14

Movimentação com as setas

Transcrição

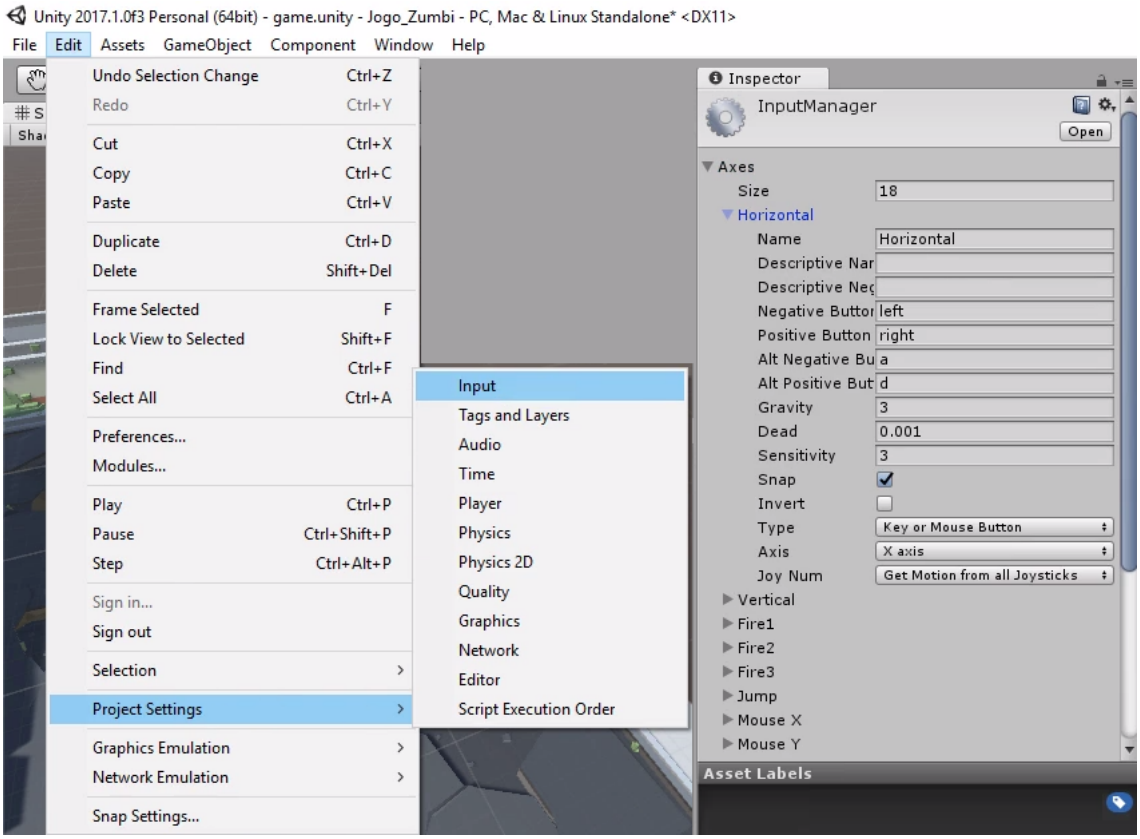
Adicionamos uma personagem ao cenário, mas ela só anda para frente. O objetivo é fazê-la se movimentar para todas as direções, utilizando as setas do teclado e o WSD. Dentro da Unity, faremos isso por meio do código, acrescentando `Input`, entre as chaves (`{}`) de `Update`, para acessar os métodos de entrada (teclado, mouse):

```
public class ControlaJogador : MonoBehaviour {


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

        Input.GetAxis("Horizontal");

        transform.Translate(Vector3.forward);

    }
}
```

Nesse trecho:

- adicionamos ponto (`.`) para entrar em `Input`;
- digitamos `GetAxis`, que pega o eixo;
- entre parênteses (`()`), indicamos o eixo (`"Horizontal"`), que por ser um texto, colocamos entre aspas (`"`).

Para entender o comando `GetAxis`, voltaremos à Unity e clicaremos em "Edit > Project Settings > Input". Em "Inpsector", abrirá "Axes" na qual encontramos funções prontas e com nomes definidos para usar no código. "Horizontal" é um exemplo delas.



