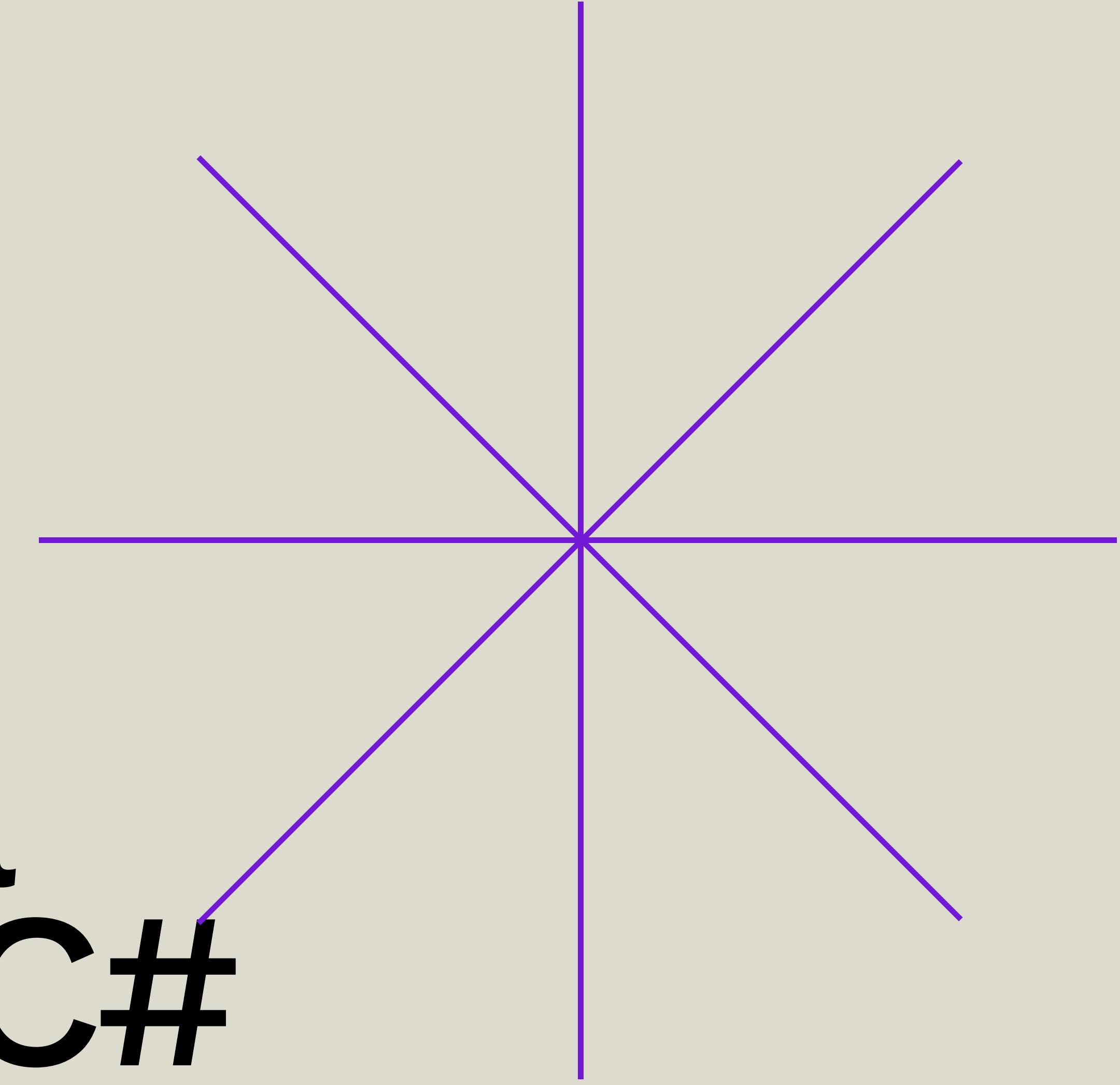
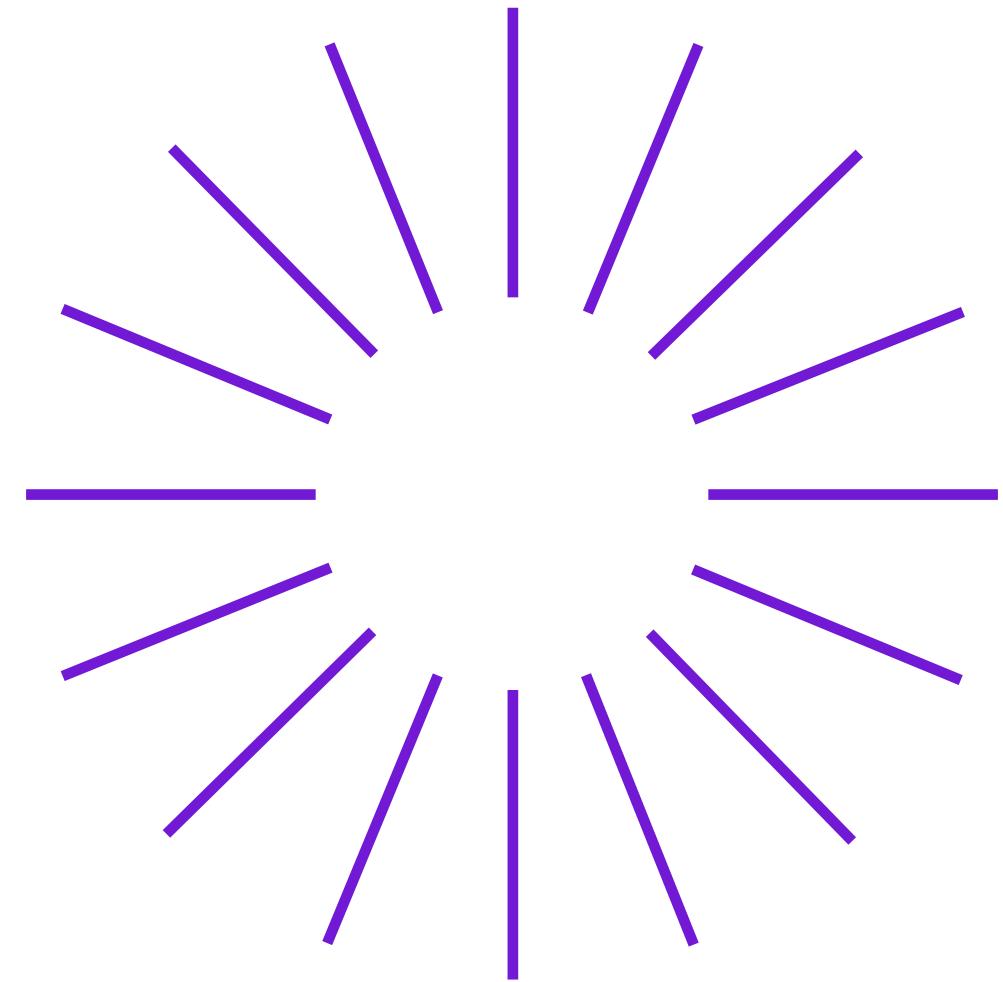


05. Básicos da Programação C# em Unity



ÍNDICE

- 01. O que é um Script
- 02. A Consola do Unity
- 03. Estrutura de um Script C#
- 04. Comentários
- 05. Variáveis
- 06. Métodos
- 07. Operadores Matemáticos
- 08. Comandos If e For



