04

Rotacionar o personagem

Vamos fazer o nosso personagem seguir o mouse? Porque aí com ele rotacionando a gente consegue mirar para depois atirar.

Agora como que eu vou fazer essa posição refletir na rotação do mouse personagem? Bom vamos fazer o seguinte, a partir da câmera vamos soltar um raio que vai apontar pra posição do mouse e a partir da posição que esse raio toca no chão a gente tem a posição que queremos que o nosso jogador olhe.

Primeiramente vamos criar o raio utilizando a variável do tipo `Ray` dando o nome de **raio**, vamos acessar a câmera pelo `Camera.main` que pega a câmera principal do nosso jogo e em seguida utilizamos o método `ScreenPointToRay` que faz a partir da tela apontar um raio para uma posição que será a posição do mouse utilizando `Input.mousePosition` , nosso código então fica assim:

Ray raio = Camera.main.ScreenPointToRay (Input.mousePosition);

Agora temos que temos o raio vamos ver ele na tela? Para isso vamos utilizar o `Debug` e o método `DrawRay` onde utilizamos a posição de origem e direção do raio, passando uma cor. Lembre-se de multiplicar a direção por um valor senão ela ficará algo muito pequeno.

Debug.DrawRay(raio.origin, raio.direction * 100, Color.red);

Vamos fazer o raio colidir com o chão para isto temos que ter uma variável do tipo `RaycastHit` que guarda o impacto que o raio tem com alguma coisa.

RaycastHit impacto;

Agora temos que lançar o raio e testar no que ele está colidindo para isso vamos utilizar o `if` e o método `Physics.Raycast` que lança um raio e liga ele com a variável `RaycastHit` , também vamos passar um valor até onde nosso raio irá em distância.

```
if(Physics.Raycast(raio, out impacto, 100))
{

}
```

Já temos o raio tocando nosso objetos vamos agora transformar isso numa direção? Para isso assim como no Inimigo temos que pegar uma posição baseada na posição em que estou e vamos salvar isto numa variável do tipo `Vector3` .

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

Lembre-se de zerar o Y dessa posição para a posição Y do personagem, assim ele não olha para cima e para baixo sempre para frente.

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

Agora assim como no Inimigo vamos gerar uma rotação a partir desta direção e passar para o **Rigidbody** do jogador.

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

    Quaternion novaRotacao = Quaternion.LookRotation(posicaoMiraJogador);
    GetComponent<Rigidbody>().MoveRotation(novaRotacao);
}
```

Assim seu código fica da seguinte forma:

```
public class ControlaJogador : MonoBehaviour
{

    public float Velocidade = 10;
    Vector3 direcao;

    // Update is called once per frame
    void Update()
    {

        float eixoX = Input.GetAxis("Horizontal");
        float eixoZ = Input.GetAxis("Vertical");

        direcao = new Vector3(eixoX, 0, eixoZ);

        if (direcao != Vector3.zero)
        {
            GetComponent<Animator>().SetBool("Movendo", true);
        }
        else
        {
            GetComponent<Animator>().SetBool("Movendo", false);
        }
    }

    void FixedUpdate()
    {
        GetComponent<Rigidbody>().MovePosition
            (GetComponent<Rigidbody>().position +
            (direcao * Velocidade * Time.deltaTime));

        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))
        {
            Vector3 posicaoMiraJogador = impacto.point - transform.position;

            posicaoMiraJogador.y = transform.position.y;

            Quaternion novaRotacao = Quaternion.LookRotation(posicaoMiraJogador);