Se analisarmos a função "Horizontal", veremos que a Unity definiu a ativação dela por meio da:

- seta para a esquerda (*left*);
- seta para a direita (*right*);
- tecla "A";
- tecla "D".

E "Vertical", outra que utilizaremos, será ativada por meio da:

- seta para baixo (*down*);
- seta para cima (*up*);
- tecla "S";
- tecla "W".

Assim, temos as setas e o WSD no mesmo lugar em "Vertical" e em "Horizontal" e podemos utilizá-las para movimentar a personagem. De volta ao código, acrescentaremos um trecho para " Vertical":

```
public class ControlaJogador : MonoBehaviour {


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

        Input.GetAxis("Horizontal");
        Input.GetAxis("Vertical");

        transform.Translate(Vector3.forward);

    }
}
```

Dessa forma, especificamos no código os comandos que farão o jogador se movimentar. Para que os trechos adicionados funcionem, precisamos guardá-los em uma **variável**, que armazena valores e expressões. Adicionaremos ela ao código:

- especificando o tipo `float`, referente à variáveis que permitem números decimais (por exemplo: 0.4, 0.5, -1.5);
- indicando o nome `eixoX`, para `"Horizontal"`, que terá valores decimais, pois a Unity definiu que o trânsito nesse eixo vai de `-1 a 0` e de `0 a 1`:
  - ao pressionar seta para esquerda ou "A", o eixo horizontal terá valor negativo (-1);
  - ao pressionar seta para direita ou "D", o eixo horizontal terá valor positivo (1);
  - se nenhuma tecla for pressionada, o eixo horizontal terá valor zero (0);
- atribuindo (`=`) o valor `Input.GetAxis("Horizontal")`.

Por variar de acordo com o pressionar das teclas, o valor de "Horizontal" deve ser armazenado em uma variável. Aplicaremos o mesmo para "Vertical". O código ficará da seguinte forma:

```
public class ControlaJogador : MonoBehaviour {


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

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

        transform.Translate(Vector3.forward);

    }
}
```

Capturamos os valores, guardamos nas variáveis `float eixoX` e `float eixoZ`. Podemos utilizá-las no código para substituir `Vector3.forward`. Porém, o `Vector3.forward` possui valores nos três eixos (X, Y e Z) e estamos utilizando somente o eixo Z. Com a inserção das variáveis, incluímos o eixo X. Precisamos transformar os números delas em um `Vector3`, especificando que o eixo:

- X terá o valor de `Input.GetAxis("Horizontal")`
- Y terá valor zero (0), pois não movimentaremos nada para cima ou para baixo;
- Z terá o valor de `Input.GetAxis("Vertical")`.

Para isso:

- adicionaremos a variável `Vector3` ao código, da mesma forma que fizemos com `float`, especificando primeiro o tipo;
- nomearemos como `direcao`, pois refere-se à direção que o jogador se moverá, de acordo com as teclas;
- atribuir (`=`) como valor `new Vector3(eixoX, 0, eixoZ)`, pois para `Vector3` é necessário definir entre parênteses os valores dos três eixos:
  - X, que preenchemos com a variável `eixoX`;
  - Y, que preenchemos com o valor zero (`0`), considerando que não o utilizaremos;
  - Z, que preenchemos com a variável `eixoZ`.

Assim, determinamos que a a personagem se movimentará de acordo com os valores dos eixos, especificados em `Vector3`, armazenado na variável `direcao`. Considerando esses fatores, em `transform`, substituiremos `Vector3.forward` por `direcao`:

```
public class ControlaJogador : MonoBehaviour {


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

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

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

        transform.Translate(direcao);

    }
}
```

Salvaremos as alterações utilizando o atalho "Ctrl + S" ou "Command + S" para Mac, minimizaremos o editor de texto, aguardaremos a Unity atualizar e clicaremos em "Play". Notem que se teclarmos a seta para cima ou "W", a personagem aparecerá andando para frente e se soltarmos as teclas, a personagem para, pois o valor do comando será zero para os três eixos.

Se testarmos as outras teclas, veremos que a personagem está se movendo de acordo com o que vimos em "Inpector" de "Axes", inclusive o WSD:

- "W" para frente;
- "S" para trás;
- "D" para a direita;
- "A" para a esquerda.

Pronto. A personagem está se movendo de acordo com as teclas que pressionamos no teclado.