Unity

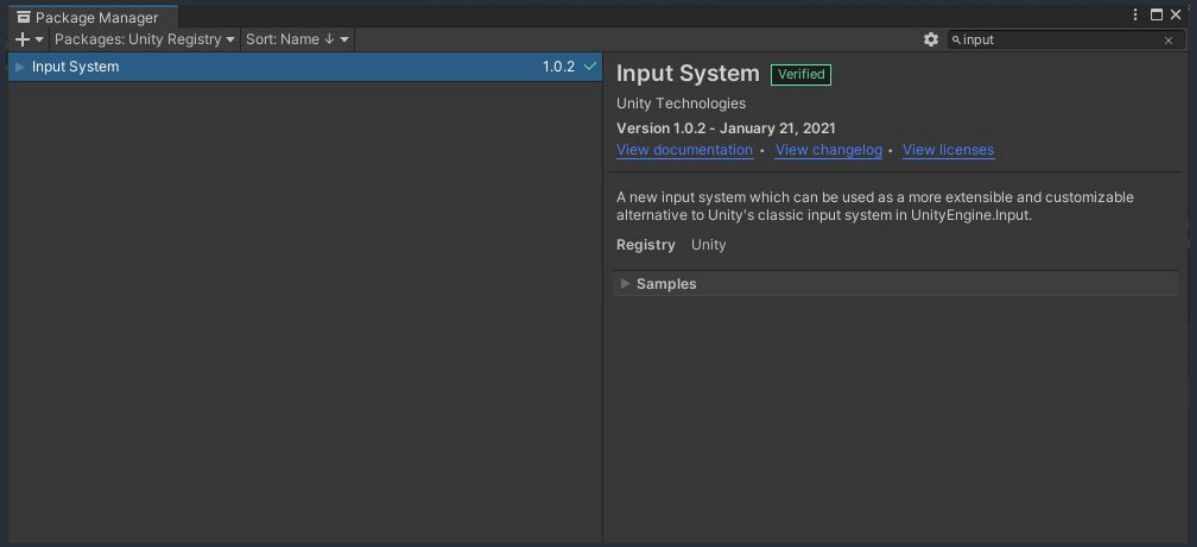
Jogo de aventura 3D: Adicionando Armas

Vamos dar mais atenção neste arquivo ao novo Sistema de Input da Unity.

Novo sistema de input

Com novo sistema de input que a Unity disponibiliza, podemos ler e receber inputs do jogador de maneira diferente. Mais organizada e mais fácil de se dar manutenção.

Para instalá-lo, basta ir ao **Package Manager**,
procurar por **Input System**, e instalar.



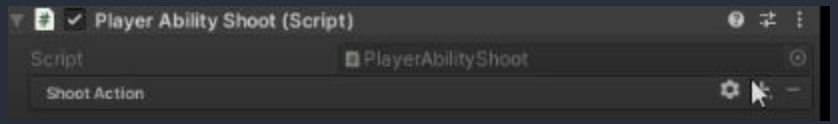
Para começar a utilizá-lo, temos basicamente duas maneiras de implementação: diretamente dentro da classe que queremos utilizar, ou utilizando um **arquivo** de configuração chamado **Input Actions**.

Para utilizar diretamente dentro da classe, devemos importar a biblioteca UNityEngine.InputSystem, e então, utilizamos a variável **InputAction**, como no exemplo abaixo.

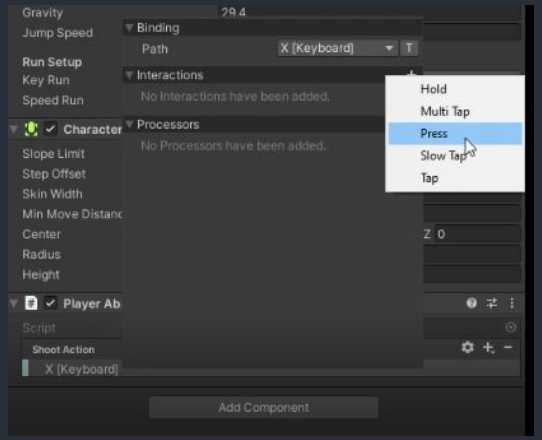
```
Unity Script | 0 references
public class PlayerAbilityShoot : PlayerAbilityBase
{
    public InputAction shootAction;

}
```