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

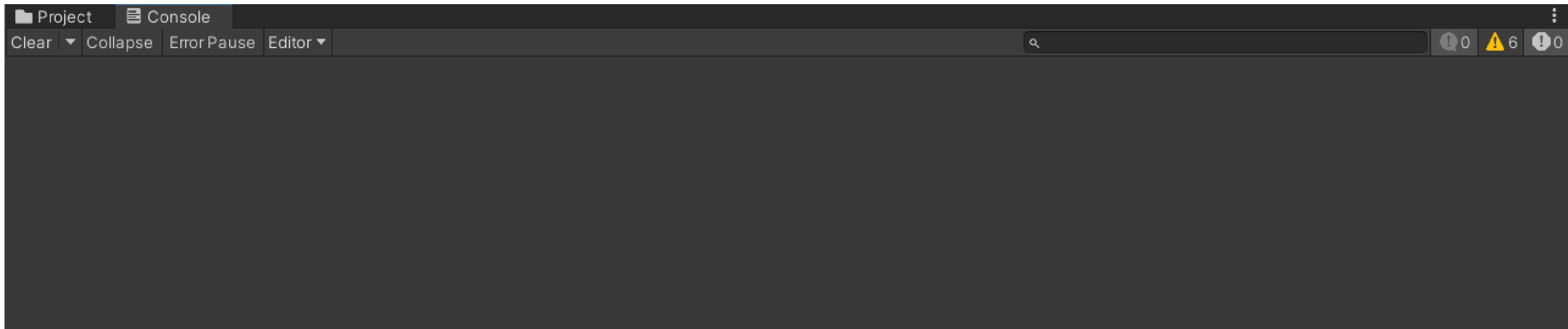
0 referências
public class MyFirstScript : MonoBehaviour
{
    // Start is called before the first frame update
    0 referências
    void Start()
    {
    }

    // Update is called once per frame
    0 referências
    void Update()
    {
    }
}
```

01. O QUE É UM SCRIPT?

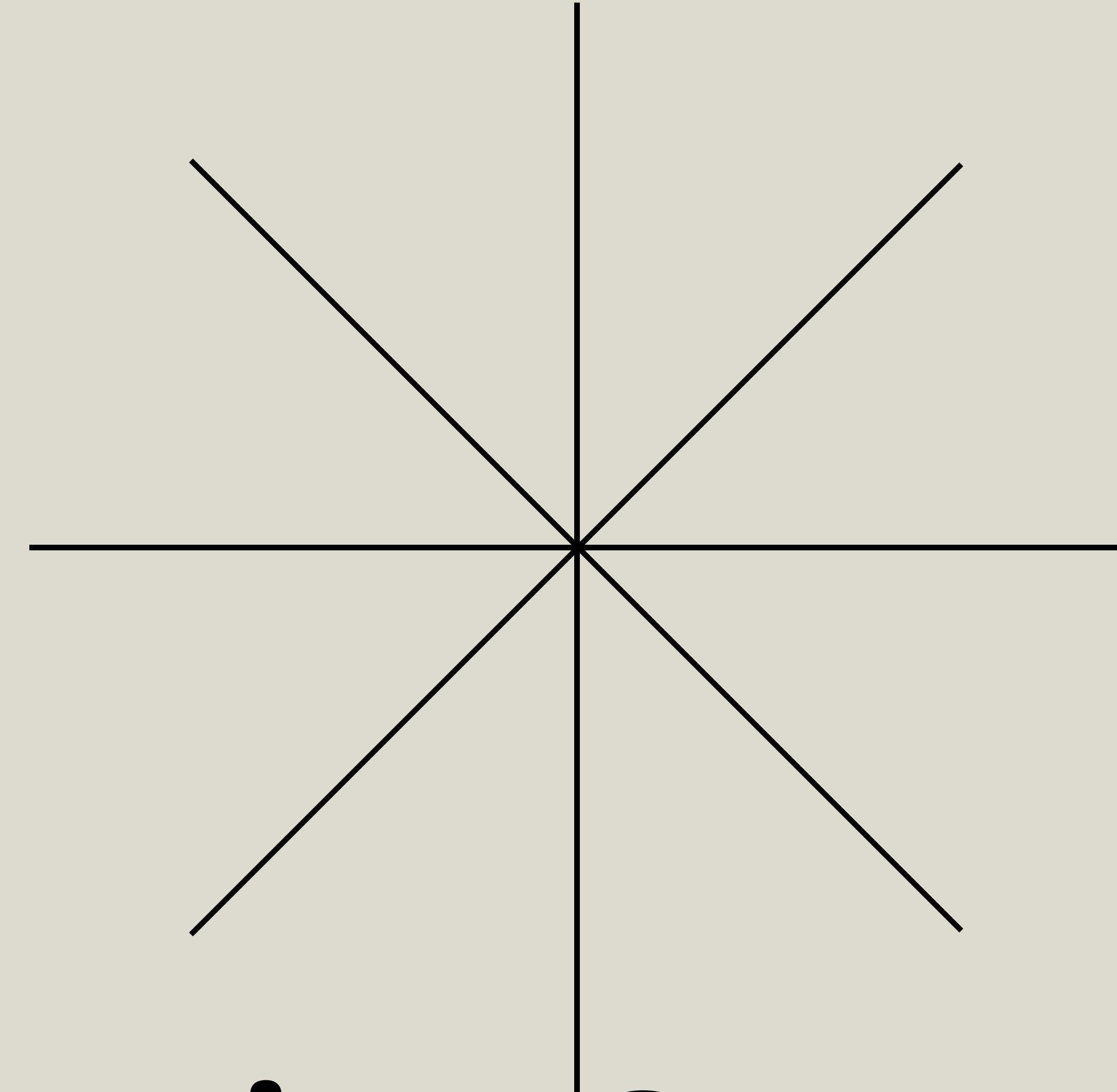
Um “Script” é um conjunto de instruções para que um programa execute determinadas funções.

02. A CONSOLA DO UNITY



A tab do Unity mais importante na programação, é através da consola que conseguimos detetar erros no código e até imprimir mensagens.

03 Estrutura do Script C#



Bibliotecas Importadas:

A nossa classe C#
(MyFirstScript):

Método Start dentro da
nossa classe:
(Só é executado 1 vez,
no início do nosso jogo)

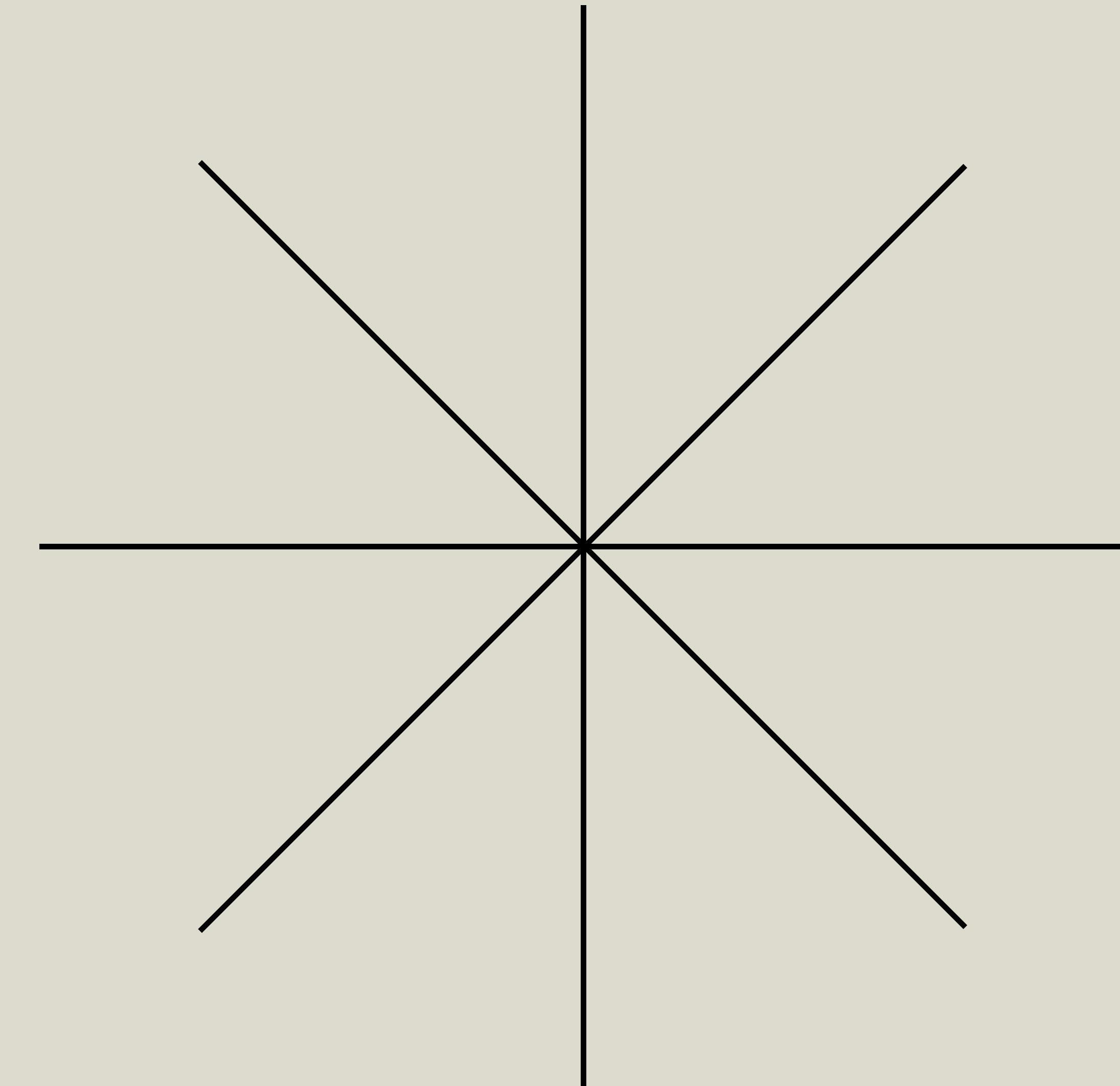
Método Update dentro
da nossa classe:
(é executado 1 vez a
cada frame)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 referências
public class MyFirstScript : MonoBehaviour
{
    // Start is called before the first frame update
    0 referências
    void Start()
    {
        ...
    }

    // Update is called once per frame
    0 referências
    void Update()
    {
        ...
    }
}
```

