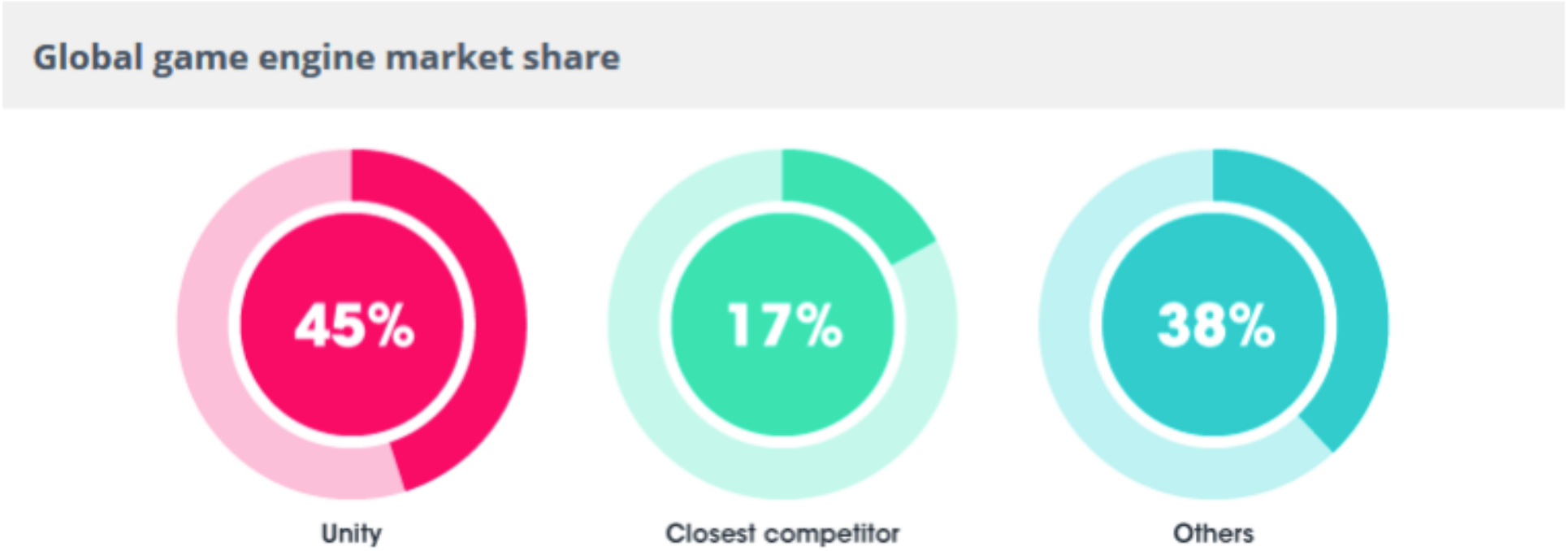
MOVIMENTO HORIZONTAL(Andar) E FÍSICA BÁSICA

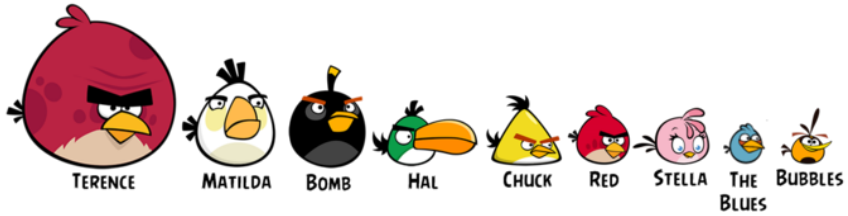
PROF. ME. HÉLIO ESPERIDIÃO



Unity

Fatia de mercado mundial para o desenvolvimento de jogos.

34% dos 1.000 maiores jogos mobile gratuitos são feitos com Unity.



Popularidade do Unity

Essa enorme popularidade faz jus à sua capacidade: a game engine permite criar jogos em 2D ou 3D com os mais diversos estilos de gráficos e mecânicas e para diferentes plataformas.

