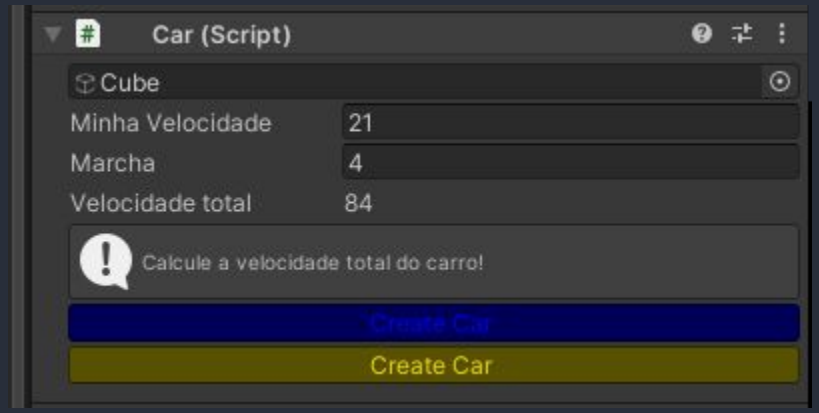
Unity

Unity Extra

Unity Editor Básico

É possível customizar os Inspectors na Unity, e criar novas janelas e editores. A Unity tem uma biblioteca feita exatamente para facilitar essa configuração.

Abaixo um exemplo simples do que é possível fazer no Inspector do script.



Para customizarmos o Inspector de um script, precisamos criar um script **Editor**, que vai ser responsável por cuidar da nova pintura do Inspector.

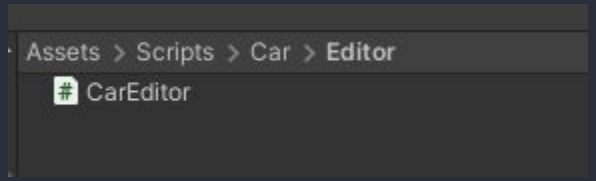
Dessa maneira, se temos um script **Car**, precisamos ter um script **CarEditor** (que extenderá de um **Editor**). Também precisamos falar que o script **CarEditor** será o **CustomInspector** do nosso script **Car.**

Veja o exemplo abaixo:

```
Unity Script | 3 references
public class Car : MonoBehaviour
{
    public GameObject carPrefab;


    public int speed = 20;
    public int gear = 5;

    2 references
    public int TotalSpeed
    {
        get { return speed * gear; }
    }

    2 references
    public void CreateCar()
    {
        var a = Instantiate(carPrefab);
        a.transform.position = Vector3.zero;
    }


}
```

```
[CustomEditor(typeof(Car))]
Unity Script | 0 references
public class CarEditor : Editor
{
    1 reference
    public override void OnInspectorGUI()...
}
```

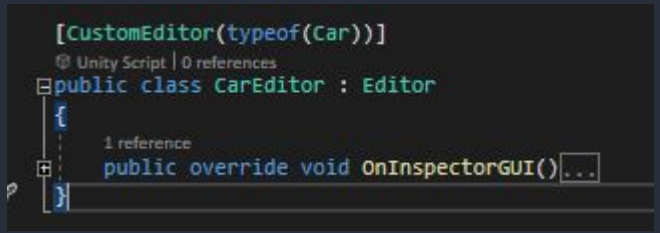
O script com **Editor** deverá sempre ficar dentro de uma pasta com o nome **Editor:**



Essa é a maneira que a Unity entende que este é um script de editor.

Dentro do CarEditor, a função mais importante que vamos utilizar é o **OnInspectorGUI**.

Essa função é a responsável por pintar o Inspector do script.



Para acessarmos o script principal, neste caso, o **Car**, devemos fazer um cast utilizando o objeto target:

```
1 reference
public override void OnInspectorGUI()
{
    //base.OnInspectorGUI();
    Car myTarget = (Car)target;
```

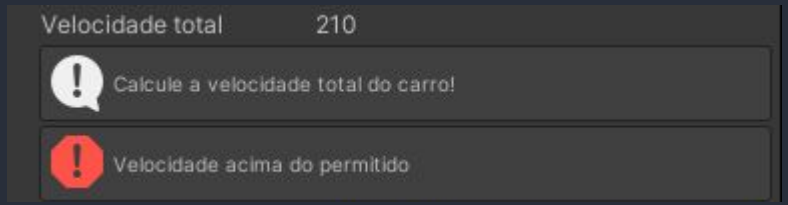
Com isso, conseguimos ter acesso a todos os elementos disponíveis no script **Car**.

Uma vez que não estamos utilizando o OnInspectorGUI default, comentado no exemplo abaixo, devemos pintar manualmente todos os itens que queremos que o inspector mostre.

```
//base.OnInspectorGUI();
Car myTarget = (Car)target;

myTarget.carPrefab = (GameObject)EditorGUILayout.ObjectField(myTarget.carPrefab, typeof(GameObject), true);

myTarget.speed = EditorGUILayout.IntField("Minha Velocidade", myTarget.speed);
myTarget.gear = EditorGUILayout.IntField("Marcha", myTarget.gear);
```

Podemos ainda utilizar componentes extras, como um InfoBox informativo, ou de erro. Dentro do inspector, podemos fazer contas e mostrar informações relevantes ao jogador:

```
EditorGUILayout.LabelField("Velocidade total", myTarget.TotalSpeed.ToString());

EditorGUILayout.HelpBox("Calcule a velocidade total do carro!", MessageType.Info);


if (myTarget.TotalSpeed > 200)
{
    EditorGUILayout.HelpBox("Velocidade acima do permitido", MessageType.Error);

}
```



Uma funcionalidade muito útil é a possibilidade de criarmos botões para acessar funções do nosso script.
Podemos ainda alterar a cor de toda GUI, utilizando **GUI.color**.

```
GUI.color = Color.blue;

if (GUILayout.Button("Create Car"))
{
    myTarget.CreateCar();
}

GUI.color = Color.yellow;

if (GUILayout.Button("Create Car"))
{
    myTarget.CreateCar();
}
```

Documentação

OnInspectorGUI
https://docs.unity3d.com/ScriptReference/Editor.OnInspectorGUI.html

EditorGUILayout *(para utilizarmos dentro do OnInspectorGUI)*
https://docs.unity3d.com/ScriptReference/EditorGUILayout.html