

# Desenvolvimento com Motores de Jogos II

## Aula 04 - Jogo Polygonal Rescue - Parte 2 - Objetos Coletáveis

# Apresentação

---

Olá! Na aula passada, desenvolvemos a primeira parte do jogo *Polygonal Rescue*, nele criamos o ambiente base do jogo (Labirinto), o nosso personagem com movimento baseado em física usando o RigidBody, usamos também simples materiais para modificar a cor de elementos da cena e aprendemos a criar um sistema de movimento com velocidade configurável no editor.

Estamos bem avançados, não acham? Sinal que o trabalho valeu a pena. E atualmente o nosso jogo já tem o piso do labirinto, suas paredes e o personagem com movimento.

Nesta aula, continuaremos a desenvolver o jogo, então vamos criar os polígonos que serão resgatados em cada labirinto do jogo, assim como aprenderemos a criar Tags as quais marcarão qual tipo de objeto pode ou não ser coletado, além disso, usaremos colisores especiais que somente geram eventos de colisão, mas sem interação física (Triggers).

## Objetivos

Ao final desta aula, você deverá ser capaz de:

- Desenvolver o Jogo Polygonal Rescue - Parte 2;
- Criar e desenvolver Objetos Coletáveis e Tags;
- Usar Trigger Colliders.

## Criando Objetos Coletáveis

---

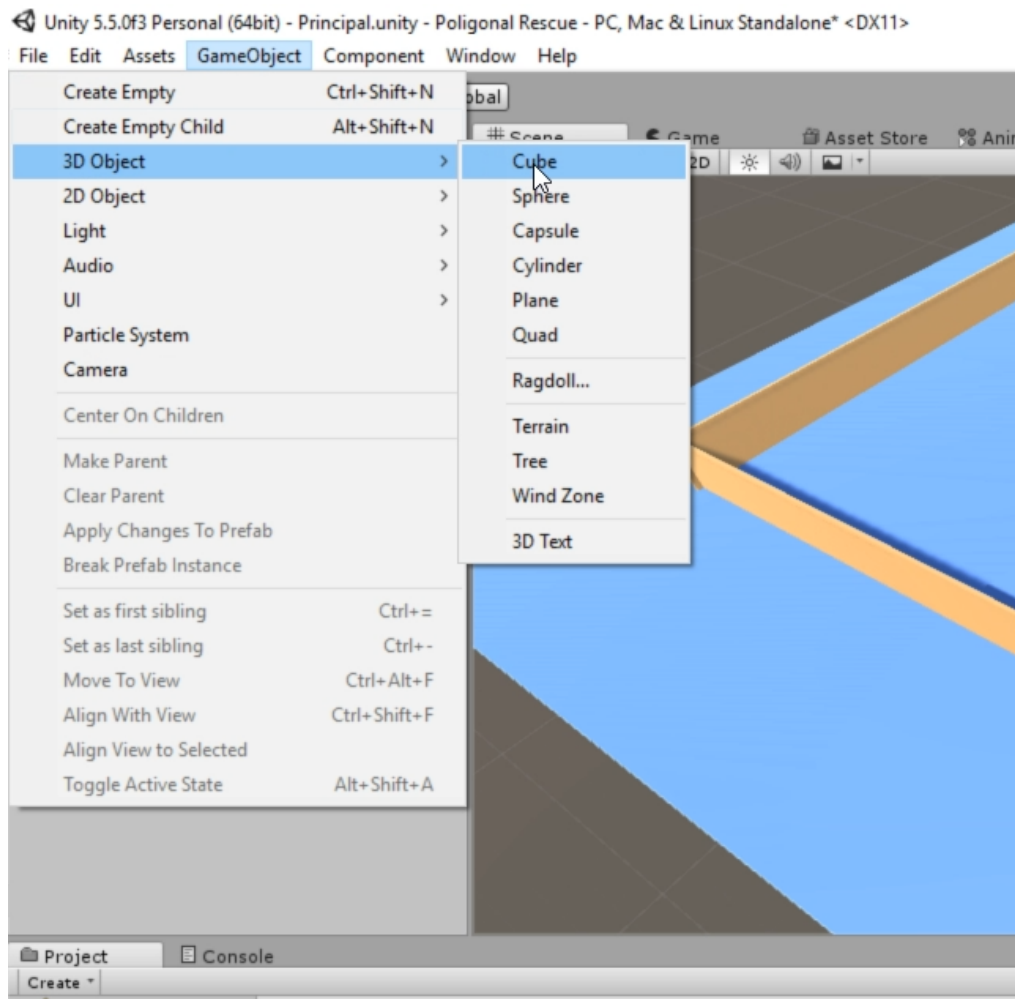
Os objetos coletáveis do jogo *Polygonal Rescue* são polígonos perdidos no labirinto que serão salvos pelo nosso personagem.



Esses polígonos serão simples GameObjects (Cubos e Cilindros) que, ao contato do personagem, são removidos da cena, adicionando um ponto ao jogador por resgate. O objetivo do jogador é resgatar o máximo de polígonos antes de finalizar o labirinto. Nas próximas aulas adicionaremos um tempo limite para que o jogador finalize cada fase do jogo, adicionando um maior grau de dificuldade ao mesmo.

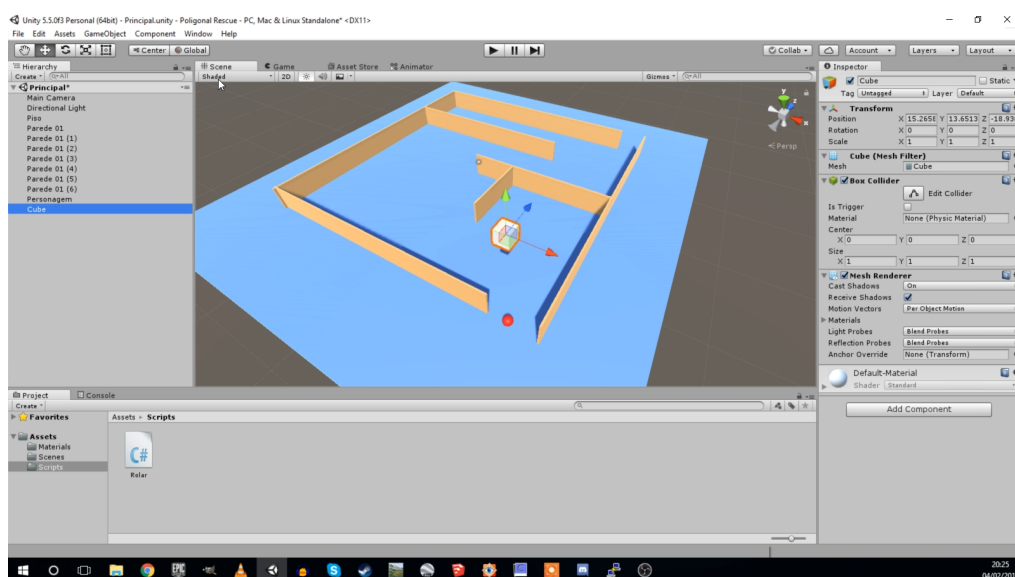
Inicialmente, criaremos o primeiro objeto resgatável (um simples cubo) escolhendo a opção no menu GameObject -> 3D Object -> Cube. Veja nas **Figura 1 e 2**.