Muitos jogos famosos, como Angry Birds 2, Bad Piggies, Roller Coaster Tycoon World e até o Pokemon GO foram criados com ela

Mais Jogos criados com Unity

Assassin's Creed: Identity
Temple Run Trilogy
Deus Ex: The Fall
Ballistic Overkill
Knights of Pen and Paper

Game Engine

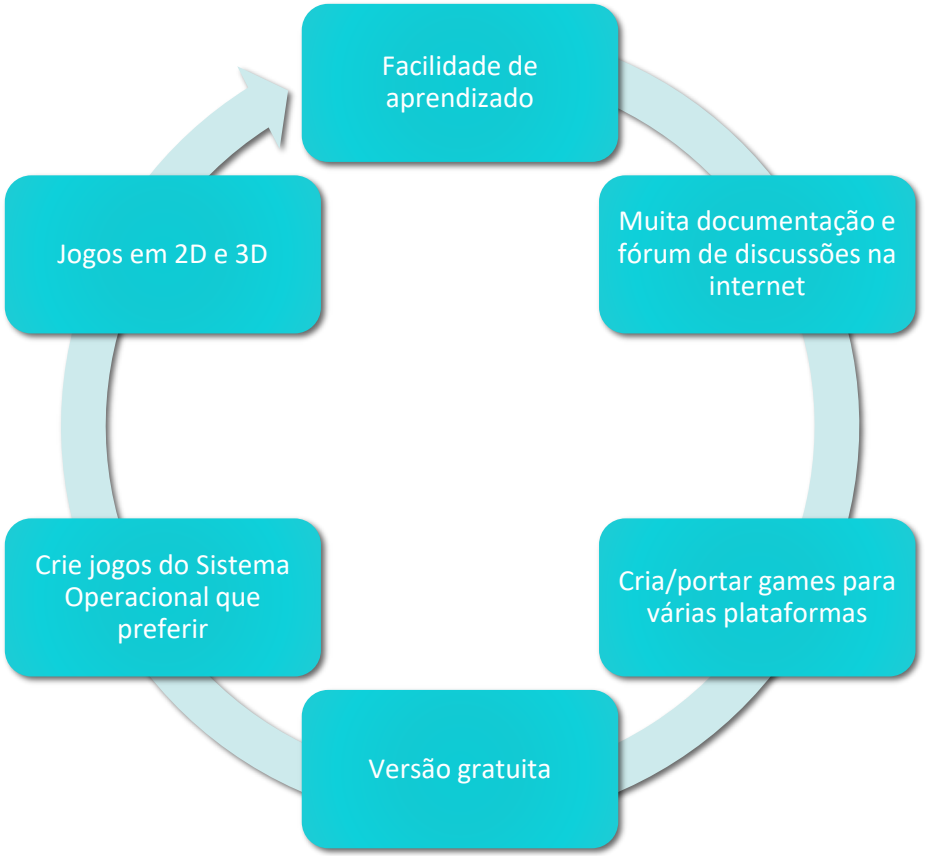
01

Game engine (ou, em português, motor de jogo), consiste em um conjunto de ferramentas capazes de facilitar o desenvolvimento de um jogo.

02

Geralmente possuem recursos para criação de funções gráficas, física aos objetos, trilhas sonoras, entre outras ações.

Por que é o mais usado?



Assets

Asset significa: Ativo.

Em tecnologia os "assets" são os recursos do seu projeto. O seu "banco" de bibliotecas.