04 Comentários



CTRL + K + C

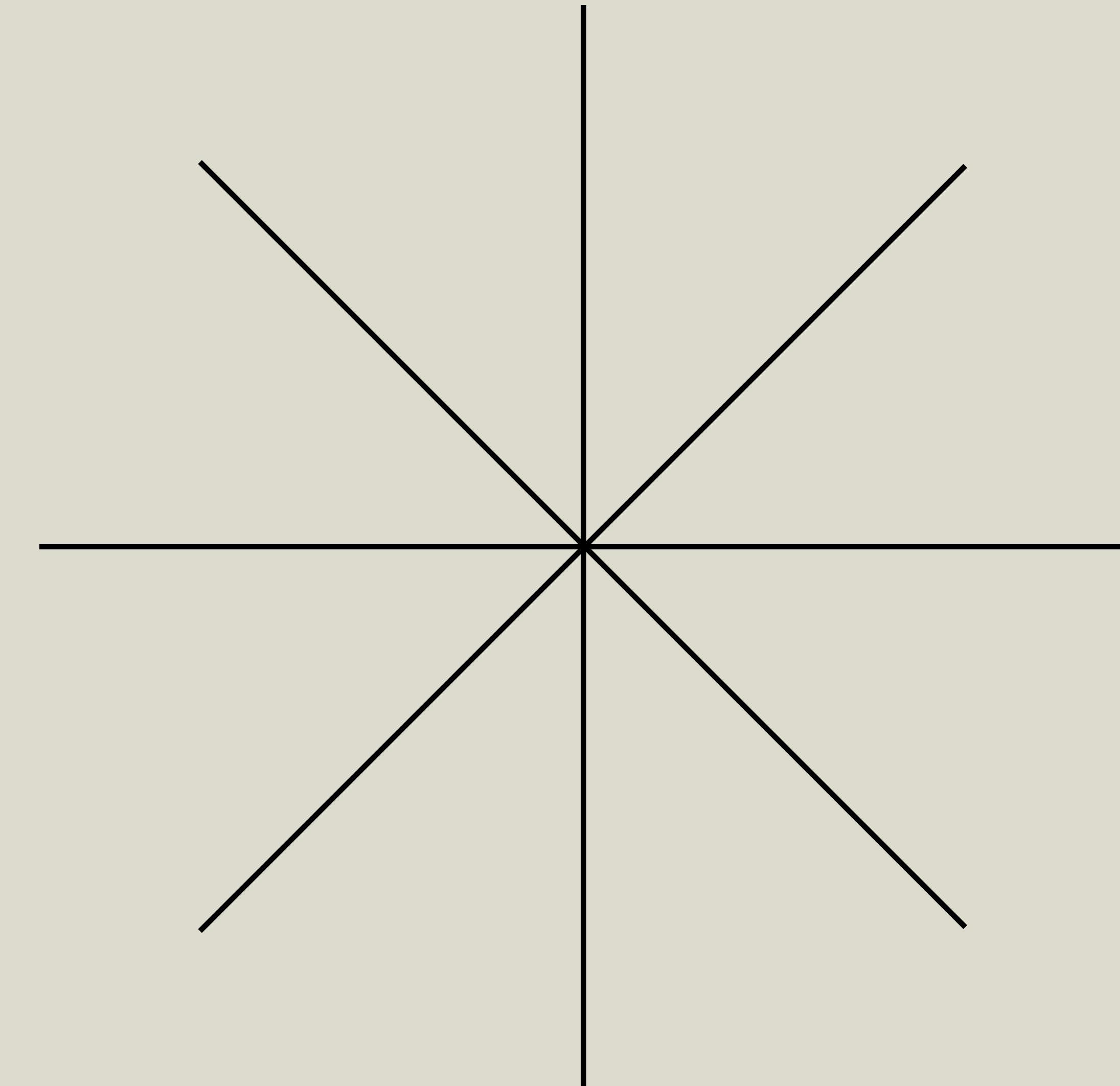
```
//indicação do seguinte código:  
/*  
já não preciso deste for  
for (int i = 0; i < 10; i++)  
{  
    myBool = false;  
    myWholeNumber++;  
    print("Hello World!");  
}  
*/
```

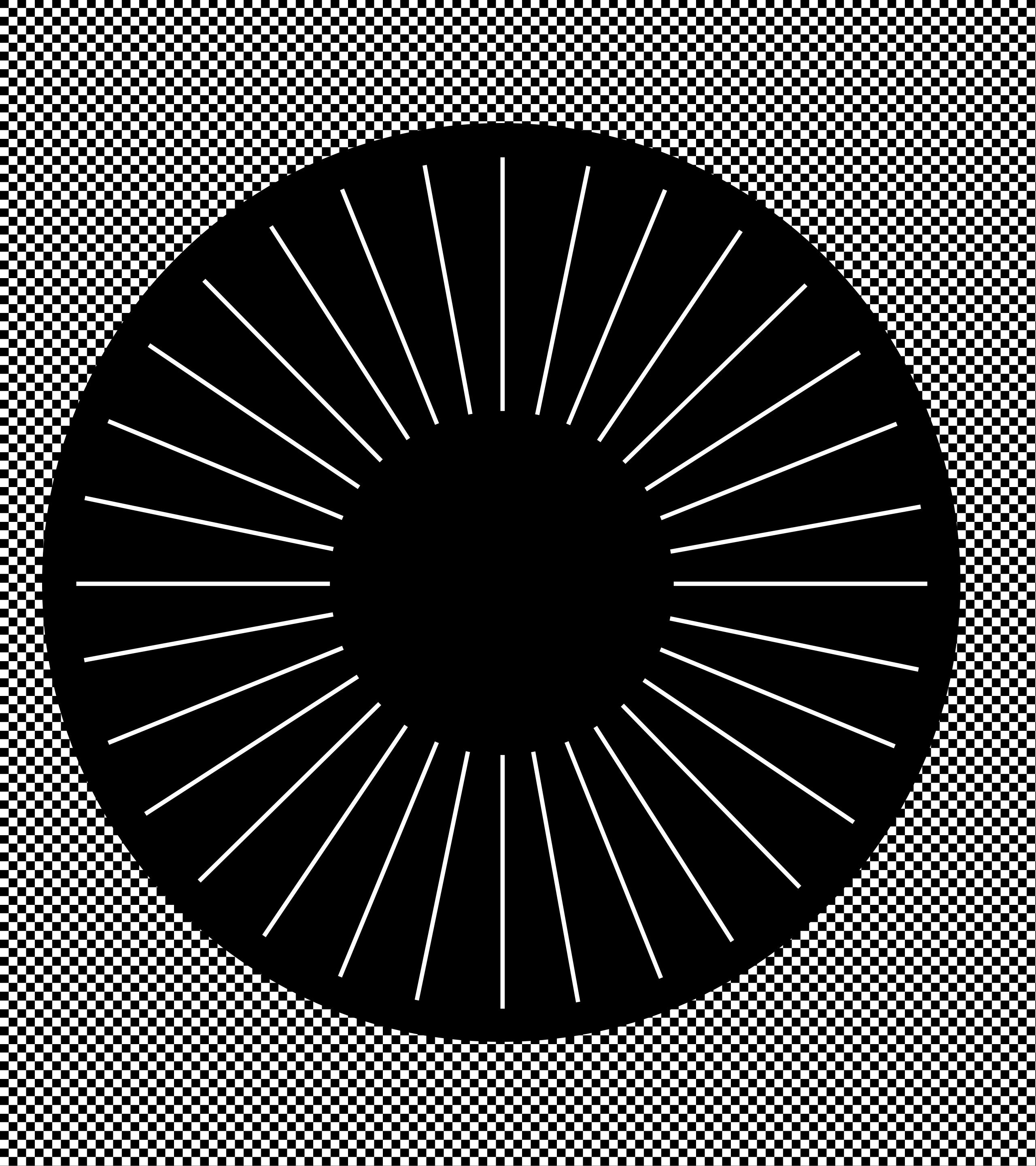
Para indicações acerca da utilidade de certas partes do código.