Isso fará com que o campo abaixo seja exposto no **Inspector**:

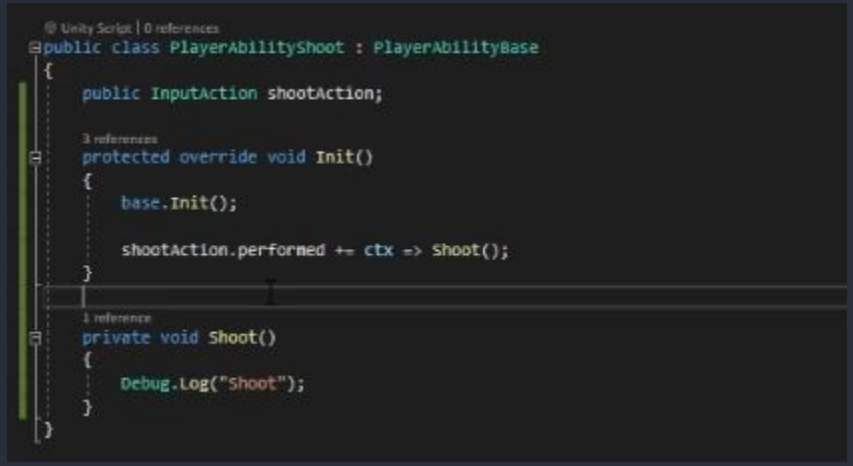


Agora, podemos clicar em adicionar um evento, e configurar o **binding** para tecla que queremos. Também precisamos adicionar ao menos uma **interaction**, que definirá como que a ação vai ser lida: press, hold, tap, etc.



Com isso, temos nossa configuração pronta.
Precisamos agora ler os eventos no código.

Para recebermos o evento quando a ação que configuramos foi performada, devemos ler o **perfomed** da seguinte maneira:



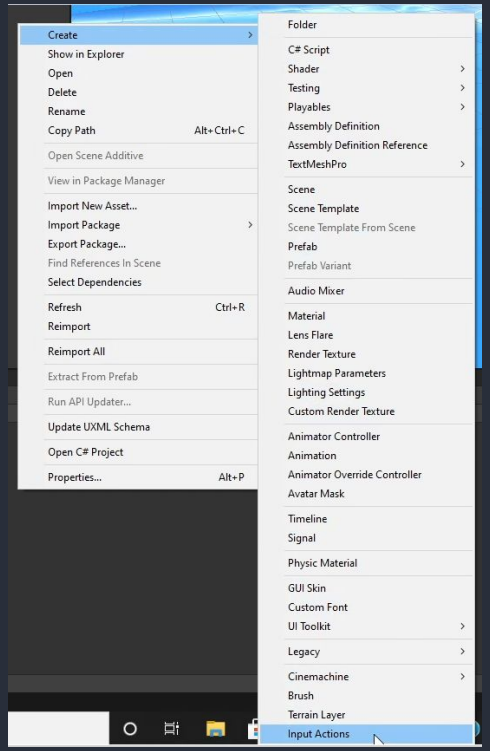
```
Unity Script | 0 references
public class PlayerAbilityShoot : PlayerAbilityBase
{
    public InputAction shootAction;

    3 references
    protected override void Init()
    {
        base.Init();

        shootAction.performed += ctx => Shoot();
    }

    1 reference
    private void Shoot()
    {
        Debug.Log("Shoot");
    }
}
```

Então, para recebermos o evento de ação cancelada *(por exemplo, quando a tecla é deixada de ser pressionada)*, ao invés do **perfomed**, usamos **canceled**.