**Figura 01** - Criando o primeiro objeto coletável.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

**Figura 02** - Cubo criado como objeto coletável.

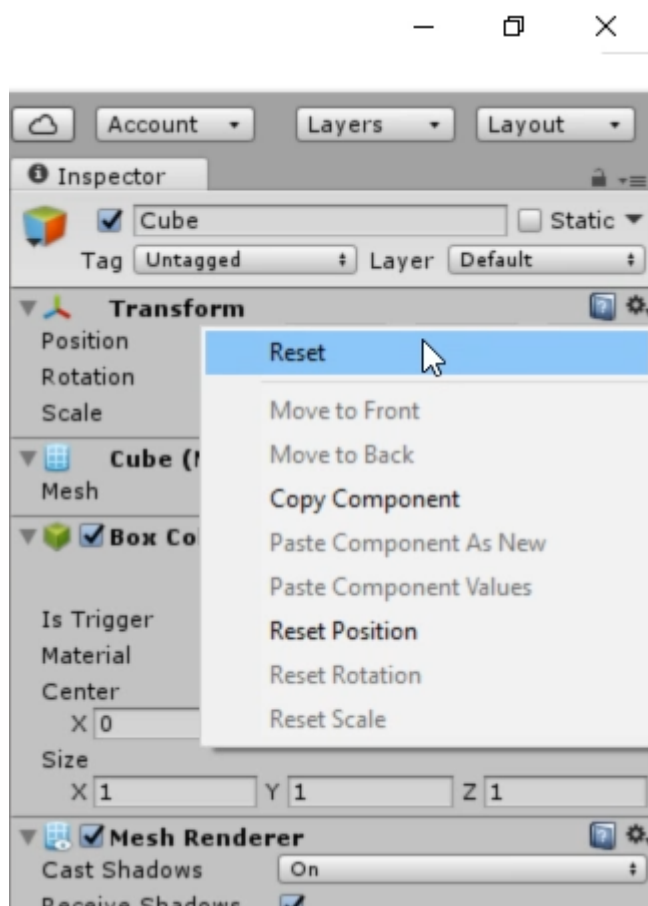


**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

---

O Cubo é criado em uma posição baseada no centro da câmera da cena atual, então é sempre uma boa prática aplicar o **Reset** da sua posição. Antes de posicioná-lo onde deseja, observe a **Figura 3**.

**Figura 03** - Reset do transform (posição, rotação e escala) do primeiro objeto coletável.



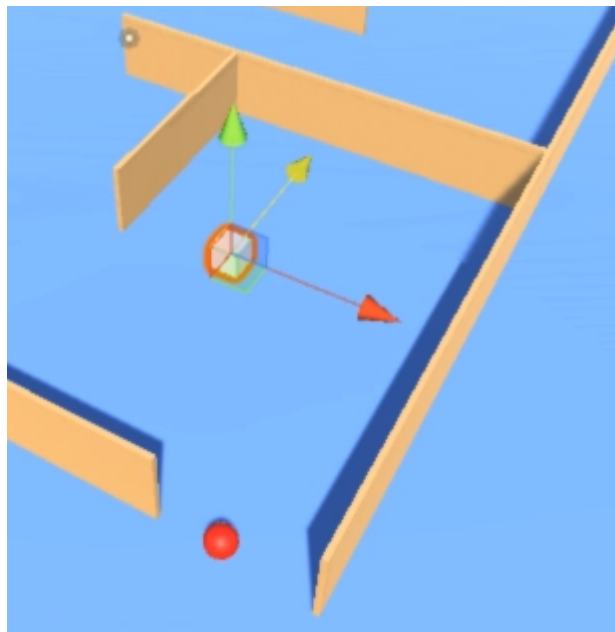
**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Após aplicar um Reset no Transform do cubo, posicione-o próximo ao personagem para realizar os primeiros testes.

## Atenção!

Não esqueça de “levantar” o cubo (eixo Y) o suficiente para que ele fique imediatamente acima do piso da cena, flutuando de forma bem baixa e tomando cuidado para que o personagem consiga entrar em contato com o cubo quando se aproximar. Se você levantar muito esse cubo no eixo Y, o personagem passará por baixo dele e não conseguirá coletá-lo. A Figura 4 mostra o cubo posicionado na cena.

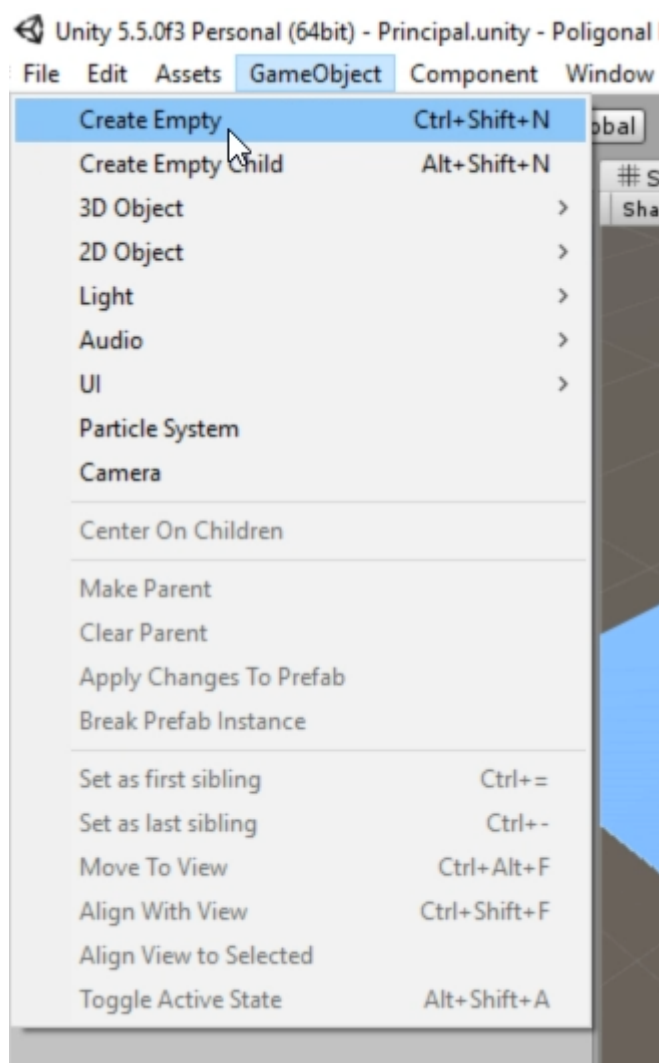
**Figura 04** - Cubo posicionado na cena próximo ao personagem.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Uma boa prática a seguir e que lhe ajudará a manter os objetos coletáveis organizados na nossa cena é criar um **GameObject vazio** o qual agrupará todos os coletáveis como filho dele, parecido com uma pasta. Para isso vá em GameObject -> Create Empty, veja na **Figura 5**. Depois renomeie esse GameObject para “Coletáveis” e reset o Transform dele para que ele fique no centro da cena (posição 0,0,0).

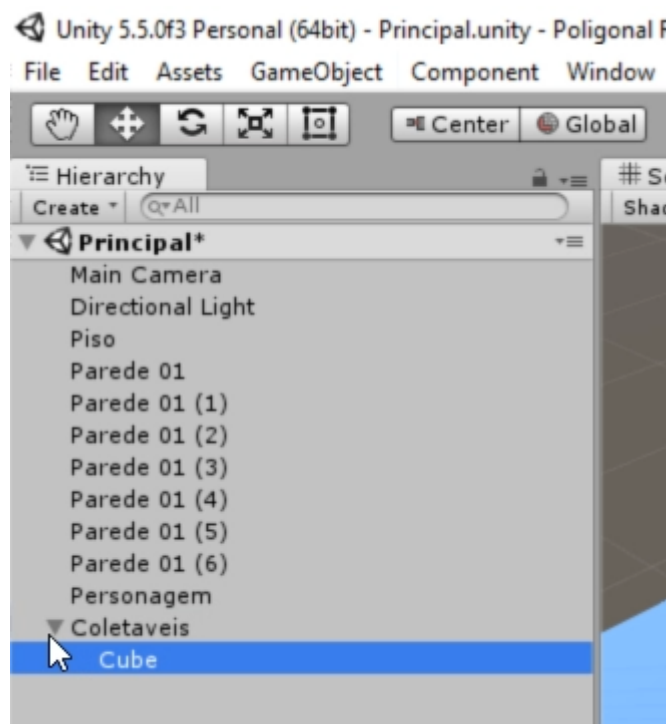
**Figura 05** - Criação de um GameObject vazio para agrupar os coletáveis.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Em seguida, adicione o Cubo criado anteriormente (nosso polígono a ser coletado) como filho de Coletáveis, arrastando com o mouse o Cubo para cima de Coletáveis, veja na **Figura 6**.