Tudo o que estiver em comentário não é executado!

ctrl + k + c para comentar e
ctrl + k + u para descomentar!

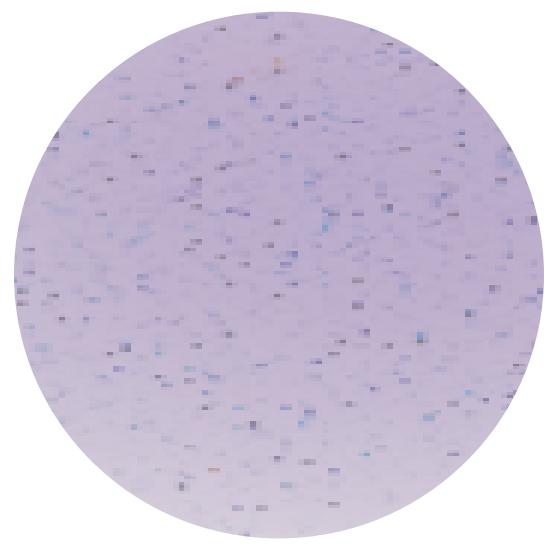
05 Variáveis





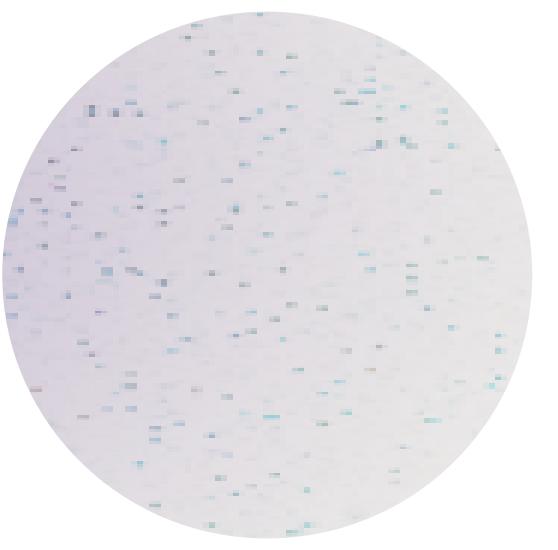
**UMA VARIÁVEL É UMA
FORMA DE GUARDAR
INFORMAÇÃO**

TIPOS DE VARIÁVEIS



int

Para
núme
ros
inteiros



double

Para
números
com
decimais



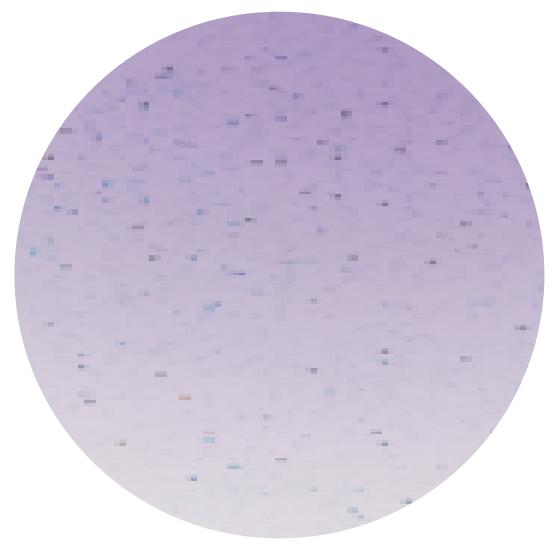
char

Para
caracteres,
e.g. 'a'; 'B'



string

Para texto,
e.g. "hello"



bool

Para valores
booleanos,
e.g. true;
false

Para serem utilizadas, variáveis devem ser instanciadas antes de qualquer método, ou seja, no início da classe.

Para ser instanciada, deve ser indicada o seu tipo e, de seguida o respetivo nome. (é boa prática nomear as variáveis segundo a denominação “camelCase”).

```
public class MyFirstScript : MonoBehaviour
{
    int myWholeNumber;
    double myDecimal;
    float myFloat;
    char myChar;
    string myString;
    bool myBool;
```

Depois de
instanciadas, um
valor já-lhes pode
ser atribuído.

```
void Start()
{
    myWholeNumber = 0;
    myDecimal = 0.1;
    myFloat = 0.1f;
    myChar = 'a';
    myString = "Hello World!";
    myBool = true;
```

Em C#, uma variável também pode ser pública ou privada.

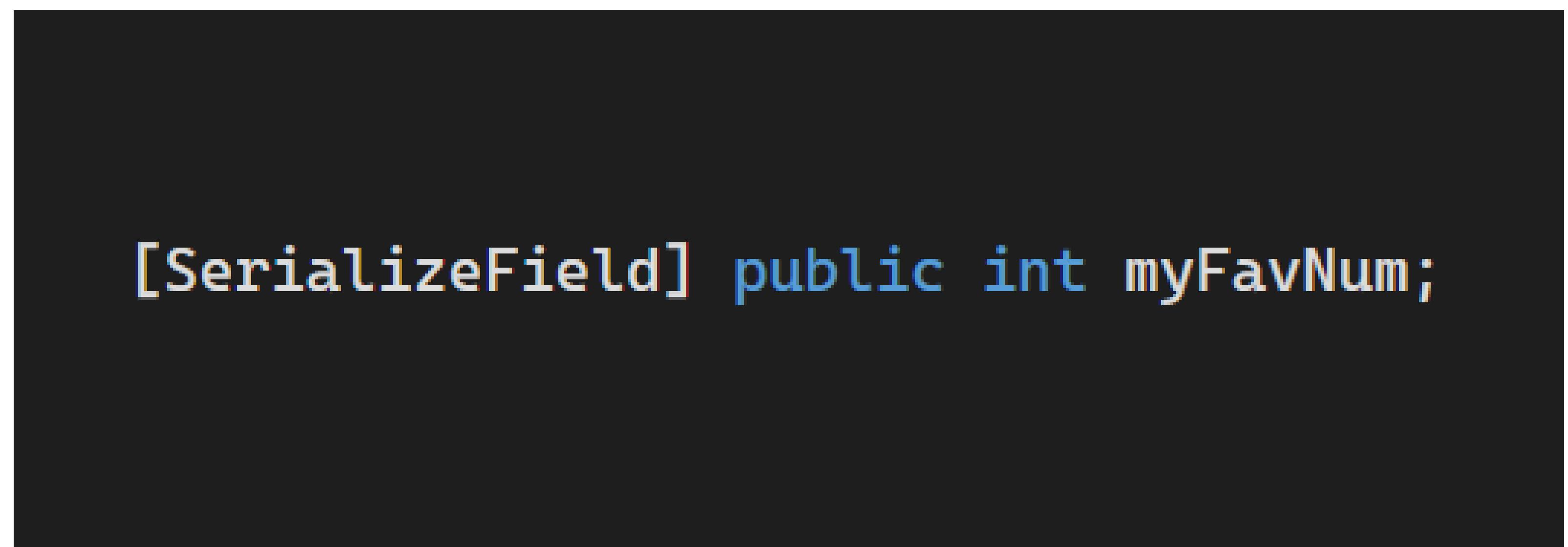
Se for pública, posso aceder a partir de qualquer outra classe/script do mesmo projeto.