            GetComponent<Rigidbody>().MoveRotation(novaRotacao);
        }
    }
}
```

E temos nosso jogador andando e atirando para todos os lados!

Mas veja que algumas vezes ele rotaciona um pouco esquisito, não?

Isso é porque o nosso raio está colidindo com vários objetos não só o chão então a referência de rotação algumas vezes ficará esquisita para corrigir este problema vamos fazer o raio colidir somente com o chão, para isso vamos separar o chão numa *Camada (Layer)* separada e falar que o raio só colide com aquela Camada.

No código basta criando no topo uma variável nova através da linha:

public LayerMask MascaraChao;

LayerMask é uma variável utilizada para criar essa separação de Camadas.

Para o nosso raio entender essa camada temos que colocar mais um item no `if` do Raycast fazendo ele entender nossa variável `MascaraChao` :

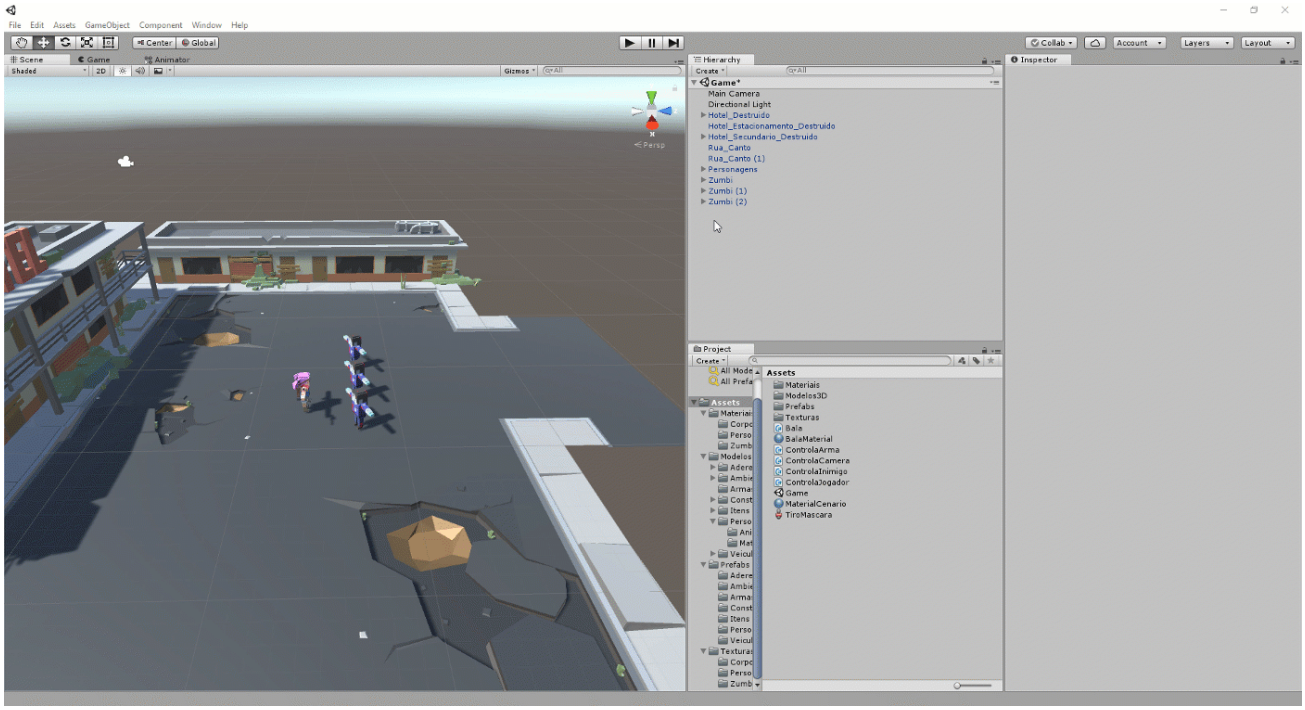
```
 if(Physics.Raycast(raio, out impacto, 100, MascaraChao))
{
    Vector3 posicaoMiraJogador = impacto.point - transform.position;

    posicaoMiraJogador.y = transform.position.y;

    Quaternion novaRotacao = Quaternion.LookRotation(posicaoMiraJogador);

    GetComponent<Rigidbody>().MoveRotation(novaRotacao);
}
```

Agora temos criar uma camada nova para o chão indo no topo do *Inspetor* ao selecionar o objeto `Hotel_Estacionamento_Destruido` , clique no item de **Layer** em seguida **Add Layer** e escreva o nome de uma nova camada, a camada **Chao** e coloque os objetos do chão do nosso cenário nesta camada.



Agora vamo fazer o nosso Raycast só reconhecer esta camada selecionando o personagem e marcando a variável `MascaraChao` no *Inspetor* somente com esta camada.