**Figura 06** - Cubo coletável adicionado como filho do GameObject Coletáveis.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

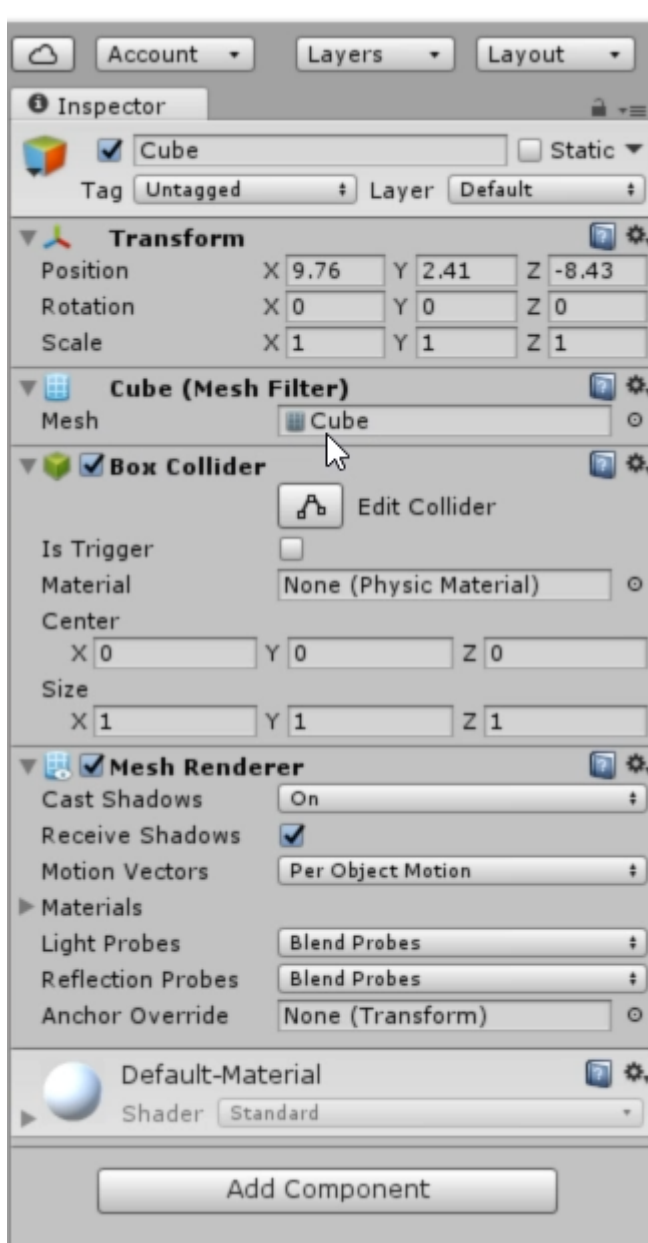
## Atenção!

Algo importante a se observar é que como os Coletáveis apresentam uma posição (Transform position) e o cubo é filho dele, se eu movimentar Coletáveis todos os objetos filhos também se movem, porém o valor Position dos objetos filho permanece o mesmo, pois esse valor é a sua posição relativa ao pai (Coletáveis) e não à cena.

O Cubo que será o nosso primeiro coletável possui alguns componentes já adicionados. Além do Transform, ele tem um **Mesh Filter**, que basicamente informa qual elemento 3D (Mesh) esse objeto carrega (nesse caso um cubo). Além disso, tem um **Box Collider**, que é um colisor no formato de um cubo e serve para informar ao Unity qual tipo de geometria ele usará para calcular as colisões com esse objeto e também tem um **Mesh Renderer**, com a função de exibir (renderizar) o Mesh 3D no **Mesh Filter** na tela usando o material aplicado nele. Veja na **Figura 7**.



**Figura 07** - Componentes do Cubo coletável.



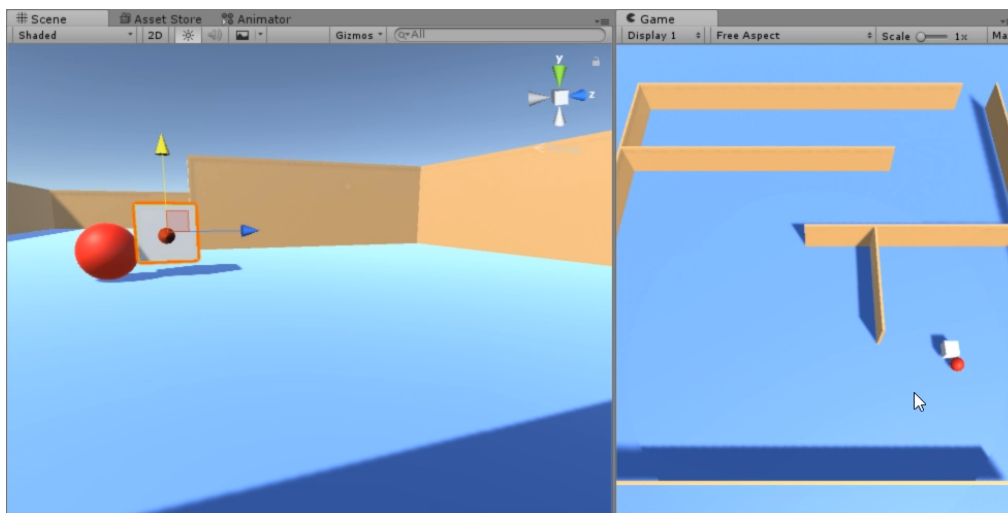
**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Nesse momento daremos uma maior atenção ao **Box Collider**, pois ele será responsável por disparar o evento de colisão do personagem com esse Cube.

Para isso, vamos iniciar o jogo e ver o que acontece com o personagem quando ele entra em contato com o Cubo. Tente movimentar o jogador de maneira que ele colida com o cubo e perceba o personagem interagindo fisicamente com o cubo, mudando de direção e velocidade. O Cubo, entretanto, fica paralisado, pois ele não

tem um Rigidbody associado (e nem deverá ter). Veja a animação que mostra o momento da colisão entre o personagem (esfera vermelha) e o objeto coletável (Cubo).

**Figura 08** - Momento da colisão entre o personagem (esfera vermelha) e o objeto coletável (Cubo) com a janela Game e Scene lado a lado.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017



**Vídeo 01** - Colisão

## Colisão com o Objeto Coletável

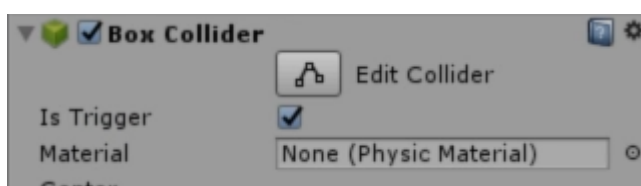
Nosso objetivo para criar um objeto coletável é simples: o personagem deve detectar a colisão e nesse instante deverá remover o cubo da cena, porém sem interagir fisicamente com ele. Para isso funcionar, primeiramente deveremos modificar o **Box Collider** do Cubo marcando a opção **"Is Trigger"**, como na **Figura 9**. Essa opção faz com que o Box Collider continue detectando as colisões, mas sem agir fisicamente sobre o personagem ou qualquer outro objeto que ele colidir, ou seja, o personagem vai passar pelo cubo como se ele não estivesse ali. Esse é o

comportamento que queremos, pois podemos remover o cubo no instante da colisão e o personagem não sofrerá mudança de velocidade e direção no seu movimento.