Se for privada, só é acessível a partir da classe/script em que foi instanciada.

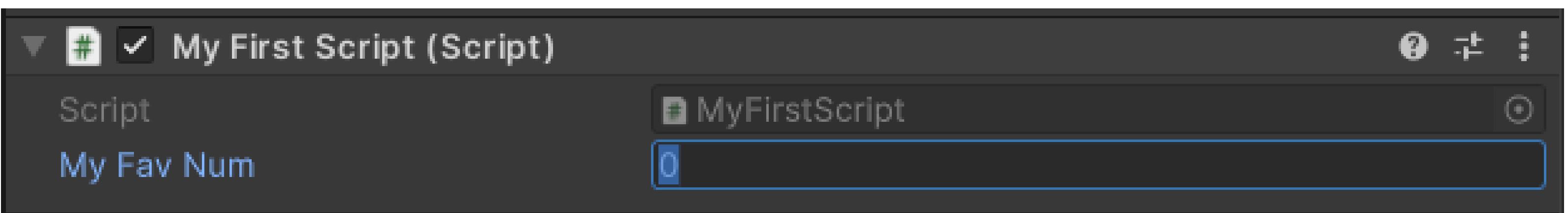
Por default, uma variável é privada.

```
public string myName = "John Doe, anyone can know!";
private string mySalary = "12€ :/, no one can know!";
```

[SERIALIZEFIELD]



Tornando uma variável serializável permite-nos mudar o seu valor através do editor do Unity!



TIPOS DE VARIÁVEIS DO UNITY

Existem tipos de variáveis específicos ao Unity.

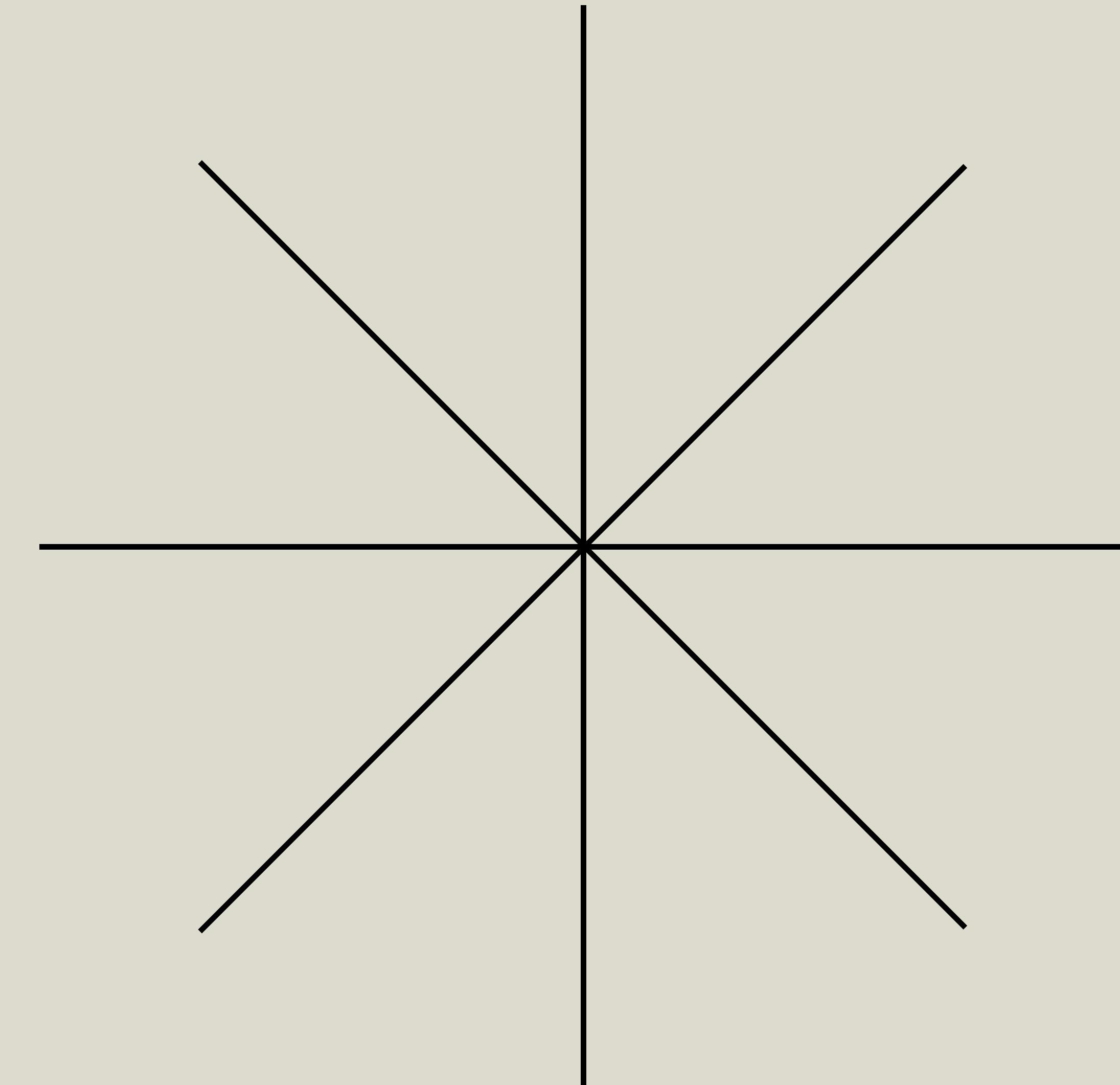
```
GameObject playerObject;  
Transform playerTransform;  
Animator playerAnimator;  
Rigidbody2D playerRigidBody;
```

