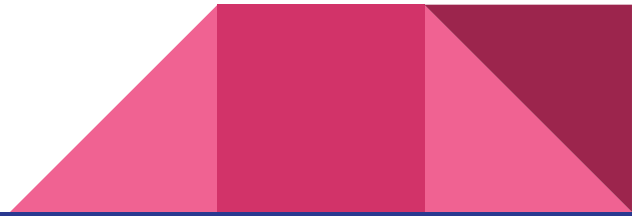
Scripts

Criando seus próprios componentes

Topicos

- A anatomia de um script na unity
  - Using, Class MonoBehaviour, Loops (Start e Update)
- Variáveis e propriedades
- Debug
- Componente detector de colisao
- Scripts Create e destroy



Criando um script

Em Projetos, entrar na pasta Script (se não houver, criar uma)

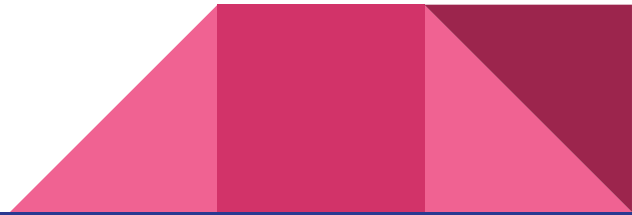
RMB, Create > C# Script

Dê o nome de **SomeScript** para este asset

(O nome do asset é importante, pois ele será utilizado internamente)

Clicar duas vezes no asset criado

(O programa MonoDevelop irá abrir)

