Unity

Unity Extra

Extensões

Extensões são códigos que podemos criar para **extender** funções ou funcionalidades de objetos na Unity.

Por exemplo: quando precisamos escalar um objeto, precisamos fazer da seguinte maneira:

transform.localScale = Vector.one * scale;

Quando utilizamos extensões de código, podemos simplificar e ganharmos tempo de desenvolvimento. O código novo ficaria assim:

transform.Scale(scale);

Para desenvolvermos funções que servirão de extensões, a função deve:

- ser estática, para poder ser acessada de qualquer lugar
- receber como primeiro parâmetro o objeto que queremos que seja uma extensão, adicionando um **this** no começo do primeiro parâmetro
- preferencialmente uma classe apenas para essa finalidade

```
0 references
public static void Scale(this Transform t, float size = 1.2f)
{
    t.localScale = Vector3.one * size;
}

0 references
public static void Scale(this GameObject t, float size = 1.2f)
{
    t.transform.localScale = Vector3.one * size;
}

0 references
public static void ScaleVector(this Vector3 t, float size = 1.2f)
{
    //t.localScale = Vector3.one * size;
}
```

Com isso, conseguimos chamar a função **Scale** através de um Transform, GameObject e Vector.

```
public Vector3 vec;

Unity Message | 0 references
private void Awake()
{
    transform.Scale(2);
    gameObject.Scale(2);
    vec.ScaleVector(2);
}
```

Outro exemplo que podemos utilizar, é o de pegar um item randômico de uma lista.

É bem comum precisarmos fazer isso repetidas vezes dentro do nosso jogo. Dessa forma, utilizar uma função com extensões pode facilitar e melhorar nossos Workflow.

Para pegarmos um item randômico de uma lista, normalmente fazemos da seguinte maneira:

list[Random.Range(0, list.Count)].

Para utilizarmos extensões para listas e arrays de diversos tipos, podemos passar como parâmetro um tipo de componente indefinido T.

Dessa maneira, conseguimos usar listas com diversos tipos de componentes. Listas de Gameobjects, de scripts próprios, de Objetos, de Transforms, etc.

Abaixo 2 tipos de randomização, que podemos usar em extensões, para **Arrays** e **Listas.**

```
0 references
public static T GetRandom<T>(this T[] array)
{
    if (array.Length == 0)
        return default(T);

    return array[Random.Range(0, array.Length)];
}
1 reference
public static T GetRandom<T>(this List<T> list)
{
    return list[Random.Range(0, list.Count)];
}
```