08

Refatoração do Jogador

Agora vamos refatorar o nosso Jogador, só que com algumas modificações diferentes do Inimigo principalmente na parte de animação.

No Jogador temos uma parte de movimentação que é só dele (nenhum outro personagem vai utilizar), que é a rotação pelo mouse.

Então para não colocar isso no nosso código que valeria para todos os personagens, vamos criar um novo código chamado **MovimentoJogador** e vamos colocá-lo no Jogador.

Só que não seria bom aproveitar a movimentação do outro código?

Para isso vamos utilizar um conceito conhecido como Herança, que é bem útil para reaproveitar essa parte do código. Então, nesse *script*, vamos fazer o seguinte:

```
public class MovimentoJogador : MovimentoPersonagem
{
}
```

Pronto! Agora seu código pode fazer tudo que o outro código tem. Vamos passar a parte de rotacionar o mouse para cá, dentro de um método que recebe como parâmetro uma *LayerMask*.

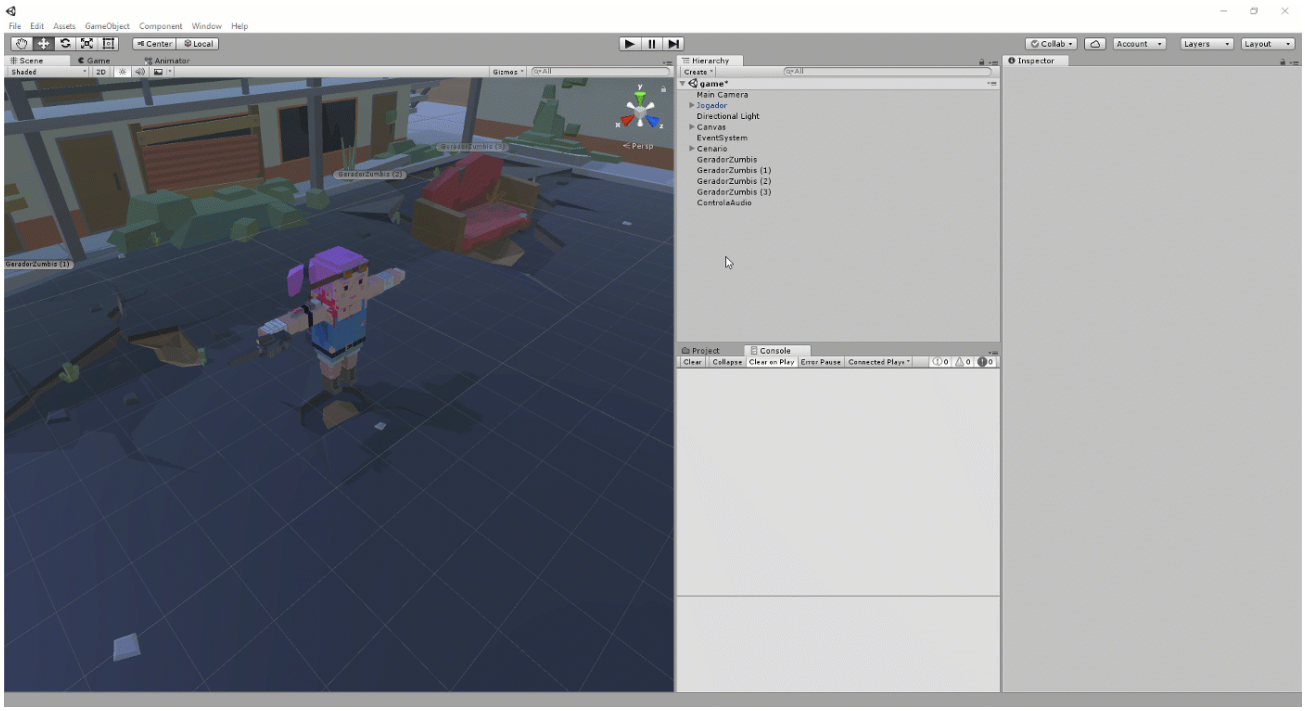
```
public void RotacionarJogador (LayerMask MascaraChao)
{
    Ray raio = Camera.main.ScreenPointToRay(Input.mousePosition);
    Debug.DrawRay(raio.origin, raio.direction * 100, Color.red);

    RaycastHit impacto;

    if(Physics.Raycast(raio, out impacto, 100, MascaraChao))
    {
        Vector3 posicaoMiraJogador = impacto.point - transform.position;

        posicaoMiraJogador.y = transform.position.y;

        Quaternion novaRotacao = Quaternion.LookRotation(posicaoMiraJogador);

        rigidbodyJogador.MoveRotation(novaRotacao);
    }
}
```

E que tal trocar essa parte da rotação para a nova que fizemos na refatoração do inimigo?

