Laboratório de Produção de Jogos 3 - Física e Simuladores

Objetivo: Neste laboratório iremos finalizar a lógica de disparo de projéteis e utilizar o padrão de projeto Singleton para criar metralhadoras.

Passo 1

Atirando - Pt 1 Uma vez que temos o alvo e o projétil "spawnado" podemos aplicar força nesse projétil para atingir o alvo. Passo a passo:

1. Adicione o seguinte código no script Metralhadora:

```
public class Metralhadora : MonoBehaviour
{
     /* Definicoes dos atributos */

    [Header("Ataque")]
    // Define que vamos atirar um projetil por segundo
    [SerializeField] private float fireRate = 1f;
    // Determina a proximo momento de disparo (ja comeca atirando)
    [SerializeField] private float fireCountdown = 0f;

    private void Update()
    {
        // Executa a logica somente se tivermos um alvo
        if (target != null)
        {
            /* Codigo de rotacao */

            // Verifica se eh hora de atirar
            if (fireCountdown <= 0f)
            {
                Atirar();
                // Atualiza a variavel fireCountdown para respeitar o fireRate
                fireCountdown = 1f/fireRate;
            }
            // Atualiza a variavel fireCountdown
            fireCountdown -= Time.deltaTime;
        }
    }

    private void Atirar()
    {
        // Coloque aqui a logica de instanciar o projetil (DESAFIO da lista
            passada)
    }

    /* Demais definicoes dos metodos */
}
```

2. Teste e veja se está tudo ok.

Passo 2

Atirando - Pt 2 Agora vamos movimentar o projetil para atingir o alvo. Passo a passo:

1. Crie um script chamado Projetil com o seguinte código:

```
public class Projetil : MonoBehaviour
{
     [SerializeField] private Transform target;

    [SerializeField] private float velocidade = 70f;


    public void BuscarAlvo(Transform umTarget)
    {
        target = umTarget;
    }

    // Implementa a logica de verificacao de alvo.
    // Se o alvo nao existir, destrua o projetil
    private void Update()
    {
        if (target == null)
        {
            Destroy(gameObject);
            return;
        }

        // Calcula a direcao para onde o tiro vai ser disparado
        Vector3 direcao = target.position - transform.position;

        //Calcula a distancia a ser percorrida float distanciaNesteFrame =
            speed * Time.deltaTime;
        float distanciaNesteFrame = velocidade * Time.deltaTime;

        // Verifica se acertamos o target
        if (direcao.magnitude <= distanciaNesteFrame)
        {
            AcertarAlvo();
            return;
        }

        // Se nao tivermos acertado o target, vamos fazer a movimentacao do
            projetil
        transform.Translate(direcao.normalized * distanciaNesteFrame,
            Space.World);

    }

    private void AcertarAlvo()
    {
        Destroy(gameObject);
    }

}
```

2. Adicione o script Projetil na prefab Projetil
3. Em seguida, adicione o seguinte código no script Metralhadora:

```
public class Metralhadora : MonoBehaviour
{
     /* Definicoes dos atributos */

    private void Atirar()
    {
        // Aqui a referencia para a prefab do projetil se chama
            projetilPrefab e sua posicao de spawn eh o Transform firePoint

        GameObject projetilGObject = (GameObject) Instantiate(projetilPrefab,
            firePoint.position, firePoint.rotation);
        Projetil projetil = projetilGObject.GetComponent<Projetil>();

        if (projetil != null)
        {
            projetil.BuscarAlvo(target);

        }

    }

    /* Demais definicoes dos metodos */
}
```

4. Teste e veja se está tudo ok. Você deverá ver o projétil sendo disparado e perseguindo o alvo.

Passo 3

Construindo Metralhadoras - Pt 1 Agora vamos aprender a constuir metralhadoras em cada um dos Nodos (ou tiles) do cenário. Passo a passo:

1. Crie um script chamado Nodo com o seguinte código para tratarmos o movimento do mouse:

```
 /* Este script vai controlar se existe ou nao uma metralhadora (torre) em
     um Nodo do cenario.
 Alem disso, ele vai controlar inputs do usuario como clique e passar o
     mouse por cima de um Nodo */
public class Nodo : MonoBehaviour
{
    [SerializeField] private Color hoverColor;
    private Color defaultColor;
    private Renderer rend;

    void Start()
    {
        // Pega o Renderer do Nodo
        rend = GetComponent<Renderer>();
        defaultColor = rend.material.color;
    }
```

```
15
16          void OnMouseEnter()
17          {
18           // Vamos alterar o material no Nodo quando passarmos o mouse por cima
19          rend.material.color = hoverColor;
20          }
21
22          void OnMouseExit()
23          {
24          // Volta com a cor padrao quando tirarmos o mouse de cima do Nodo
25          rend.material.color = defaultColor;
26          }
27      }
```

2. Coloque este script na prefab Nodo.
3. Preencha os atributos no Inspector.
4. Teste e veja se está tudo ok. Você deverá ver o material sendo trocado a medida que você passa por cima e sai de cima de um Nodo.

Passo 4

Construindo Metralhadoras - Pt 2 Agora criar uma classe para controlar a criação de novas Metralhadoras. Passo a passo:

1. Crie o script BuildManager com o seguinte código:

```
1   public class BuildManager : MonoBehaviour
2   {
3       // Aqui estamos implementando o padrao de projeto Singleton.
4       // A ideia eh que estamos garantindo que existe *somente um*
            BuildManager na cena.
5       public static BuildManager instance;
6
7       private GameObject metralhadoraAConstruir;
8
9       // Define qual a prefab vamos utilizar para construir uma nova
            metralhadora
10      private GameObject metralhadoraPadraoPrefab;
11
12      void Awake()
13      {
14          // Verifica se ja existe uma instancia de BuildManager na cena
15          if (instance != null)
16          {
17              Debug.LogError("Mais de uma instancia de BuildManager rodando!");
18              return;
19          }
20          instance = this;
21      }
22
23      void Start()
24      {
25          metralhadoraAConstruir = metralhadoraPadraoPrefab;
```

```
26          }
27
28          public GameObject GetMetralhadoraAConstruir()
29          {
30              return metralhadoraAConstruir;
31          }
32
33      }
```

2. Adicione este script no GameManager da cena.

Passo 5

Construindo Metralhadoras - Pt 3 Agora vamos tratar o clique de mouse em um Nodo para criação de uma metralhadora. Passo a passo:

1. Adicione no script Nodo o seguinte código:

```
1       public class Nodo : MonoBehaviour
2       {
3           /* Definicao de atributos */
4
5           [SerializeField] private GameObject metralhadora;
6
7           /* Definicao de metodos */
8
9          void OnMouseDown()
10         {
11          // Verifica se ja existe uma metralhadora no Nodo
12          if(metralhadora != null)
13          {
14              Debug.Log("Impossivel criar metralhadora, pois o Nodo já está
                    ocupado");
15              return;
16          }
17          // Constroi uma metralhadora
18          // Note que estamos utilizando o padrao de projeto Singleton aqui
19          GameObject metralhadoraAConstruir =
                BuildManager.instance.GetMetralhadoraAConstruir();
20
21          // Instancia a metralhadora
22          metralhadora = (GameObject) Instantiate(metralhadoraAConstruir,
                transform.position, transform.rotation);
23         }
24      }
```

2. Preencha os atributos no Inspector.

3. Teste e veja se está tudo ok. Você deverá instanciar uma metralhadora em Nodos que estiverem vazios.

Atividade 1

Currency Implemente a lógica de *currency* no jogo. A cada inimigo morto você ganha X dinheiros e uma torre custa Y dinheiros para constuir.

Atividade 2

Health Points Implemente a lógica de *healthPoints* dos inimigos e a lógica de dano de ataque das metralhadoras.

Atividade 3

Game Over Implemente a lógica de "corações" do player. A cada inimigo que chega no destino o player deve perder um coração.