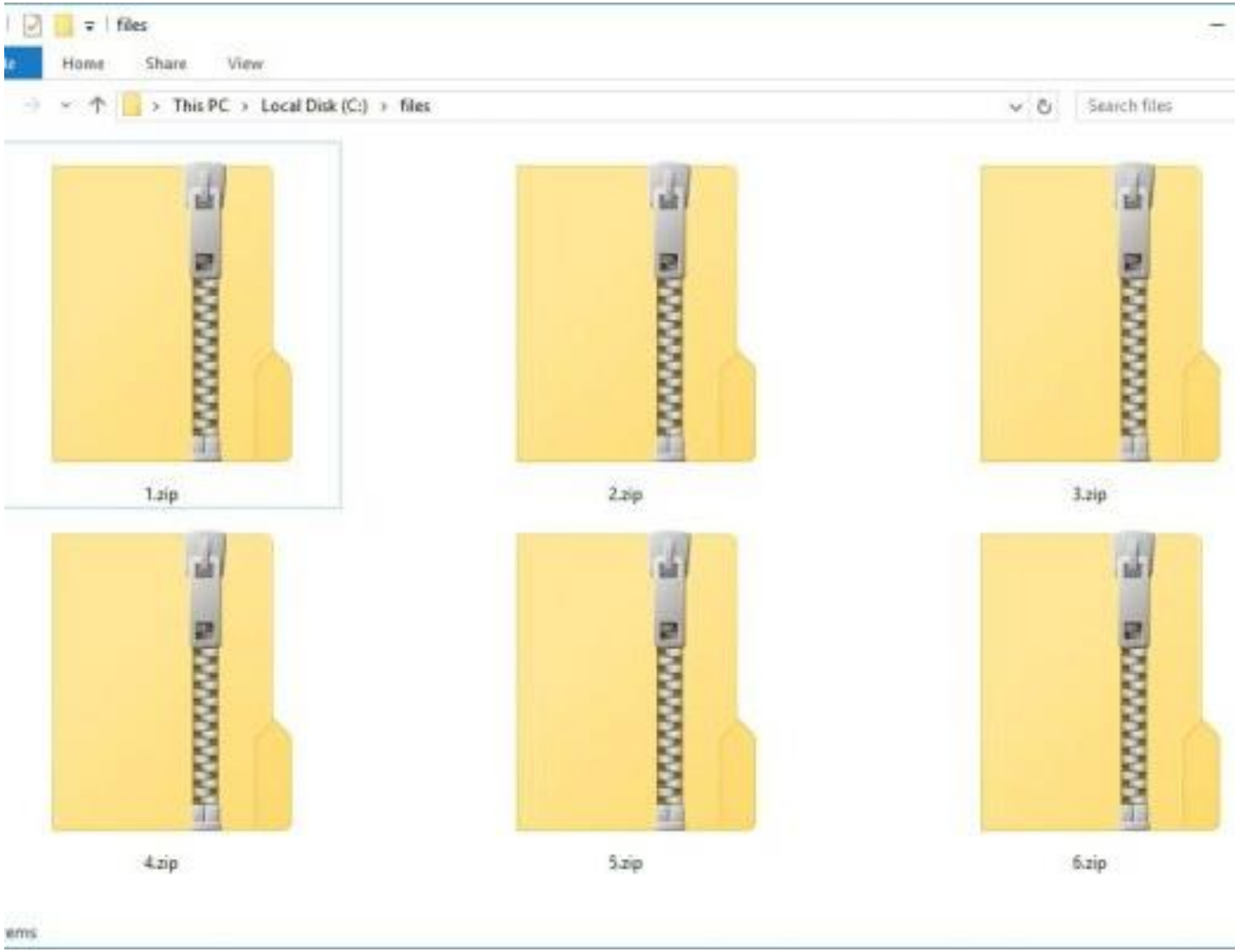
No caso de jogos pode ser entendido como o conjunto de imagens, gráficos, sons e recursos que são utilizados para o desenvolvimento do jogo.

Onde encontrar?

Os assets podem ser construídos pelo próprio desenvolvedor ou encontrados na web nas seguintes formas de distribuição: Gratuito ou pago.

Site com conteúdo gratuito:

- https://www.gameart2d.com/freebies.html



* Arquivos compactados (.zip)

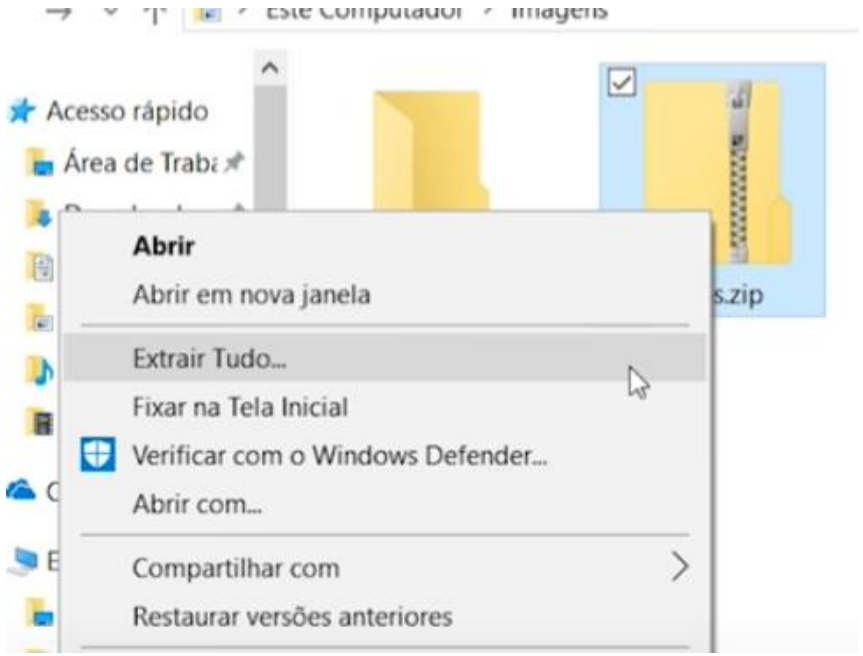
É bem provável que os assets que você baixe estejam compactados ou zipados.