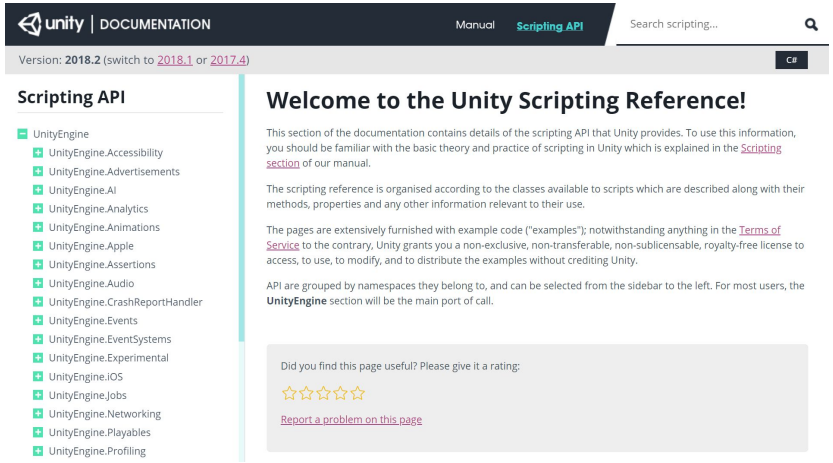


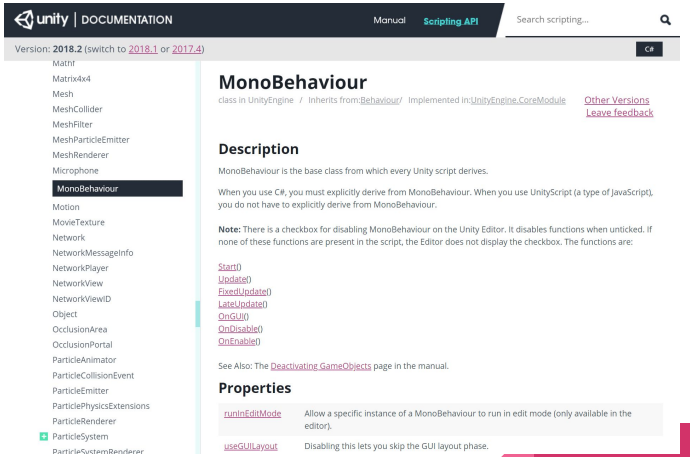
Anatomia

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class SomeScript : MonoBehaviour {
6
7     // Use this for initialization
8     void Start () {
9
10    }
11
12    // Update is called once per frame
13    void Update () {
14
15    }
16 }
17
```

Documentação

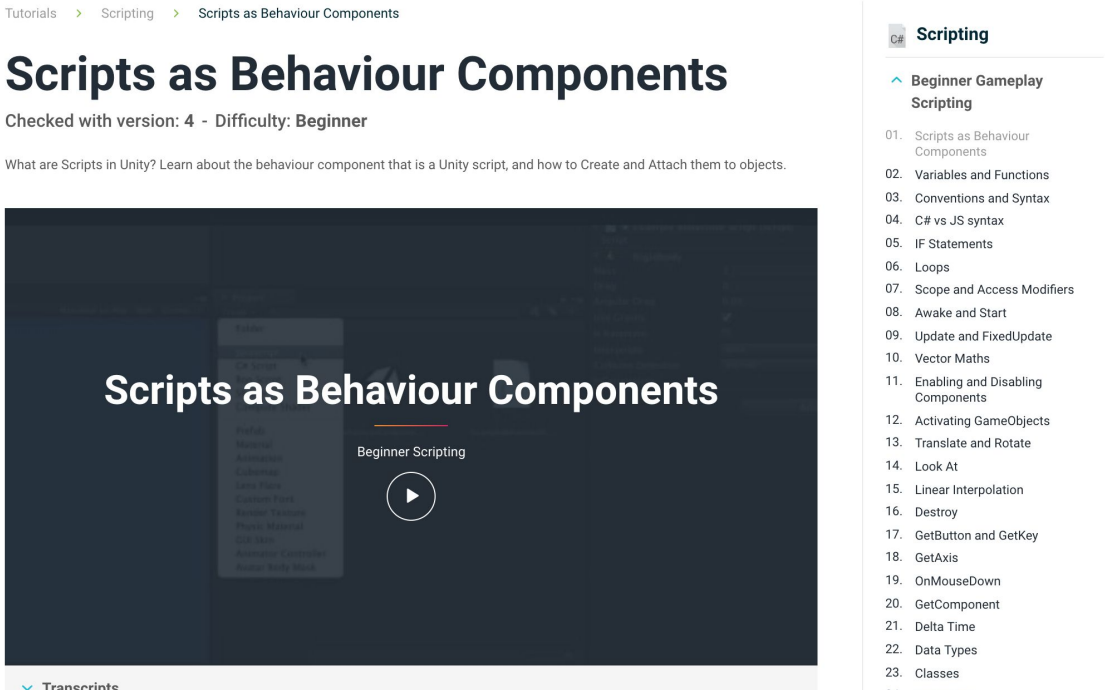
docs.unity3d.com/ScriptReference





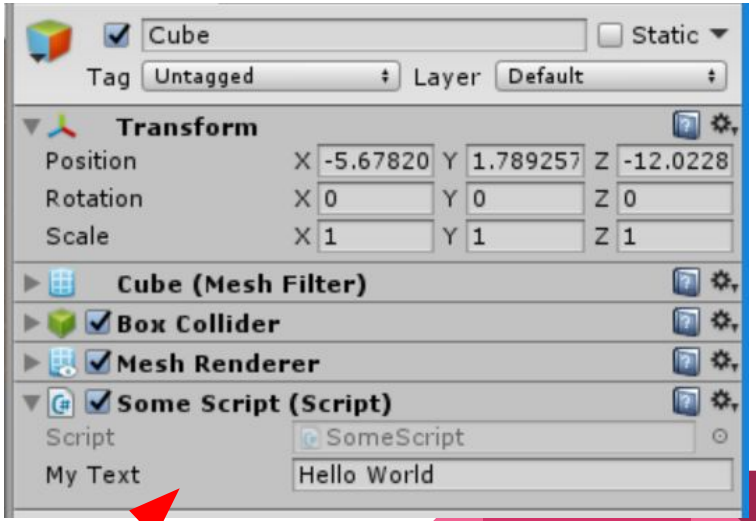
Video Aulas

unity3d.com/learn/tutorials/s/scripting



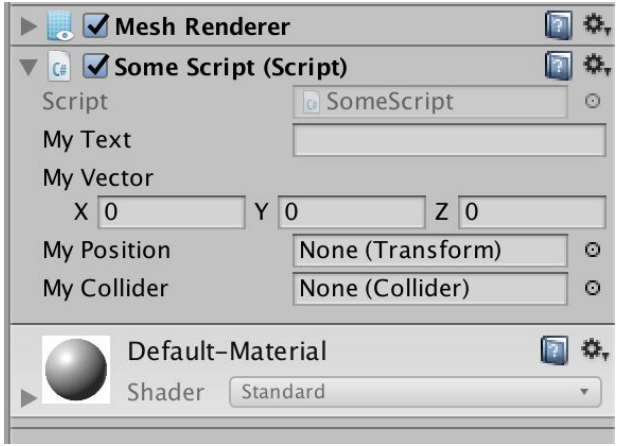
Variáveis e Propriedades