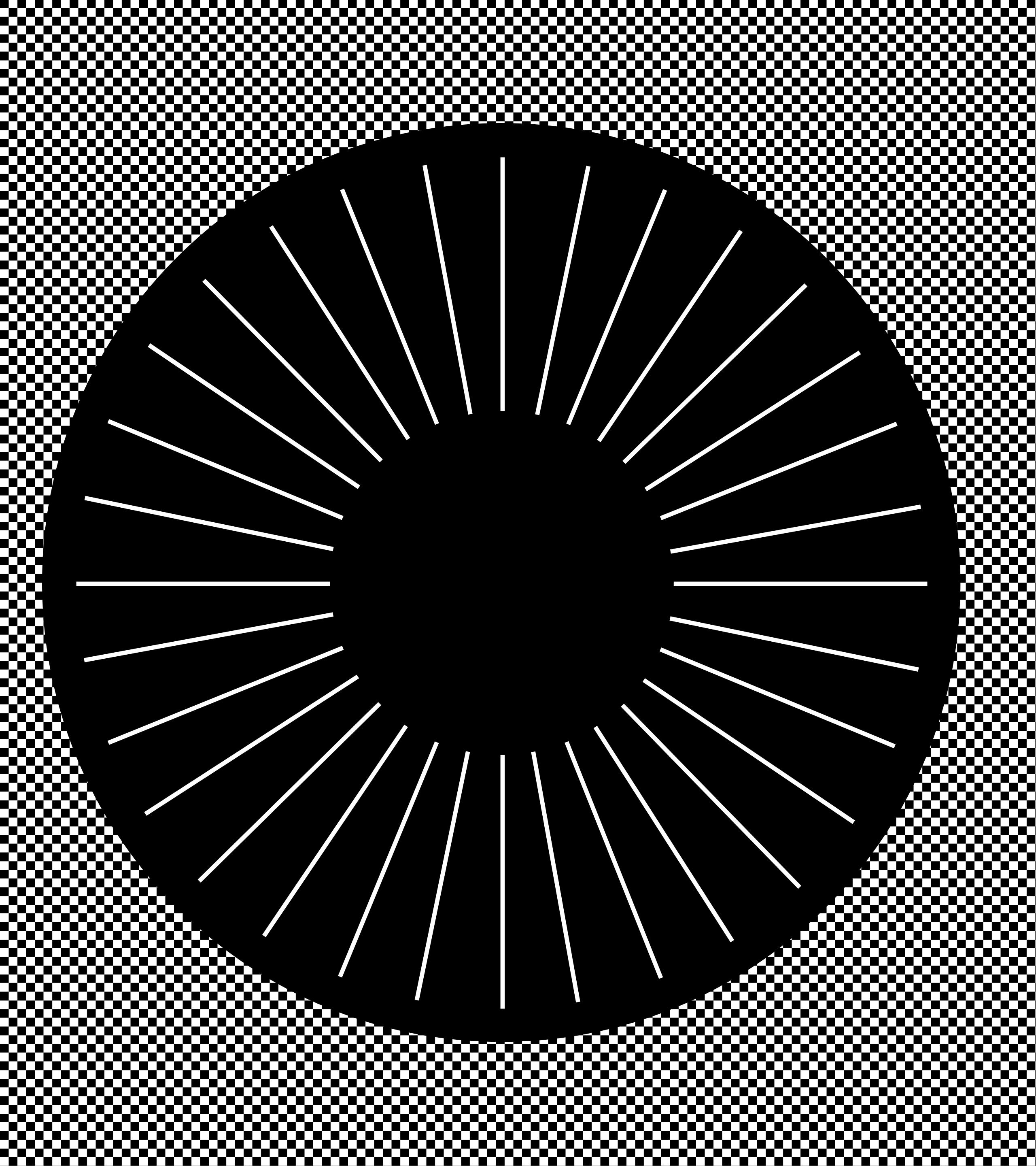
TIPOS DE VARIÁVEIS DO UNITY

Como estes tipos de variáveis estão tipicamente associados aos componentes dos gameObjects, ao atribuir-lhes um valor é necessário ir buscar esse componente.

```
playerObject = this.gameObject;
playerTransform = playerObject.transform;
playerAnimator = playerTransform.GetComponent<Animator>();
playerRigidbody = playerTransform.GetComponent< Rigidbody2D>();
```

06 Métodos





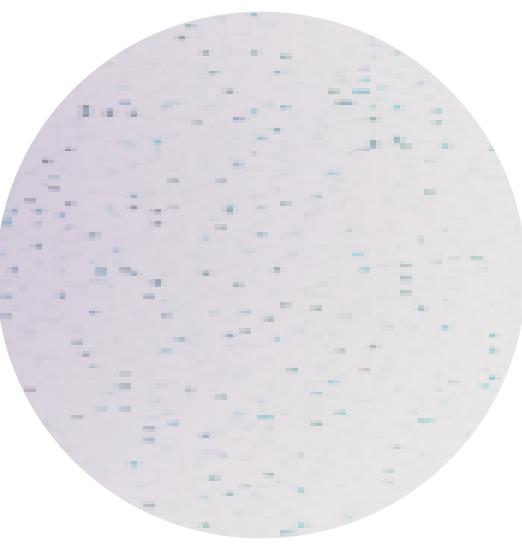
**UM MÉTODO É UM
CONJUNTO DE
INSTRUÇÕES
CONTIDA QUE PARA
SER EXECUTADO
DEVE SER CHAMADO.**

PULSE CHECK



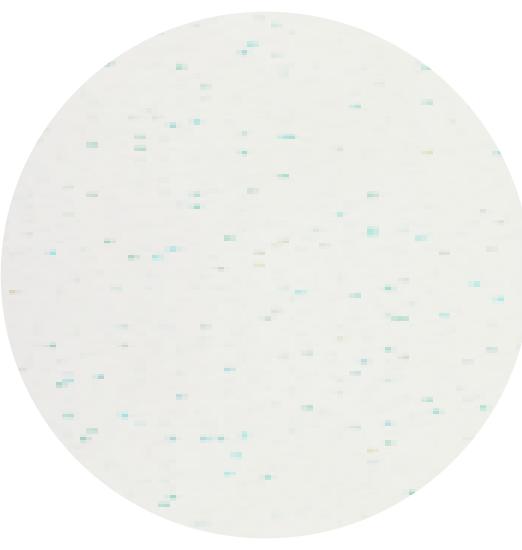
Assim como as variáveis, os métodos podem ser públicos ou privados e também têm tipos, de acordo com aquilo que retornam.

TIPOS DE MÉTODOS



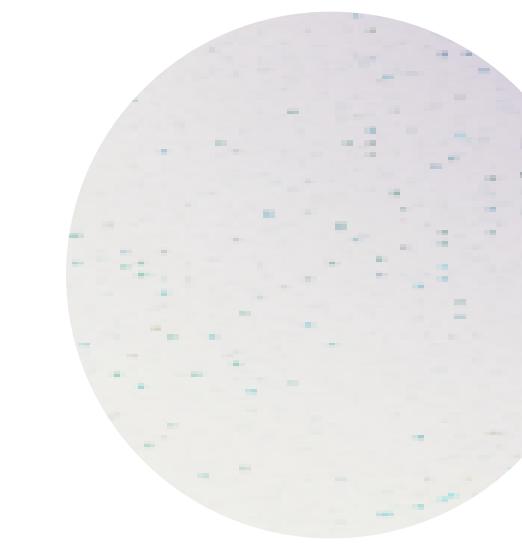
void

Não retorna
qualquer
valor



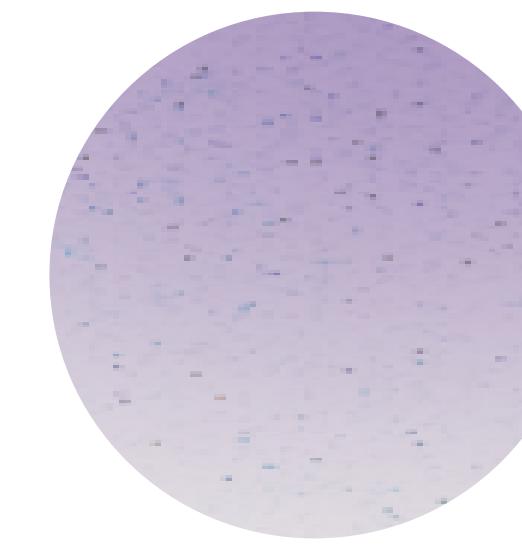
int

Apenas
retorna
valores int



string

Apenas
retorna
valores
string



bool

Apenas
retorna
valores
booleanos

COMO CRIAR UM MÉTODO

É boa prática que os nomes de métodos sejam capitalizados.

```
0 referências  
public void MyMethod()  
{  
}-  
}
```

COMO CHAMAR UM MÉTODO

Um método só corre no código quando é chamado. Por exemplo, ao ser chamado no ‘Start’ vai apenas correr uma vez.

• Mensagem do Unity | O referênci
void Start()
{
 MyMethod();
}

MÉTODO DO TIPO INT

Qualquer método,
que não do tipo
'void' deve sempre
retornar um valor, de
acordo com o seu
tipo.

```
1 referência
public int MyMethod()
{
    return 0;
}
```

MÉTODO DO TIPO INT

Ao ser chamado, neste caso, como retorna o valor 0, a sua utilidade será mudar o valor da variável ‘myWholeNumber’ para 0.

```
❶ Mensagem do Unity | 0 referências
void Start()
{
    |
    |
}
myWholeNumber = MyMethod();
```

EMBUTIR DADOS NUM MÉTODO

É possível transformar variáveis ou trabalhar com elas dentro de uma função.

```
1 referência  
public int MyMethod(int anyNumberIwant)  
{  
    //  
    return anyNumberIwant;  
}
```

EMBUTIR DADOS NUM MÉTODO

Ao ser chamado
deve ser passado os
parâmetros
necessários (neste
caso requer apenas
um número inteiro).