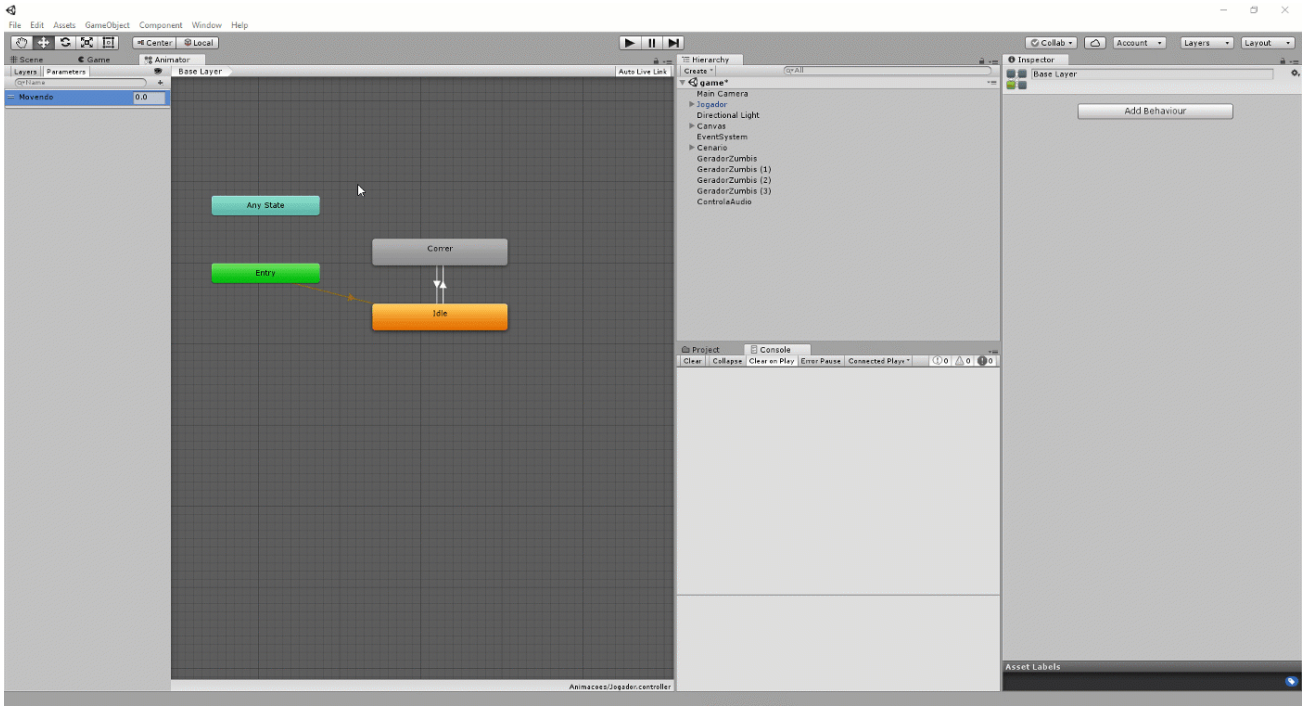
```
public void RotacionarJogador (LayerMask MascaraChao)
{
    Ray raio = Camera.main.ScreenPointToRay(Input.mousePosition);
    Debug.DrawRay(raio.origin, raio.direction * 100, Color.red);
```

```
        RaycastHit impacto;

        if(Physics.Raycast(raio, out impacto, 100, MascaraChao))
        {
            Vector3 posicaoMiraJogador = impacto.point - transform.position;

            posicaoMiraJogador.y = transform.position.y;

            Rotacionar(posicaoMiraJogador);
        }
    }
```

Vamos agora trabalhar um pouco com a parte de animação, então no nosso Animator do Jogador troque o parâmetro de **Movendo** para `Float` .



E vamos mudar as transições: quando o jogador for rodar a animação de correr para se movendo é mais (greater) do que `0.1` , e a volta é menos (less) que `0.1` .



Legal agora vamos selecionar o script chamado **AnimacaoPersonagem** e vamos arrastá-lo para o Jogador.

Agora vamos fazer um método que casa com esse `float` que fizemos para a animação. Vamos pegar um float pela direção, pois quando estamos parados ela é próxima de zero. Só que a direção é um `Vector3`, então como vamos fazer pra transformar isso num `float`?

Vamos utilizar a Magnitude do Vetor que é o tamanho dele, a distância do início para o fim do mesmo.

Então vamos criar um método que muda o valor desse float lá no Animator e que recebe um `Vector3`.

```
public void AnimarMovimento (Vector3 direcao)
{
    animator.SetFloat("Movendo", direcao.magnitude);
}
```

Pronto! Agora é só ir no **ControlaJogador** criar as variáveis para referenciar esses *script*:

```
private MovimentoJogador movimentoJogador;
private AnimacaoPersonagem animacaoJogador;

private void Start()
{
    movimentoJogador = GetComponent<MovimentoJogador>();
    animacaoJogador = GetComponent<AnimacaoPersonagem>();
}
```

E no FixedUpdate chamar os métodos para movimentação:

```
movimentoJogador.Movimentar(direcao, status.Velocidade);

movimentoJogador.RotacaoJogador(MascaraChao);
```

E no Update o de animação:

```
animacaoJogador.AnimarMovimento (direcao);
```