## Atenção!

Cuidado para não marcar como “Is Trigger” o collider do personagem e sim o do Cubo coletável.

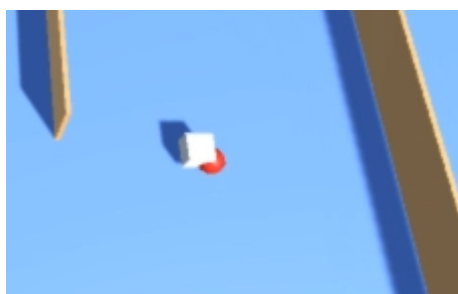
**Figura 09** - Box Collider do cubo coletável marcado como “Is Trigger”.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Inicie o jogo novamente e tente colidir com o Cubo. Você verá que agora o personagem passa por dentro do cubo. Este instante pode ser visto na **Figura 10**.

**Figura 10** - Personagem passando por dentro do cubo com o Box Collider marcado como “Is Trigger”.



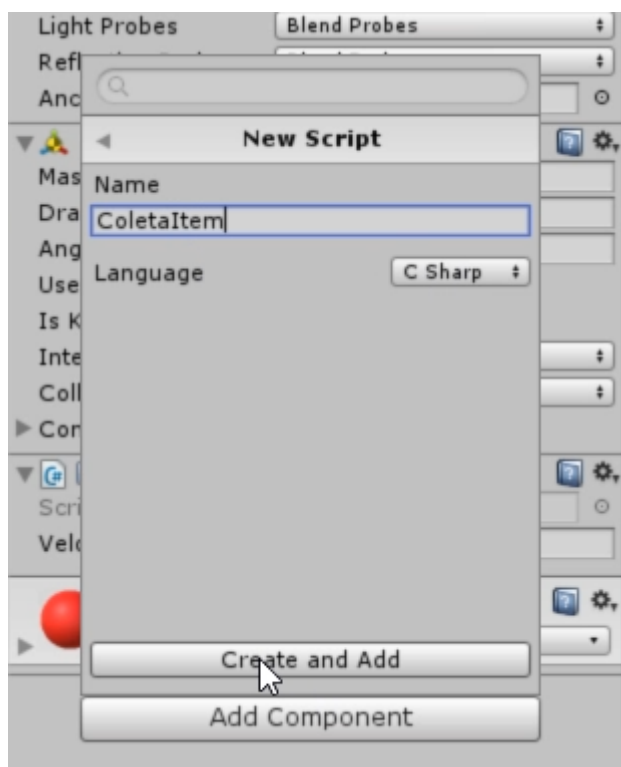
**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017



**Vídeo 02** - Is Trigger

Para verificar se ocorreu a colisão entre o personagem e o cubo, é necessário escrever um Script. Selecione o seu personagem, vá até a janela Inspector, clique em Add Component, escolha "New Script" e nomeie o script para "ColetaItem", em seguida clique no botão Create and Add (Veja na **Figura 11**). Isso criará o script ColetaItem.cs e já o associará ao seu personagem. O script será criado na pasta Assets e você poderá movê-lo livremente para a pasta Scripts para manter o seu projeto organizado.

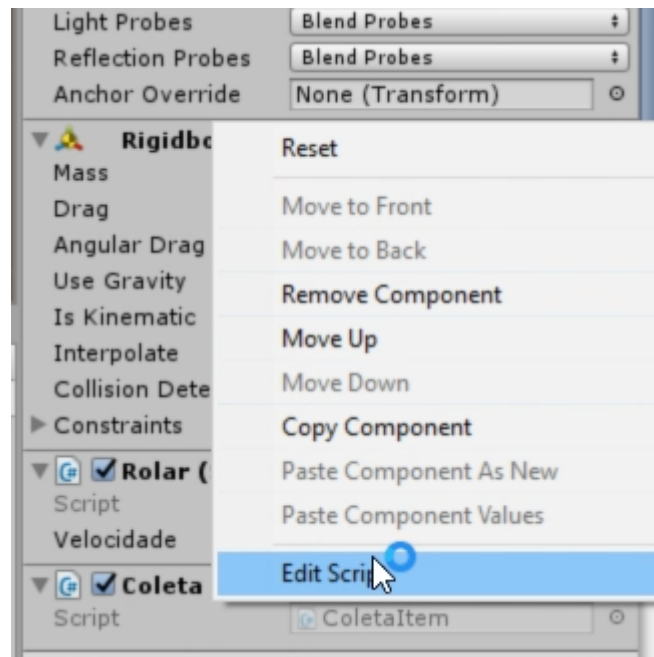
**Figura 11** - Criação do script ColetaItem.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Agora vamos editar o script para adicionar o método que será executado quando uma colisão acontecer. Escolha a opção de editar o script ColetaItem, como na **Figura 12**.

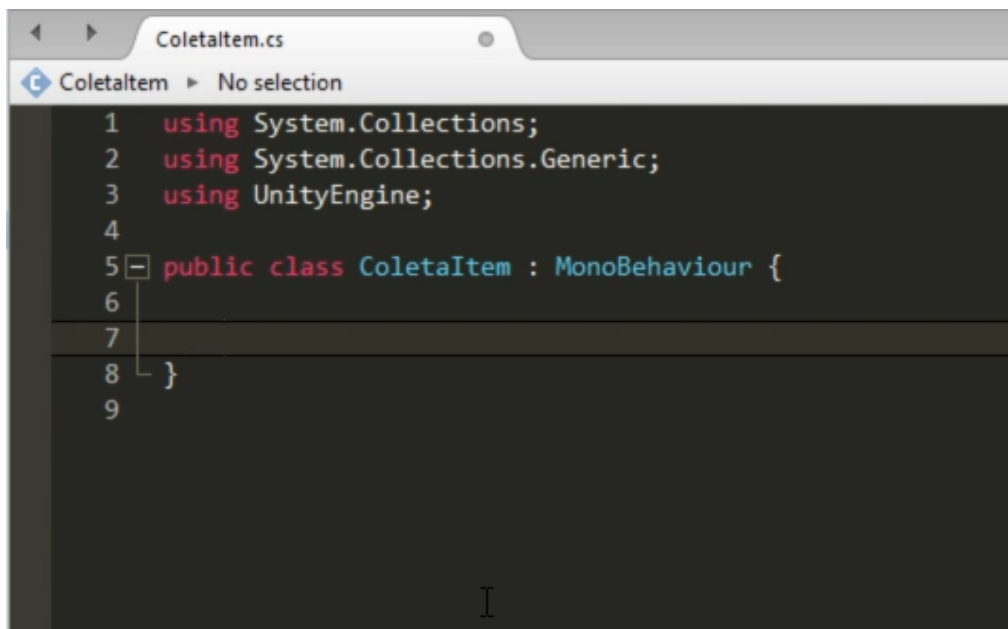
**Figura 12** - Opção de editar script ColetaItem do personagem.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

O MonoDevelop será aberto com o script ColetaItem.cs. Ele virá com o código fonte padrão, já com os métodos **Start** e **Update** criados. Remova esses dois métodos deixando apenas a classe criada, veja na **Figura 13**.