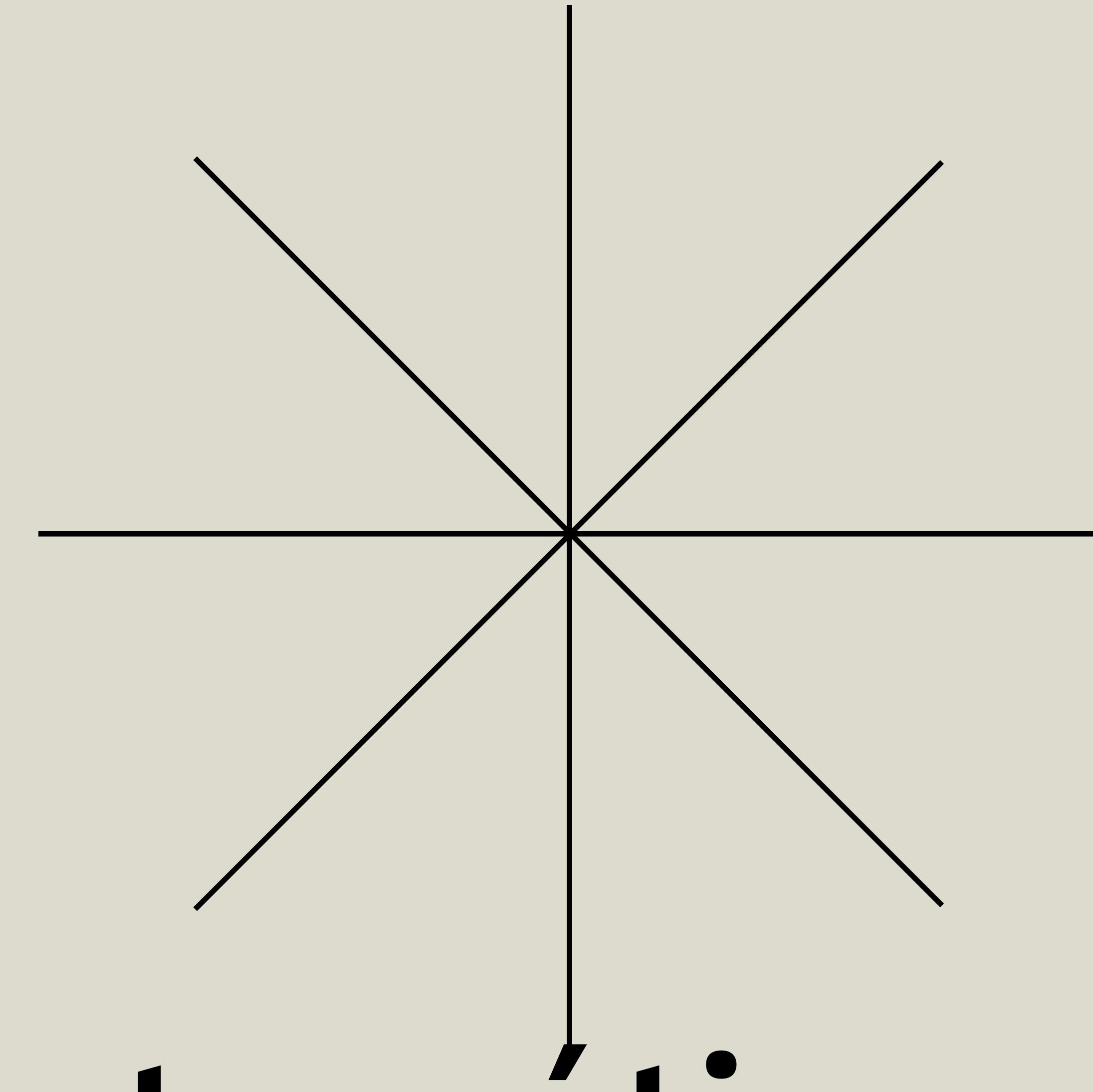
```
❷ Mensagem do Unity | 0 referências
void Start()
{
    myWholeNumber = MyMethod(5);
}
```

MÉTODO PRINT

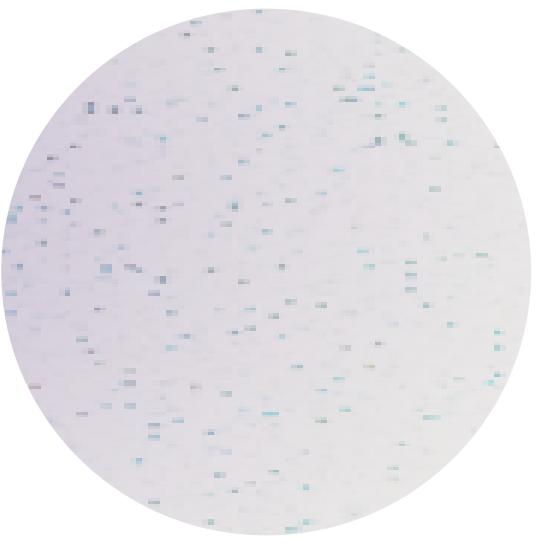
```
print("Hello World!");
print(myFavNum);
print("This is my fav number: " + myFavNum);
```

Um método onde é possível imprimir qualquer informação na consola.