```
5  public class SomeScript : MonoBehaviour {
6
7      public string MyText = "Hello World";
8
9      // Use this for initialization
10     void Start () {
11
12     }
13
14     // Update is called once per frame
15     void Update () {
16
17     }
18 }
19
```

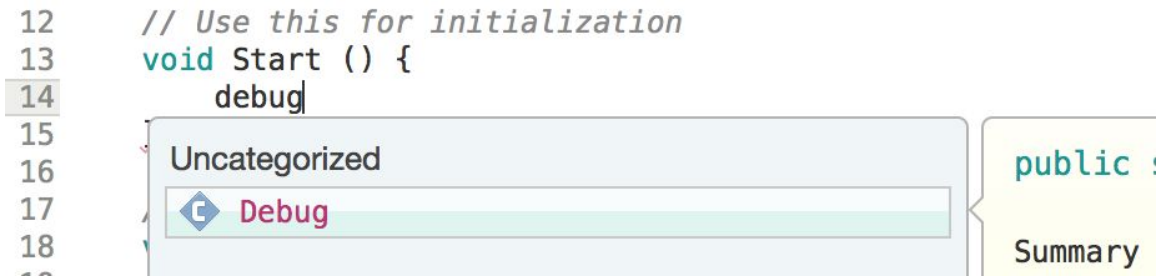


Tipos de Variavel

```
5  public class SomeScript : MonoBehaviour {
6
7      public string myText;
8      public Vector3 myVector;
9      public Transform myPosition;
10     public Collider myCollider;
11
12     // Use this for initialization
13     void Start () {
14
```

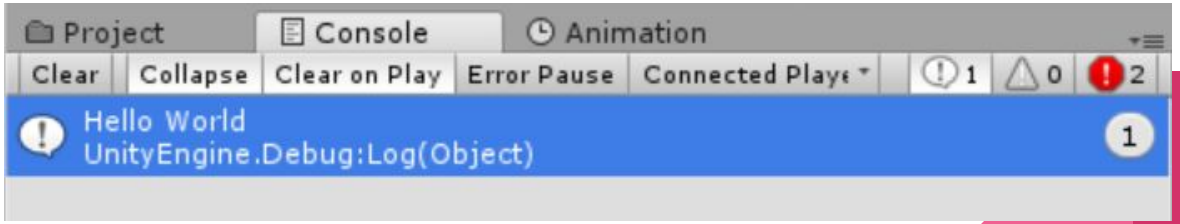


Context Help, Classes e Métodos



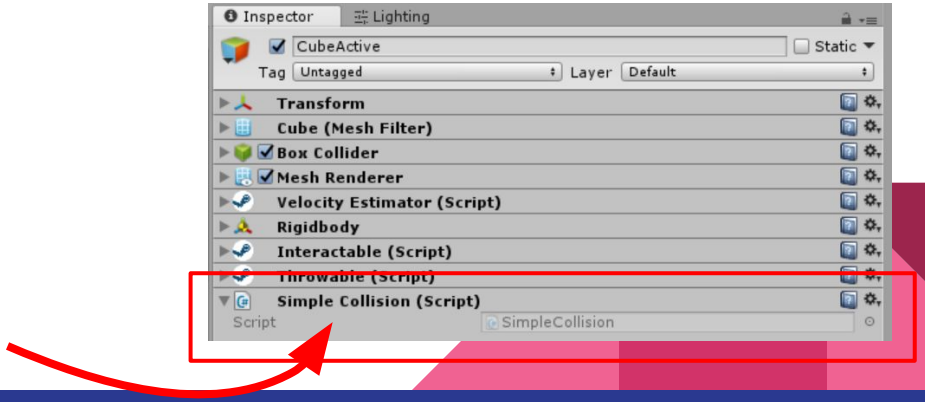
Console