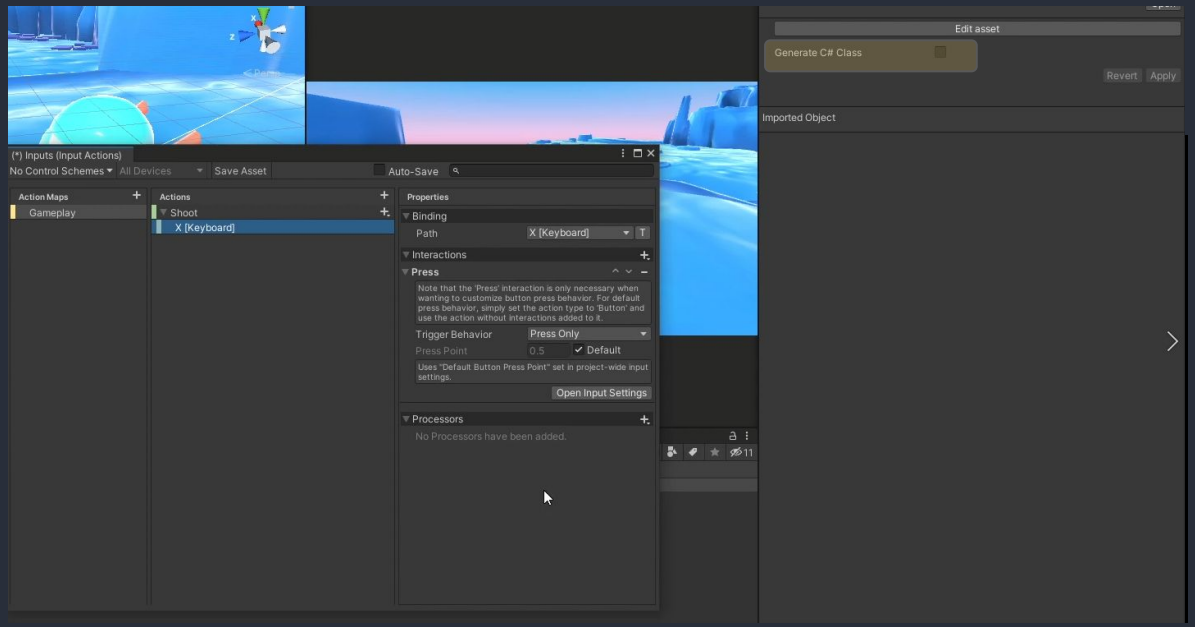


A segunda maneira que temos de configurar os inputs, é utilizando um arquivo **Input Actions.**

Para criá-lo, basta clicar com o botão direito no projeto, e ir em:
Create -> InputActions

Isso vai gerar um arquivo ***.inputactions**, que vai conter toda a configuração de mapeamento de teclas do jogo.

É importante deixar a opção **Generate C# Class ativada**, para o sistema gerar uma classe C# automaticamente, facilitando o acesso da configuração pelo código.



A classe que será gerada será do nome do arquivo que criamos. Se criarmos um arquivo chamado **Input.inputactions**, o nome da classe será **Input**.

Com isso, podemos acessá-la no código, usando exatamente o nome da classe:

```
protected Inputs inputs;
```

Precisamos sempre ativar e desativar os inputs quando estamos usando na classe. Dessa maneira, ao criar um novo input, podemos já ativá-lo, com ***.Enable();**.

Lembre-se de ativar e desativar usando as funções de OnEnable, e OnDisable do script. Assim, o input será sempre ativado quando o script estiver ativo, e sempre desativado quando o script estiver desativado, evitando possíveis bugs.

```
protected Player player;

protected Inputs inputs;

Unity Message | 1 reference
private void OnValidate()
{
    if (player == null) player = GetComponent<Player>();
}

Unity Message | 0 references
private void Start()
{
    inputs = new Inputs();
    inputs.Enable();

    Init();
    OnValidate();
    RegisterListeners();
}

Unity Message | 0 references
private void OnEnable()
{
    if(inputs != null)
        inputs.Enable();
}

Unity Message | 0 references
private void OnDisable()
{
    inputs.Disable();
}
```

Para acessar os eventos criados dentro do arquivo **Input.inputactions**, primeiro acessamos o mapa, e então o nome do evento. O resto, é igual ao mostrado no primeiro exemplo.

Aqui queremos acessar o mapa **Gameplay**, e o evento **Shoot**, então:

```
3 references
protected override void Init()
{
    base.Init();

    inputs.Gameplay.Shoot.performed += cts => StartShoot();
    inputs.Gameplay.Shoot.canceled += cts => CancelShoot();
}

1 reference
private void StartShoot()
{
    Debug.Log("Start Shoot");
}
1 reference
private void CancelShoot()
{
    Debug.Log("Cancel Shoot");
}
```

Documentação

Documentação e guia de começo para o novo sistema de input.

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/QuickStartGuide.html