**Figura 13** - Script ColetaItem.cs somente com a classe criada, sem métodos.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Como previsto, uma colisão acontecerá entre o personagem e o cubo coletável, e como o seu **Box Collider** está marcado como “Is Trigger”, não haverá interação física, porém, um evento de colisão ocorrerá e executará um método especial nesse script ColetaItem, caso o método exista, então poderemos escrever o código que irá remover o cubo coletável da cena. Esse método especial é chamado de “OnTriggerEnter”, pois recebe um colisor como parâmetro e deve ser escrito dentro da classe ColetaItem. Veja na **Figura 14**.

**Figura 14** - Método OnTriggerEnter na classe ColetaItem associada ao personagem.

```
ColetaItem.cs
ColetaItem ▶ OnTriggerEnter (Collider other)
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class ColetaItem : MonoBehaviour {
6      void OnTriggerEnter(Collider other) {
7
8      }
9  }
10
```

**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Repare que o método **OnTriggerEnter** tem o parâmetro “other”, que é do tipo Collider. Como o script está associado ao personagem, esse método será chamado todas as vezes em que o personagem colidir com algo, então o parâmetro “other” recebido terá uma referência ao outro collider participante da colisão. Então, no nosso caso, em uma colisão do personagem com um cubo coletável, o parâmetro “other” será uma referência ao Box Collider do Cubo.

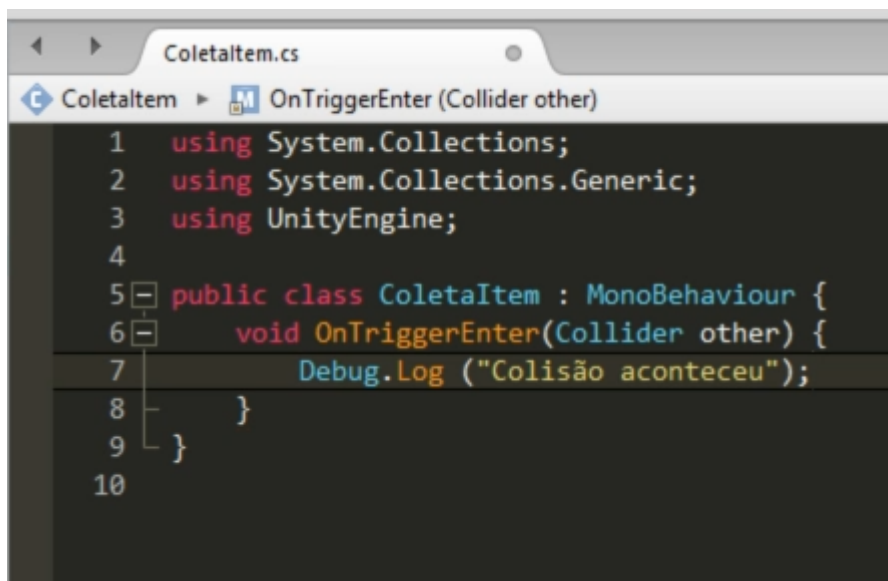
## Atenção!

Lembre-se que o Box Collider do Cubo é um componente do nosso objeto coletável e não o objeto coletável em si. Para acessar o Cubo por completo a partir do seu box colider (na variável other), você pode usar o comando “**other.gameObject**”, o qual trará uma referência ao Cubo coletável.

Antes de remover o cubo da cena, vamos verificar se quando houver uma colisão o método `OnTriggerEnter` estará sendo chamado com sucesso. Para isso usaremos o comando **Debug.Log**, que é uma forma de escrever mensagens na janela Console do editor, muito útil para verificar se o seu jogo está se comportando de maneira adequada durante a sua execução.

Escreva `Debug.Log("Colisão aconteceu");` dentro do método `OnTriggerEnter`, veja na **Figura 15**.

**Figura 15** - Método `OnTriggerEnter` com um `Debug.Log`, verificando se ele está sendo chamado durante as colisões.



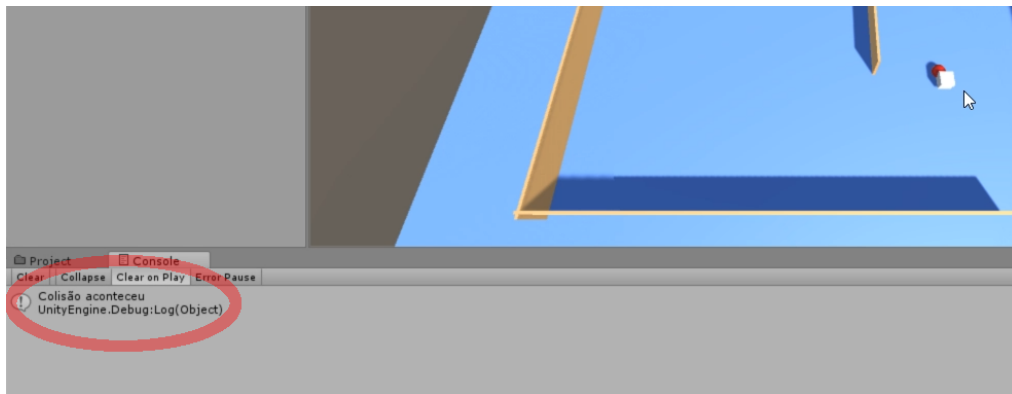
A imagem mostra uma captura de tela do editor de código do Unity. No topo, há uma aba intitulada 'ColetaItem.cs'. Abaixo dela, há uma barra de busca ou filtro que contém 'ColetaItem' e 'OnTriggerEnter (Collider other)'. O código exibido é o seguinte:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class ColetaItem : MonoBehaviour {
6     void OnTriggerEnter(Collider other) {
7         Debug.Log ("Colisão aconteceu");
8     }
9 }
10
```

**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Clique na aba da janela Console para deixá-la aberta e execute o jogo, movimente o personagem até que ele colida com o Cubo e veja se a mensagem **“Colisão aconteceu”** aparece na janela Console, conforme mostra a **Figura 16**.

**Figura 16** - Mensagem de log escrita no Console durante a colisão do personagem com o Cubo coletável.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

A mensagem apareceu? Ótimo, nossas colisões estão sendo detectadas. Repare que se o personagem colidir com uma parede, por exemplo, nada acontecerá, pois somente o colisor do Cubo está marcado como Is Trigger e só estamos detectando esse tipo de colisão por usarmos o método `OnTriggerEnter` para isso. Nas próximas aulas exploraremos outros tipos de colisões e estudaremos esse assunto mais profundamente.

## Removendo o Cubo da Cena

Com a colisão já funcionando entre o personagem e o cubo coletável, vamos removê-lo da cena nesse instante. Para isso você pode simplesmente usar o comando **`Destroy(other.gameObject);`** dentro do método `OnTriggerEnter` no script `ColetaItem`.

Lembrando, o parâmetro “other” é o Box Collider do Cubo e o `other.gameObject` é o Cubo em si, o qual queremos remover. Se você simplesmente executar o comando `Destroy` diretamente em `other`, somente o Box Collider do cubo será removido, e não o cubo por inteiro. Então o método `OnTriggerEnter` ficará assim:

```
1 void OnTriggerEnter(Collider other) {  
2   Destroy(other.gameObject); }
```



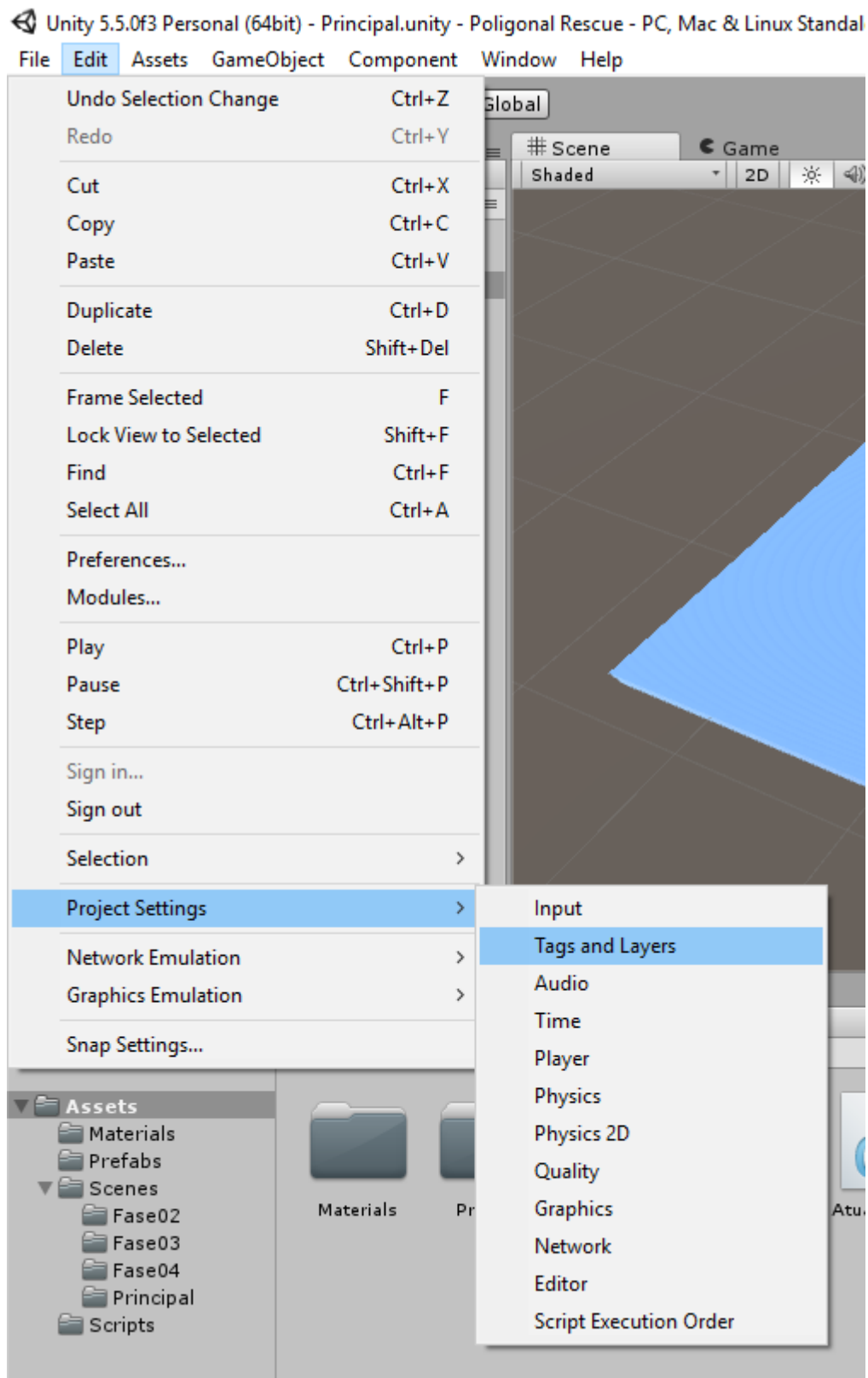
Isso funcionará normalmente no nosso caso, pois, por enquanto, apenas os objetos coletáveis têm colisores com o Is Trigger marcado. Entretanto esse código não segue uma boa prática, já que ele não verifica de forma alguma se estamos colidindo realmente em um objeto, o qual deverá ser coletado (removido com o Destroy). Se no futuro o nosso jogo tiver objetos que tenham colisores com os Is Trigger marcados, mas não se trate de um coletável, o personagem mesmo assim irá removê-lo da cena. Para resolver esse problema vamos marcar os objetos coletáveis com um recurso especial do Unity chamado de **Tags**.

## Tags

---

No Unity podemos marcar qualquer GameObject com um tipo especial de propriedade chamado de Tag. Tags, no Unity, são configuradas por projeto e cada novo projeto do Unity vem com uma série de Tags por padrão, são elas: **Respawn, Finish, EditorOnly, MainCamera, Player e GameController**. Você pode usar essas Tags para marcar seus GameObjects livremente, porém também é possível adicionar novas Tags em um projeto. Para adicionar uma nova Tag em um projeto, escolha a opção no menu Edit -> Project Settings -> Tags and Layers, conforme mostra a **Figura 17**.

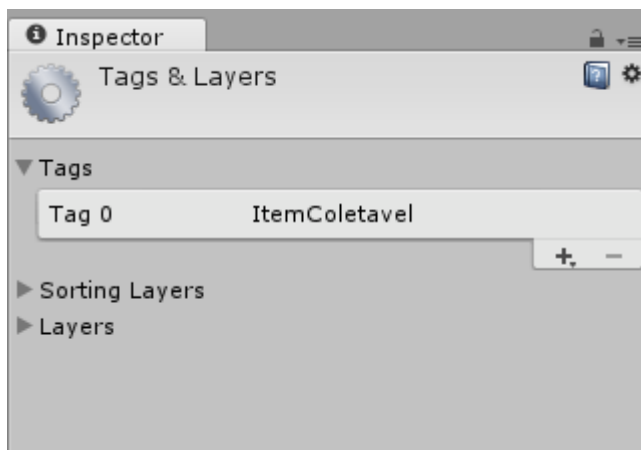
**Figura 17** - Menu Tags and Layers.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Logo após abrirá uma janela no local do Inspector com três listas: **Tags**, **Sorting Layers** e **Layers**. Por enquanto vamos dar atenção à lista de Tags. Clique no ícone “+” e crie uma nova Tag colocando o nome “**ItemColetavel**”, veja na **Figura 18**.

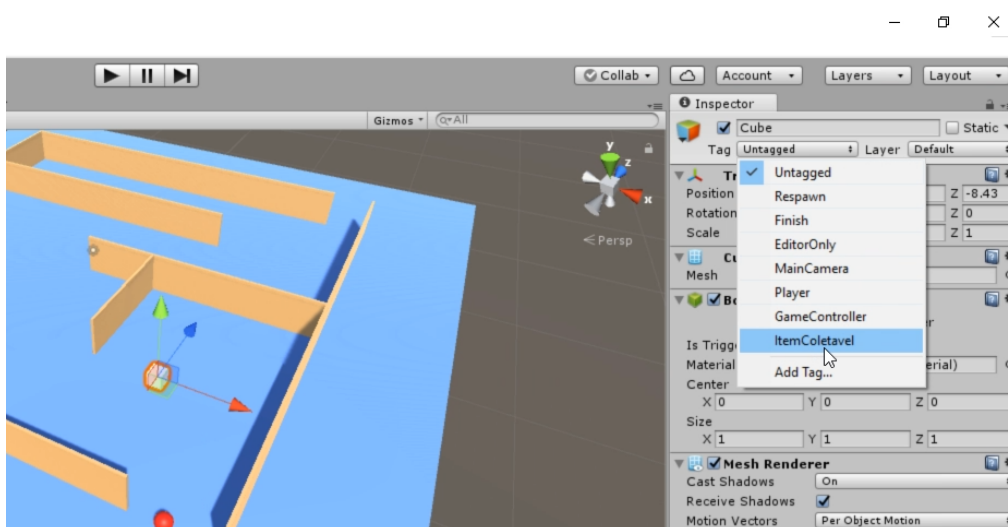
**Figura 18** - Nova Tag “ItemColetavel” adicionada.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Com a nova Tag ItemColetavel criada, podemos agora clicar no nosso Cubo e no Inspector, na opção Tag, escolhemos ItemColetavel (Figura 19). Isso marcará esse Cubo com essa Tag que poderá ser verificada durante a colisão para saber se o objeto deve ou não ser coletado. Essa Tag ItemColetavel será utilizada por todos os itens coletáveis no nosso jogo, não apenas pelo Cubo.

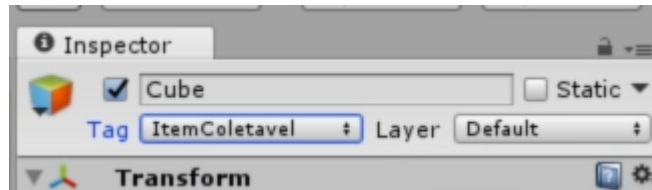
**Figura 19** - Cubo sendo marcado com a Tag ItemColetavel (ver no Inspector).



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Na **Figura 20** você pode ver nas propriedades do Cubo que ele está agora marcado com a Tag ItemColetavel.

**Figura 20** - Cubo já marcado com a Tag ItemColetavel.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Pronto! Agora podemos verificar no momento da colisão se o objeto que o personagem colidiu será realmente coletável caso ele possua essa Tag.

## Removendo o Item Coletável

---

Edite o script ColetaItem associado ao personagem e modifique o código do método OnTriggerEnter para que fique desta forma:

```
1 void OnTriggerEnter(Collider other) {  
2     Debug.Log ("Colisão aconteceu");  
3     if (other.CompareTag("ItemColetavel")) {  
4         Destroy (other.gameObject); } }
```

Veja que usamos o comando `other.CompareTag("ItemColetavel")` para verificar se realmente o objeto com o qual o personagem colidiu é um item marcado com essa Tag.

Assim seu código fica muito mais seguro e você não vai sair coletando coisas que não deveria :).

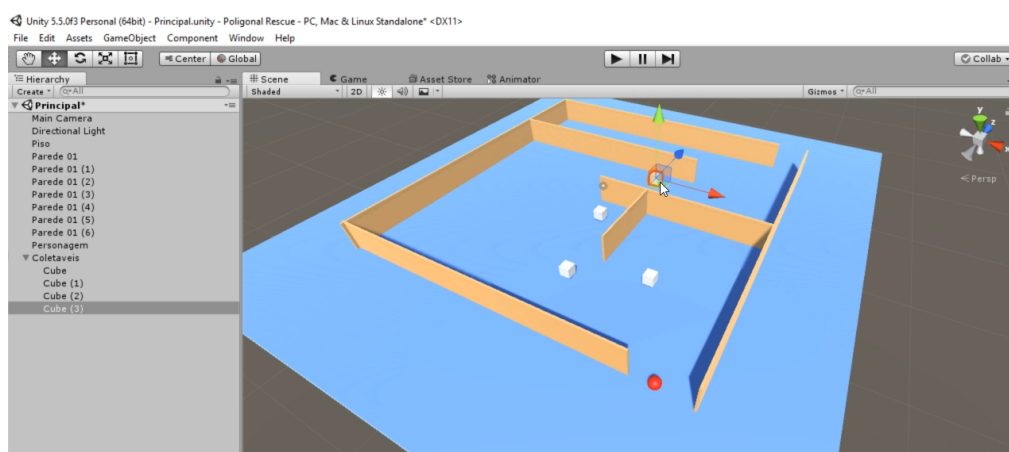
Se o objeto tiver a Tag ItemColetavel, usaremos o comando `Destroy(other.gameObject)` para remover efetivamente o objeto da cena. Execute o jogo e movimente o jogador para colidir com o Cubo. Então você verá que agora ele é removido da cena.

## Criando outros Itens Coletáveis

Como você já criou um Cubo marcado com a Tag ItemColetavel e também criou adequadamente o script Coletaltem no personagem para remover esses itens da cena, agora poderá criar outros itens coletáveis facilmente, duplicando o cubo atual da cena.

Clique no Cubo e digite CTRL+D para duplicá-lo. Uma nova cópia idêntica do Cubo será criada já com a Tag configurada corretamente. Mova essa cópia para outro local da cena e repita o processo até criar alguns Cubos coletáveis pelo labirinto. Veja na **Figura 21**.

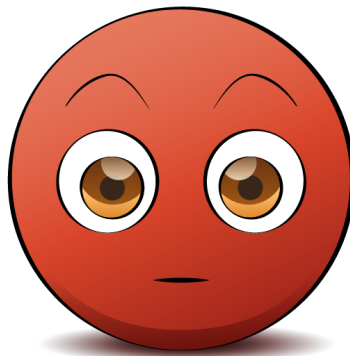
**Figura 21** - Vários Cubos com a Tag ItemColetavel criados pelo labirinto.



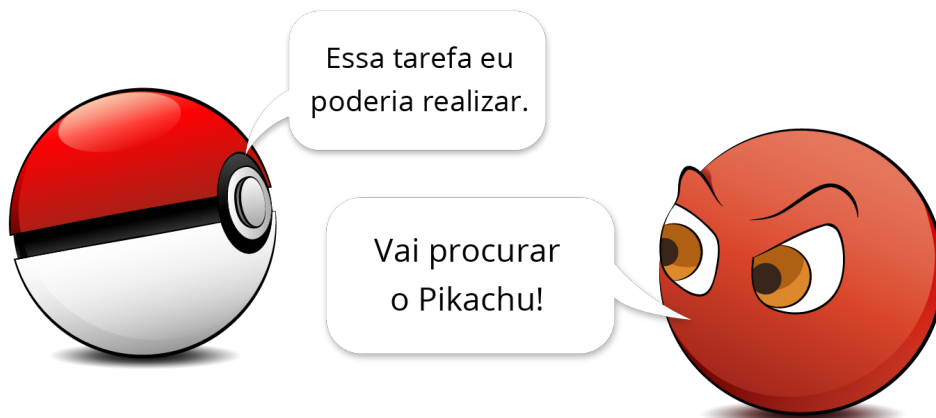
**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Execute o jogo e tente coletar todos os Cubos criados. No nosso jogo *Polygonal Rescue*, coletar um item significa que você está resgatando um polígono perdido, então já pode se sentir um herói!

**Figura 22** - Não se sentindo um herói ainda...



Execute o jogo e tente coletar todos os Cubos criados. No nosso jogo Polygonal Rescue, coletar um item significa que o personagem está resgatando um polígono perdido. Então o jogo já tem sua primeira mecânica de interação com objetos implementada, que se refere a coletar Itens.



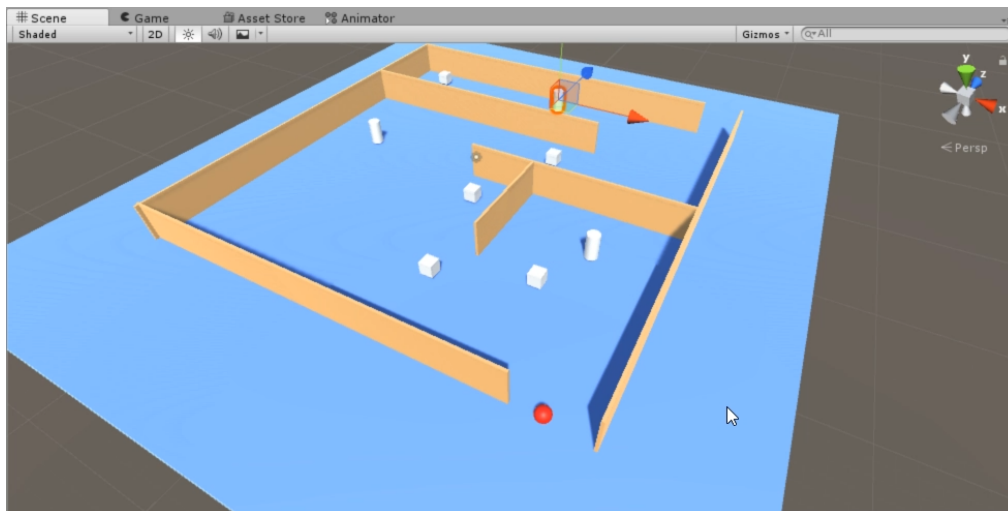
## Criando Coletáveis de Outros Tipos

---

Com nosso sistema de resgate de polígonos pronto (coleta de itens), podemos agora criar outros objetos para também serem resgatados pelo nosso personagem, de modo que não fiquemos presos somente a Cubos. Crie um objeto do tipo

Cylinder, posicione-o no labirinto similar a um dos Cubos (próximo ao piso) e adicione nele também a Tag ItemColetavel. Agora duplique esse Cylinder algumas vezes e posicione as cópias em outros locais do labirinto, veja na **Figura 23**.

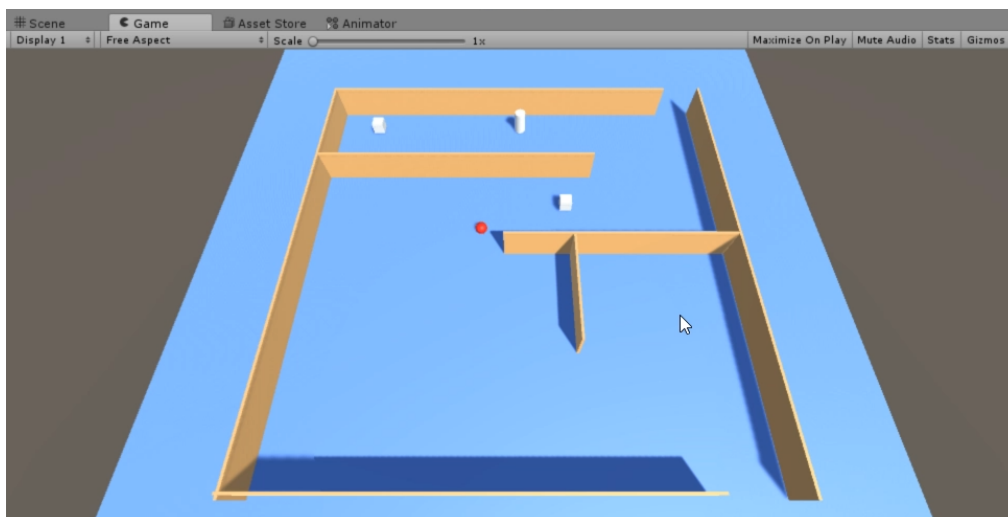
**Figura 23** - Polígonos coletáveis (resgatáveis) do tipo Cube e Cylinder posicionados no labirinto.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Inicie novamente o jogo e veja a possibilidade de coletar normalmente todos os itens, tanto Cubos como Cilindros, já que todos eles têm a Tag ItemColetavel. A **Figura 24** mostra o jogo em andamento com alguns Polígonos já resgatados pelo personagem.

**Figura 24** - Jogo em andamento com alguns polígonos já resgatados.

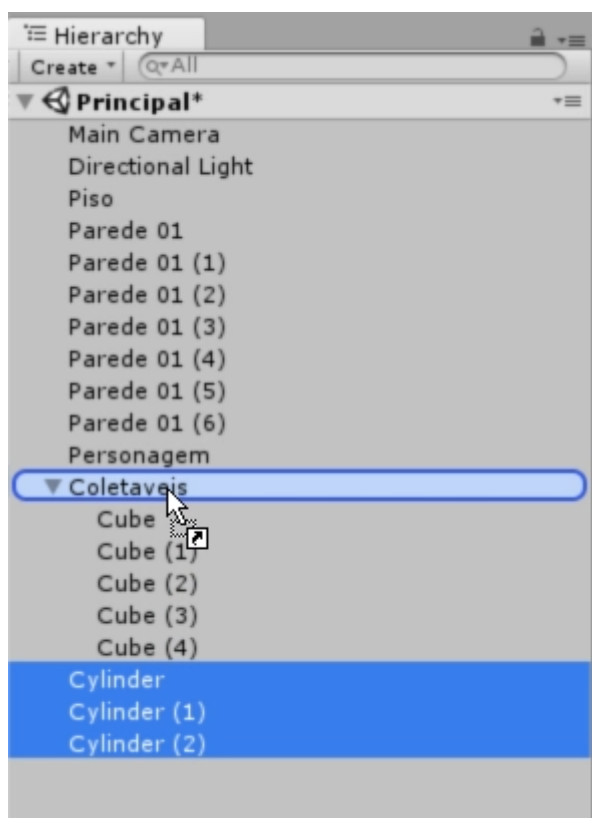