```
4
5 public class SomeScript : MonoBehaviour {
6
7     public string MyText = "Hello World";
8
9     // Use this for initialization
10    void Start () {
11        Debug.Log (MyText);
12    }
```



Criando um Script e aplicando a um objeto

- na janela de Projetos, criar pasta “Scripts”
- Botão direito do mouse > Create > C# Script
- Nomear como “ SimpleCollision ”
- Adicione (arraste) o novo componente para o objeto CubeActive



Editando o script

- Clicar duas vezes sobre o script na janela Projects
- (O editor de scripts será aberto)
- Entre as chaves da classe SimpleCollision, adicionar as linhas como abaixo:

```
5  public class SimpleCollision : MonoBehaviour {
6
7      void OnCollisionEnter (){
8          Debug.Log ("Colidiu");
9      }
10 }
```

- Testar - toque os obstáculos com o CubeActive e observe a mensagem no console

Incluindo dependentes

```
6 [RequireComponent( typeof( Rigidbody ) )]
7 public class SimpleCollision : MonoBehaviour {
8
9     void OnCollisionEnter(){
10        Debug.Log ("Colidiu");
11    }
```

Informações da colisão

```
5 public class SimpleCollision : MonoBehaviour {
6
7     void OnCollisionEnter (Collision info){
8
9         if (info.
10    }
11 }
12
```

collider
contacts
Equals
gameObject
GetEnumerator
GetHashCode
GetType

```
public GameObject gameObject { get; }

Summary
The GameObject whose collider we
are colliding with. (Read Only).
```

Filtrando colisões por nome

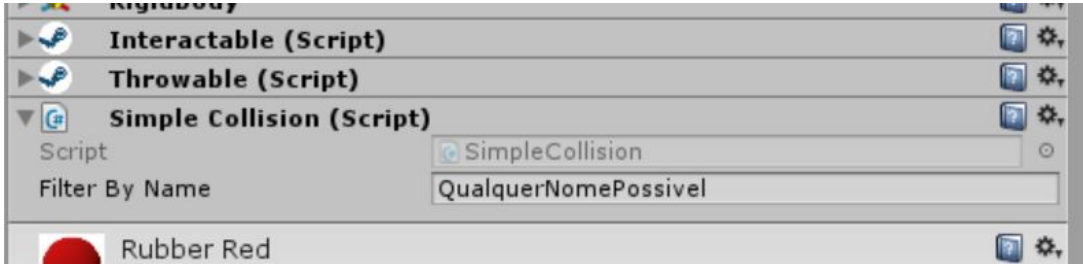
- Altere o script como mostrado abaixo:

```
5  public class SimpleCollision : MonoBehaviour {
6
7      void OnCollisionEnter (Collision info){
8
9          if (info.gameObject.name == "Obstacle") {
10             Debug.Log ( "Colidiu" );
11         }
12     }
13 }
14
```

- Testar - repita o teste tocando os obstáculos

Scripts genéricos: variáveis

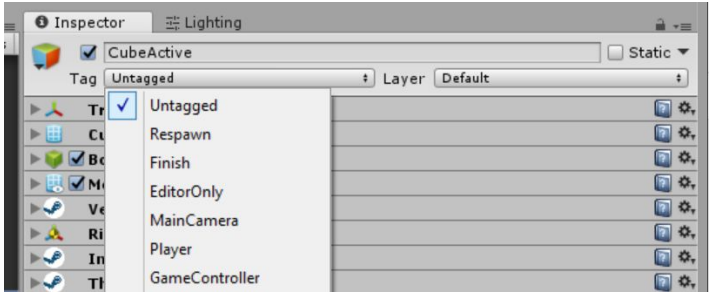
- Adicione a variável “FilterByName” no script como abaixo
- Veja a mudança no componente Simple Collision




