Algoritmo e Lógica de Programação

Aplicação de estrutura de decisão:

Código verificando o estado das animações de movimento e de agachar.

```
// Verificando as animacoes

      if (movimento != Vector3.zero && estaAgachado)
      {
          anim.SetBool("estaAndandoAgachado", true);
          anim.SetBool("estaAndando", false);
          anim.SetBool("estaAgachado", false);
      }
      else if (movimento == Vector3.zero && estaAgachado)
      {
          anim.SetBool("estaAndandoAgachado", false);
          anim.SetBool("estaAgachado", true);
      }

      else if (movimento != Vector3.zero && estaAgachado &&
Input.GetKeyDown(KeyCode.C))
      {
          anim.SetBool("estaAgachado", false);
          anim.SetBool("estaAndando", true);
      }
```

Explicação:

Se o personagem estiver em movimento (movimento != Vector3.zero) e a variável “estaAgachado” for verdadeira, a animação do personagem andando agachado será verdadeira, enquanto as demais serão falsas.

Se não, se o personagem estiver parado e a variável “estaAgachado” for verdadeira, então a animação do personagem parado agachado será verdadeira, enquanto a de andar agachado será falsa.

Se não, se o personagem estiver em movimento, se a variável “estaAgachado” for verdadeira e se o jogador pressionar a tecla “C”, a animação de andar normalmente será verdadeira, enquanto a animação do personagem agachado será falsa.

Código da mecânica do personagem agachar.

```
void Agachar()
{
    // Se o jogador apertar a tecla "C", ele agacha
    if (Input.GetKeyDown(KeyCode.C))
    {
        estaAgachado = !estaAgachado;
        anim.SetBool("estaAgachado", estaAgachado);
        anim.SetBool("estaAndando", false);
    }

    // Se o jogador apertar a tecla "C" novamente, ele deixa de ficar
agachado
    if (Input.GetKeyDown(KeyCode.C) && estaAgachado)
    {
        anim.SetBool("estaAgachado", false);
        anim.SetBool("estaParado", true);
    }
}
```

Explicação:

Se o jogador pressionar a tecla “C”, o estado da variável “estaAgachado” será o oposto da atual (Se ela for verdadeira, a mesma ficará falsa e vice e versa), a animação do personagem agachado vai ser de acordo com a variável “estaAgachado” e a animação de movimento será falsa.

Porém, se o jogador pressionar a tecla “C” e a variável “estaAgachado” for verdadeira (ou seja, se o personagem estiver agachado), a animação do personagem agachado será falsa e a animação do personagem parado em pé será ativada.

Aplicação da estrutura de repetição:

Código para verificar se a variável “luz” no array “luzes” está acesa ou apagada.

```
public class AtivarLuz : MonoBehaviour
{
    // Variavel do tipo Luz
    public Light luz;
    // Variavel para determinar se a luz está ligada ou não
    private bool ligada = true;

    // Metodo que é utilizada em outro script para alterar a luz
    public void AlternarLuz()
    {
        // Muda o estado da variavel "ligada"
        ligada = !ligada;
        // Se a luz estiver acesa, a variavel ligada sera verdadeira
```

```
        luz.enabled = ligada;
    }

    // Metodo para verificar se a luz esta ligada
    public bool LuzEstaLigada()
    {
        // Se a referencia da luz existir e se a luz estiver ligada, ele
retorna true
        return luz != null && luz.enabled;
    }
      }
```

Explicação:

Se o método AlternarLuz() for chamado, a luz será ligada ou desligada.

Controle das luzes no mapa.

Código para alternar o estado da luz de acordo com a região atual da luz.

```
public class MapaLuzes : MonoBehaviour
{
    // Array de regioes do tipo AtivarLuz (outro script)
    public AtivarLuz[] regioes;

    // Metodo para mudar a luz da regiao
    public void AlternarLuzRegiao(int index)
    {
        // Ele pega o index da regiao e aplica o metodo AlternarLuz, criado
no script "AtivarLuz"
        regioes[index].AlternarLuz();
    }
      }
```

Explicação:

O código pega o index da regiao e aplica o metodo AlternarLuz, criado no script AtivarLuz.

Inteligência Artificial do robô.

Código que controle a Inteligência Artificial do Robô

```
public class IARobo : MonoBehaviour
```

```
{
    // Variavel que indica a velocidade de rotacao do robo
    [SerializeField]
    private float rotationSpeed;

    // Variavel que pega a luz da area do tipo script "AtivarLuz"
    public AtivarLuz luzDaArea;

    // 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()
    {
        // Rotacao do robo
        transform.Rotate(0, rotationSpeed * Time.deltaTime, 0);
    }

    // OnTriggerStay - Ele é chamado a cada frame enquanto algum objeto esta
dentro do collider do robo
    private void OnTriggerStay(Collider other)
    {
        // Se a referencia da luz existe e ela estiver ligada
        if (luzDaArea != null && luzDaArea.LuzEstaLigada())
        {
            // Se a colisao for do objeto com a tag "Player"
            if (other.CompareTag("Player"))
            {
                Debug.Log("Robô viu o jogador com a luz acesa!");
            }
        }
    }
      }
```

Explicação:

O robô fica rotacionando constantemente para verificar se o jogador está presente e, se o jogador entrar na área do robô e a luz estiver acesa, o jogador perde e o robô captura o jogador. Porém, se o jogador estiver na área do robô e a luz estiver apagada, ele consegue passar pelo robô e ir para a próxima área.

Inteligência Artificial do robô.

Código que controle a Inteligência Artificial do Robô

```
public class ControleMapa : MonoBehaviour
{
    // Variavel do tipo GameObject, para referenciar o tablet
    public GameObject tablet;
    // Variavel do tipo Image, para referenciar o mapa
    public Image map;
    // Variavel do tipo Image, para referenciar o fundo mais escuro quando o
mapa estiver ativado
    public Image darkBg;

    // Variavel que determina se o jogador esta com o tablet ou nao
    private bool comTablet;
    // Variavel que determina se o jogador esta com o mapa aberto ou nao
    private bool mapaAberto;


    // Start is called once before the first execution of Update after the
MonoBehaviour is created
    void Start()
    {
        // Referenciando o tablet
        tablet = GameObject.Find("Tablet");

        // Deixando os objetos não ativos
        map.gameObject.SetActive(false);
        darkBg.gameObject.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {
        // Chamando o metodo "AbrirTablet"
        AbrirTablet();
    }

    void AbrirTablet()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            if (comTablet && !mapaAberto)
            {
                map.gameObject.SetActive(true);
                darkBg.gameObject.SetActive(true);
                mapaAberto = true;
            }
            else if (mapaAberto)
            {
                map.gameObject.SetActive(false);
                darkBg.gameObject.SetActive(false);
                mapaAberto = false;

            }
        }
```

```
    }

    // Se o jogador entrar em colisao com o tablet, o tablet some e a
variavel "comTablet" fica verdadeira
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Tablet"))
        {
            tablet.gameObject.SetActive(false);
            comTablet = true;
        }
    }
      }
```

Explicação:

O código verifica se o jogador possui o tablet ou não e, se ele possui o tablet, o jogador pode abrir o mapa e controlar as luzes por ele.