07 Operadores Matemáticos

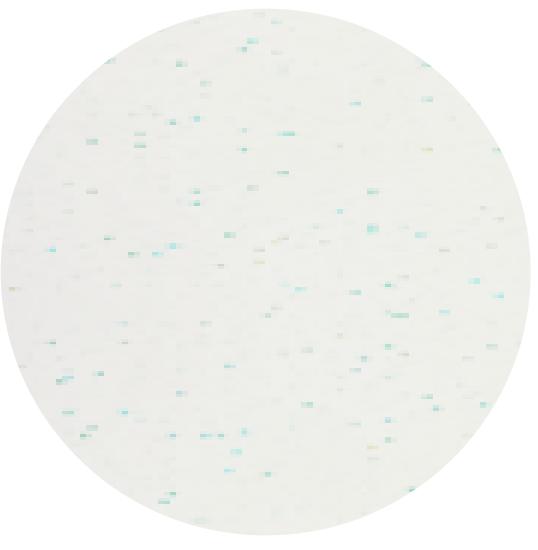


OPERADORES MATEMÁTICOS



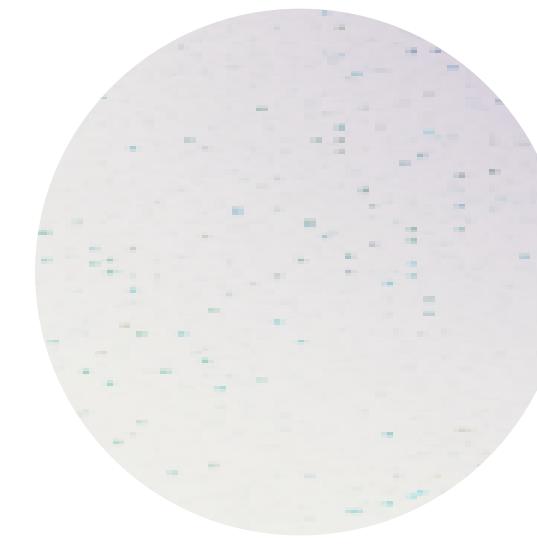
$==$

equivalente



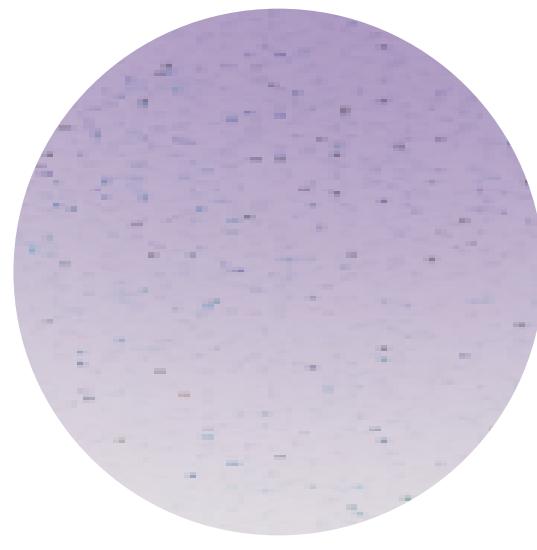
$!=$

diferente



$\&\&$

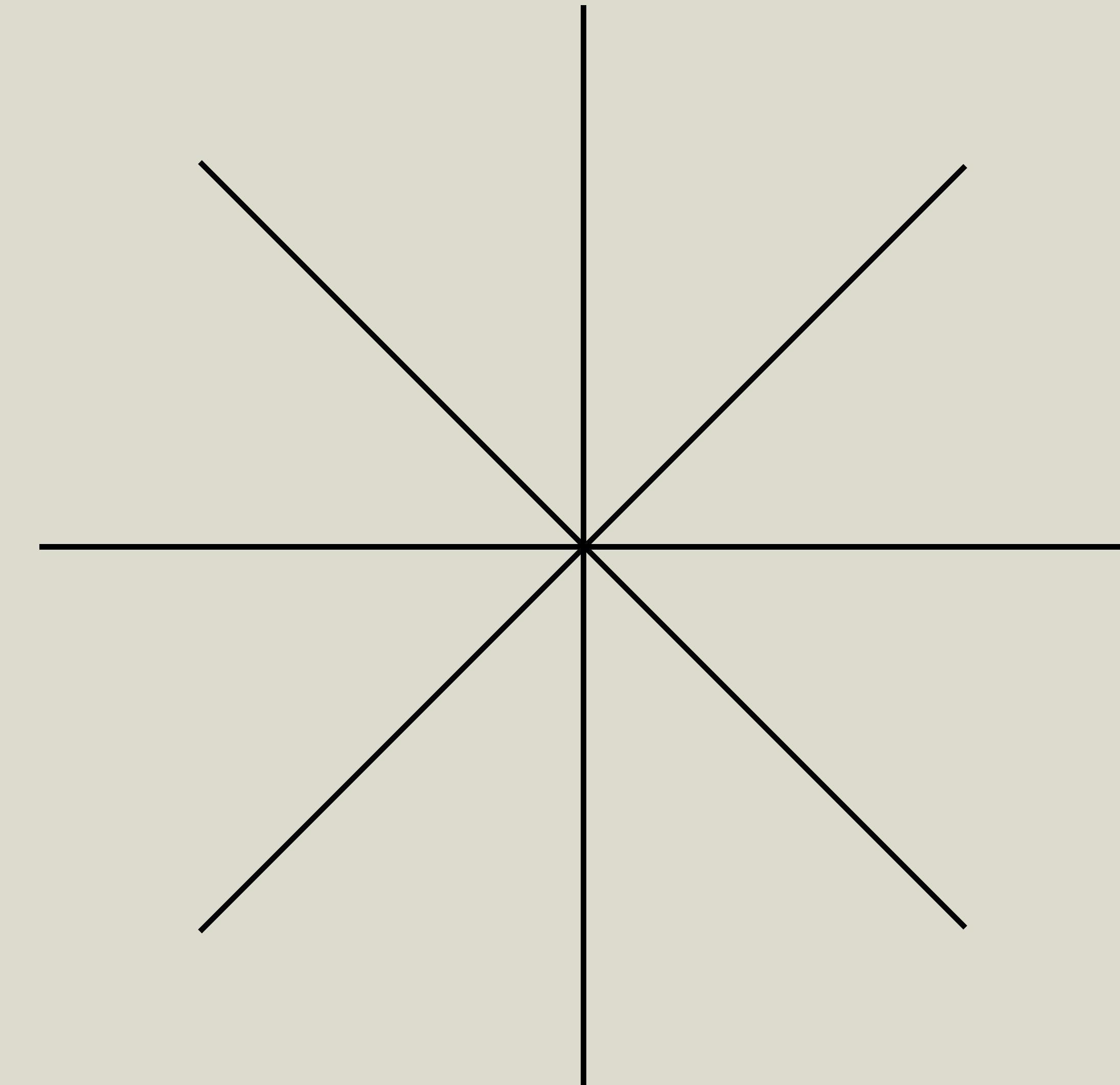
AND



\parallel

OR

08 Comandos If e For



COMANDO IF

O ‘if’ é uma condição lógica cujo seu código só é executado caso satisfaça as condições necessárias.

Neste caso, se a variavel ‘myFavNum’ for maior que 5.

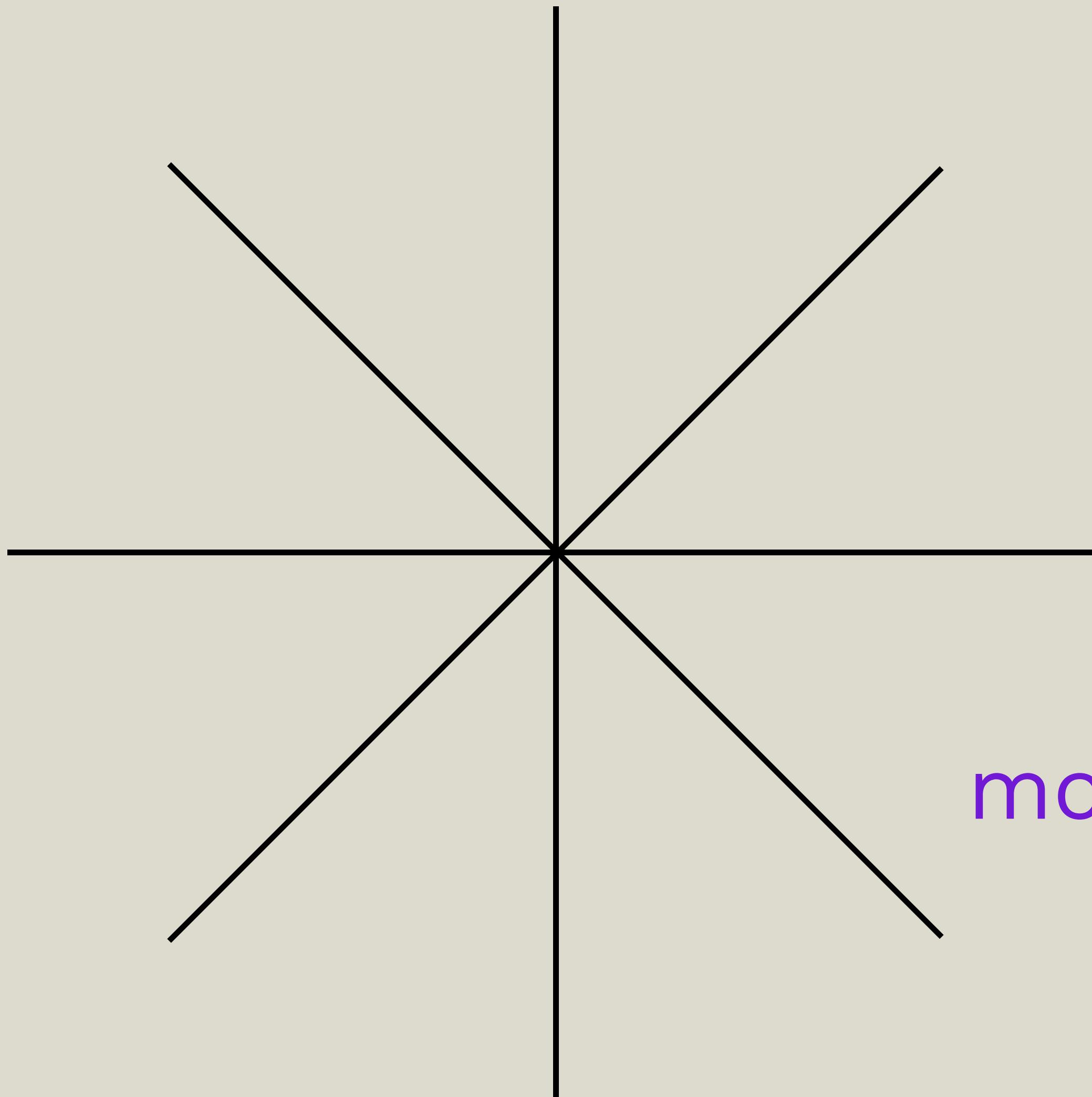
```
if(myFavNum > 5)
{
    print("maior que 5!");
}
```

COMANDO FOR

Uma estrutura de repetição que irá executar um código “em loop” até a condição já não se aplicar.

Neste caso, desde que o i seja menor que 5.

```
for (int i = 0; i < 5; i++)  
{  
    print("Hello!");  
}
```



Obrigada!

Não te esqueças onde
encontrar este ppt:

motamdaniela.github.io/tajd