```
5 public class SimpleCollision : MonoBehaviour {
6
7     public string FilterByName;
8
9     void OnCollisionEnter (Collision info){
10
11         if (info.gameObject.name == FilterByName) {
12             Debug.Log ("Colidiu");
13         }
14     }
15 }
16
```

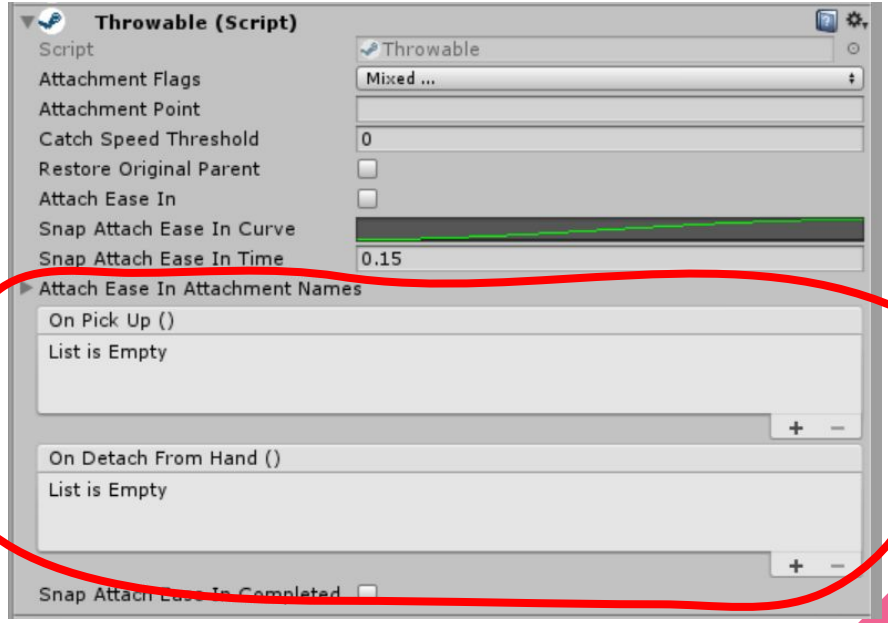
+Filtros: Tags

- Agora cria a variável “FilterByTag” como abaixo

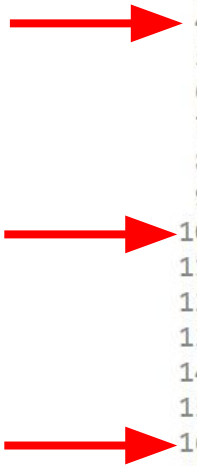



