false;
        }
    }
      }
```

Explicação:

Script anexado aos botões do *canvas* do mapa do jogo. Tem como objetivo verificar e alterar o estado da luz, entre ligada e desligada.

Função da tela de carregamento

```
// Função pública para iniciar o carregamento da cena com base no ID
public void LoadScene(int sceneId)
{
    // Inicia a coroutine para carregar a cena de forma assíncrona
    StartCoroutine(LoadSceneAsync(sceneId));
}

// Coroutine que realiza o carregamento da cena de forma assíncrona
IEnumerator LoadSceneAsync(int sceneId)
{
    // Inicia o carregamento da cena de forma assíncrona
    AsyncOperation operation = SceneManager.LoadSceneAsync(sceneId);

    // Ativa a tela de carregamento
    loadingScreen.SetActive(true);

    // Enquanto o carregamento não estiver completo
```

```
    while (!operation.isDone)
    {
        // Calcula o progresso (normalizando com base em 0.9f)
        float progressValue = Mathf.Clamp01(operation.progress / 0.9f);

        // Atualiza visualmente a barra de carregamento
        loadingBarFill.fillAmount = progressValue;

        // Espera o próximo frame antes de continuar o loop
        yield return null;
    }
      }
```

Explicação:

Script anexado ao *Game Object* que manuseia o painel da tela de carregamento, juntamente ao botão “Jogar”, presente no menu. Tem como objetivo ligar o painel da tela de carregamento, fazendo com que o jogo carregue assincronamente a próxima cena por meio de uma corrotina, em relação a velocidade de carregamento do computador do jogador.

Ele também é utilizado para controlar o *fill* da barra de carregamento, fazendo ela acompanhar o processo.

Função de ligar o temporizador da barra de energia

```
void EnableTimer()
{
    if (!isRecharging && luz.enabled)
    {
        // Gasta energia
        timer -= Time.deltaTime;

        if (timer <= 0.01f)
        {
            timer = 0f;
            isRecharging = true;
            luz.enabled = false; // só desliga a luz, NÃO o gerador
            controleMapa.luzLigada = false;
            luzApagada = true;
        }
    }
    else if (isRecharging && luzApagada)
    {
        // Recarregando
        timer += Time.deltaTime;

        if (timer >= maxTime)
        {
            timer = maxTime;
            isRecharging = false;
```

```
            luz.enabled = true;
            luzApagada = false;
        }
    }

    lightBar.value = timer;
      }
```

Explicação:

Script anexado a todas as luzes manuseáveis presentes no jogo. Tem como objetivo verificar e controlar o estado da luz e da barra de energia de cada luz, presente no *canvas* do mapa.

O código verifica que, se o gerador já estiver ligado, a quantidade de energia disponível irá reduzir, fazendo com que, quando a barra chegue a 0, ela tenha que ser recarregada, impossibilitando de a luz ser ativada ou do jogador poder a controlar.

Função de patrulha dos robôs

```
void Update()
{
    // Se o gerador estiver ligado
    if (controleMapa.geradorLigado)
    {
        // Executa patrulha do inimigo
        PatrulhaInimigo();
        // Define a tag do robo como "Robo"
        gameObject.tag = "Robo";
    }
    else
    {
        // Se o gerador estiver desligado, define a tag como "Door"
        gameObject.tag = "Door";
    }
}

// Metodo responsavel por movimentar o robo entre os pontos de patrulha
void PatrulhaInimigo()
{
    // Se o robo chegou ao ponto de patrulha atual
    if (transform.position == patrolPoints[targetPoint].position)
    {
        // Avança para o próximo ponto
        IncreaseTargetInt();
    }
    // Move o robo em direção ao ponto de patrulha atual
    transform.position = Vector3.MoveTowards(transform.position,
patrolPoints[targetPoint].position, speed * Time.deltaTime);
}
```

```
// Metodo que avança para o proximo ponto de patrulha
void IncreaseTargetInt()
{
    targetPoint++;
    // Se chegou ao fim do array, reinicia para o primeiro ponto
    if (targetPoint >= patrolPoints.Length)
    {
        targetPoint = 0;
    }
      }
```

Explicação:

Script anexado aos robôs presentes na cena. Tem como objetivo controlar a patrulha de cada robô com relação aos seus *waypoints* (locais por onde o robô pode se locomover).

O código verifica as posições de cada *waypoint* e conta elas como um número inteiro, assim, sempre quando o robô atingir um determinado *waypoint*, o número inteiro em relação aos waypoints irá aumentar e, se o robô alcançar o máximo de waypoints possíveis, ele retorna ao seu *waypoint* de origem e o número é resetado para 0.

Função de abrir tablet

```
void AbrirTablet()
{
    if (Input.GetKeyDown(KeyCode.E))
    {
        // Se tem o tablet e o mapa está fechado
        if (comTablet && !mapaAberto)
        {
            // Mostra o mapa e o fundo escuro
            map.gameObject.SetActive(true);
            darkBg.gameObject.SetActive(true);
            mapaAberto = true;
        }
        // Se o mapa está aberto
        else if (mapaAberto)
        {
            // Esconde o mapa e o fundo escuro
            map.gameObject.SetActive(false);
            darkBg.gameObject.SetActive(false);
            mapaAberto = false;
        }
    }
      }
```

Script anexado ao jogador. Tem como objetivo verificar se o jogador possui o tablet, se o mapa não estiver aberto e se o jogador pressionar a tecla “E”, o *canvas* do mapa será ativado e a variável que verifica se o mapa está aberto ou não terá seu estado alterado para “verdadeira”. Porém, se o mapa já estiver aberto e o jogador pressionar a tecla “E” novamente, o canvas do mapa será desativado e a variável que verifica se o mapa está aberto ou não terá seu estado alterado para “falsa”.