**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Esse sistema é muito poderoso, pois apenas requer que o objeto tenha essa Tag para ser considerado um objeto a ser resgatado pelo personagem. Assim você pode estender o seu jogo criando GameObjects bem mais complexos, por exemplo o modelo 3D criado em outros programas como o Blender, Maya ou baixando de sites especializados na Internet, bastando adicionar no objeto a Tag correta.

Se ainda não fez isso, aproveite agora para organizar seus GameObjects no hierarchy, movendo os Cilindros adicionados para cima do GameObject “Coletaveis” tornando-os filho dele, assim como os Cubos, conforme a **Figura 25**.

**Figura 25** - Organizando todos os Coletáveis.



**Fonte:** Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/>. Acesso em: 23 fev. 2017

Salve sua cena e pronto! Temos nossa primeira versão do jogo. O personagem pode resgatar os polígonos 3D perdidos no labirinto.





Nas próximas aulas vamos implementar mais recursos no *Polygonal Rescue*, como a contabilização da pontuação do jogador, criação de múltiplas fases, etc. Bom, por hoje encerramos a nossa aula, lembrando que qualquer dúvida pode ser esclarecida nos encontros presenciais. Bons estudos e até a próxima aula!

## Resumo

---

Nesta aula aprendemos a criar os itens coletáveis do jogo, ou seja, os polígonos perdidos que o nosso personagem precisa resgatar. Utilizamos uma técnica chamada de colisores (Collider) para detectar se um objeto entrou em contato com outro. Criamos também um sistema para identificar se esse contato foi entre o personagem e um polígono através do uso de Tags. No Unity, os Colliders são fundamentais para a criação de jogos e, portanto, demos mais um passo importante na nossa trajetória de aprendizagem.

## Leitura Complementar

---

O sistema de colisão do Unity é bastante completo e nesse curso abordaremos algumas de suas funcionalidades.

Para se aprofundar mais no assunto você pode consultar a documentação do Unity no seguinte link: <https://docs.unity3d.com/Manual/CollidersOverview.html>

## Autoavaliação

---

1. Crie outros tipos de polígonos coletáveis utilizando objetos primitivos do Unity.
2. O que aconteceria se fosse adicionado aos itens coletáveis um Rigidbody?

# Referências

---

UNITY TECHNOLOGIES. 2016 (C). Unity 3D Online Tutorials [online]. Disponível em: <https://unity3d.com/pt/learn/tutorials> [Acessado em 16 de novembro de 2016].

K. Aava Rani. 2014. Learning Unity Physics. Packt Publishing

STOY, C. 2006. Game object component system. In Game Programming Gems 6, Charles River Media, M. Dickheiser, Ed., Páginas 393 a 403.

MARQUES, Paulo; PEDROSO, Hernâni - C# 2.0 . Lisboa: FCA, 2005. ISBN 978-972-722 208-8

UNITY TECHNOLOGIES. 2016 (C). Unity 3D Manual [online]. Disponível em: <https://docs.unity3d.com/Manual/index.html> [Acessado em 16 de novembro de 2016]