```
5 public class SimpleCollision : MonoBehaviour {
6
7     public string FilterByName;
8     public string FilterByTag;
9
10    void OnCollisionEnter (Collision info){
11
12        if (info.gameObject.name == FilterByName || info.gameObject.tag == FilterByTag) {
13            Debug.Log ("Colidiu");
14        }
15    }
16 }
```

Componentes modulares?



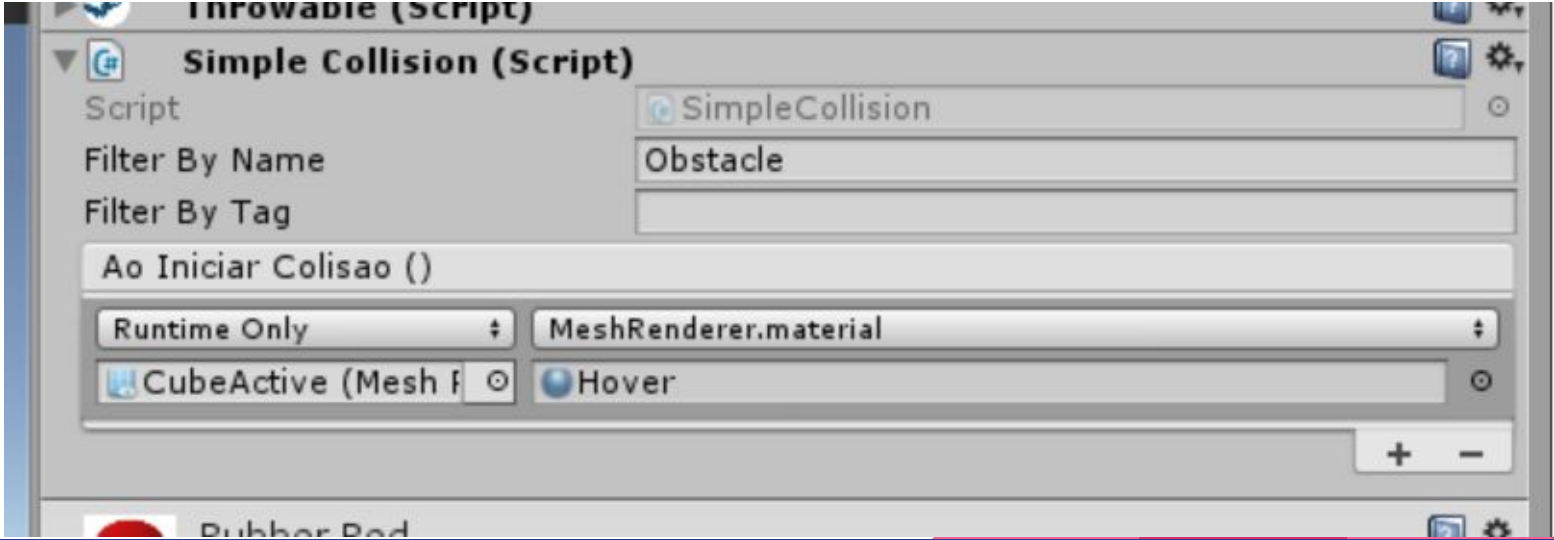
Unity Events



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.Events;
5
6 public class SimpleCollision : MonoBehaviour {
7
8     public string FilterByName;
9     public string FilterByTag;
10    public UnityEvent aoIniciarColisao;
11
12    void OnCollisionEnter (Collision info){
13
14        if (info.gameObject.name == FilterByName || info.gameObject.tag == FilterByTag) {
15            Debug.Log ("Colidiu");
16            aoIniciarColisao.Invoke();
17        }
18    }
19 }
```

Testando

- Arrastar o objeto “CubeActive” da Hierarchy para o campo Object
- Selecionar propriedade MeshRenderer Material material
- Escolher/arrastar um material, como o material “Hover” das aulas passadas
- Testar



OnCollisionExit...

- Declarar variavel pública tipo UnityEvent “aoTerminarColisao”
- Copiar método OnCollisionEnter > trocar nome para OnCollisionExit e trocar evento para aoTerminarColisao

```
6  public class SimpleCollision : MonoBehaviour {
7
8      public string FilterByName;
9      public string FilterByTag;
10     public UnityEvent aoIniciarColisao;
11     public UnityEvent aoTerminarColisao;
12
13     void OnCollisionEnter (Collision info){
14
15         if (info.gameObject.name == FilterByName || info.gameObject.tag == FilterByTag) {
16             aoIniciarColisao.Invoke();
17         }
18     }
19
20     void OnCollisionExit (Collision info){
21         if (info.gameObject.name == FilterByName || info.gameObject.tag == FilterByTag) {
22             aoTerminarColisao.Invoke();
23         }
24     }
25 }
26
```

OnCollisionExit...

No componente Simple Collision, método Ao Terminar Colisao ():

- Arrastar o objeto “CubeActive” da Hierarchy para o campo Object
- Selecionar uma função MeshRenderer Material material
- Escolher/arrastar o material inicial “Default-Material”
- Testar

Script: Create Instance

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CreateInstance : MonoBehaviour
6 {
7     public void Create(GameObject prefab)
8     {
9         GameObject instance = Instantiate (prefab, this.transform.position, this.transform.rotation);
10    }
11 }
12
```

Script: Create Instance

- Na pasta Scripts crie o script “CreateInstance” e abra-o para edição
- Crie o método “Create” como na figura abaixo
  - Recebe um prefab
  - Uma variável temporária (tipo GameObject) armazena uma instância do prefab, gerada pelo
  - Método “Instantiate”, que o coloca na posição do objeto que contém este script

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CreateInstance : MonoBehaviour
6 {
7     public void Create(GameObject prefab)
8     {
9         GameObject instance = Instantiate (prefab, this.transform.position, this.transform.rotation);
10    }
11 }
12
```

Script: Create Instance

- Arraste o script para o objeto CubeActive
- No componente Simple Collision, altere o método “Ao Iniciar Colisao”: na função chamada, selecione agora CreateInstance - Create ()
- No campo de Objeto arraste o “ObstacleSphere” (da Hierarquia)
- Teste - arraste e solte o ActiveCube em cima do Obstacle

Script: Self Destroy

```
4
5 public class SelfDestroy : MonoBehaviour {
6
7     public void Destroy(){
8         Destroy(gameObject.transform.root.gameObject);
9     }
10 }
11
```

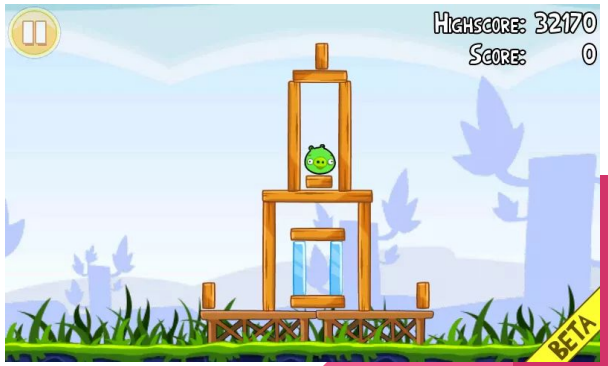
Script: Self Destroy

- Na pasta Scripts crie o script “SelfDestroy” e abra-o para edição
- Crie o método “Destroy ()” como na figura abaixo
- Arraste-o para o objeto CubeActive
- No componente Simple Collision, altere o método “Ao Terminar Colisao”: na função chamada, selecione agora Self Destroy - Destroy()
- Testar

```
4
5 public class SelfDestroy : MonoBehaviour {
6
7     public void Destroy(){
8         Destroy(gameObject.transform.root.gameObject);
9     }
10 }
11
```

Exercicios

- Easy: Dragonball Capsules
  - Quando uma capsula encosta no chão, se transforma em outro objeto!
- Tricky: AngryBirds Stack
  - Um porco se protege no alto de um castelo de blocos
  - Quando o porco é derrubado no chão, a fase recomeça



Lab: Projeto

- Consolidar as equipes
- Para semana que vem (entrega pelo representante no e-disciplinas):
  - definição do escopo - objetivo, principais features, descrever a experiência esperada
  - divisão de tarefas - listar representante / papel principal de cada membro
  - previsão de logística - reuniões e trabalho fora de aula; computadores pessoais

Exercícios p/ casa - próximo abre dia 05

- Óculos anaglifo - devolver
  - Bruno Akio Shirasuna
  - Bruno da Costa Braga
  - Bruno Mucha Pasini
  - Guilherme de Agrela Lopes
  - Lucas Giannella de Oliveira
  - Raul da Silva Souza
  - Gabriel de Souza Oliva
  - Gustavo Kimura
  - **Quem ainda não devolveu**
- permanecer
  - Igor Freitas Sym
  - Rodrigo Zanette de Magalhães
- Pegar - devolver próxima aula
  - Adriano Carvalho e Sousa
  - Augusto Ruy Machado
  - Felipe Caracciolo Goncalves
  - Gustavo Henrique Alves Gregorio
  - Juliana de Abreu Faria
  - Lincoln Makoto Kawakami
  - Lucas Almeida Santos
  - Lucas Gabriel de Castro Negrini
  - Lucas Grob Sponchiado
  - Lucas Paiva da Costa
  - Marcos Roberto Franco Filho
  - Rodrigo M. Magaldi
  - Thiago Perroni Meletti
  - Vinicius H. de Freitas
  - Vitor Martes Sternlicht

Proxima aula

Mais scripts!

A lendária arma de um jogo clássico

Lab: Projeto

- Combinar os componentes estudados para criar interações mais complexas
- Não esquecer dos eventos em componentes do SteamVR
  - Throwable : OnPickUp, OnDetachFromHand
  - Hover : OnHandHoverBegin/End, OnAttachedTo/DetachedFromHand
- Rotina de testes!