Arquivos zipados (compactados) ocupam menos espaço de armazenamento e podem ser transferidos para outros computadores mais rapidamente do que arquivos não compactados.

* Arquivos compactados (.zip)

https://www.youtube.com/watch?v=pg7QWAsUvzA

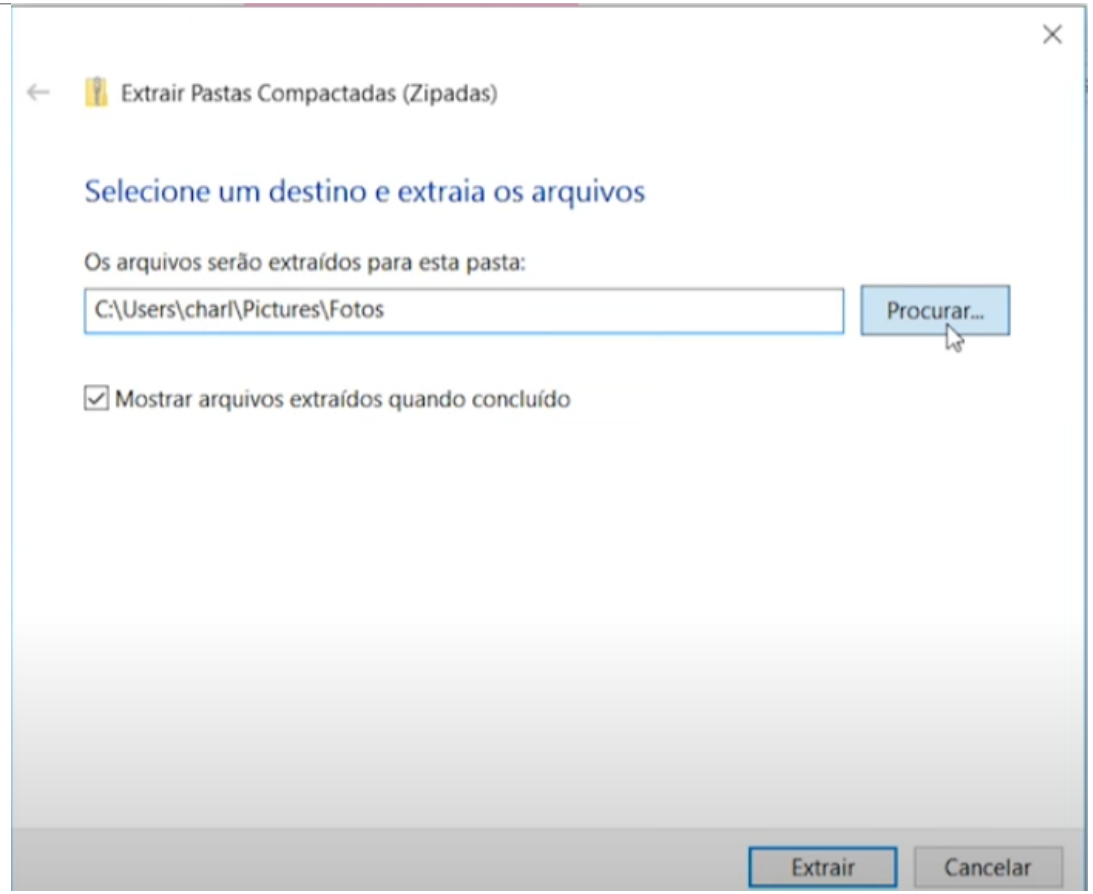
Para descompactar click no arquivo .zip com o botão direito do mouse e escolha a opção “Extrair Tudo”



* Arquivos compactados (.zip)

Selecione uma pasta para salvar os arquivos que serão descompactados

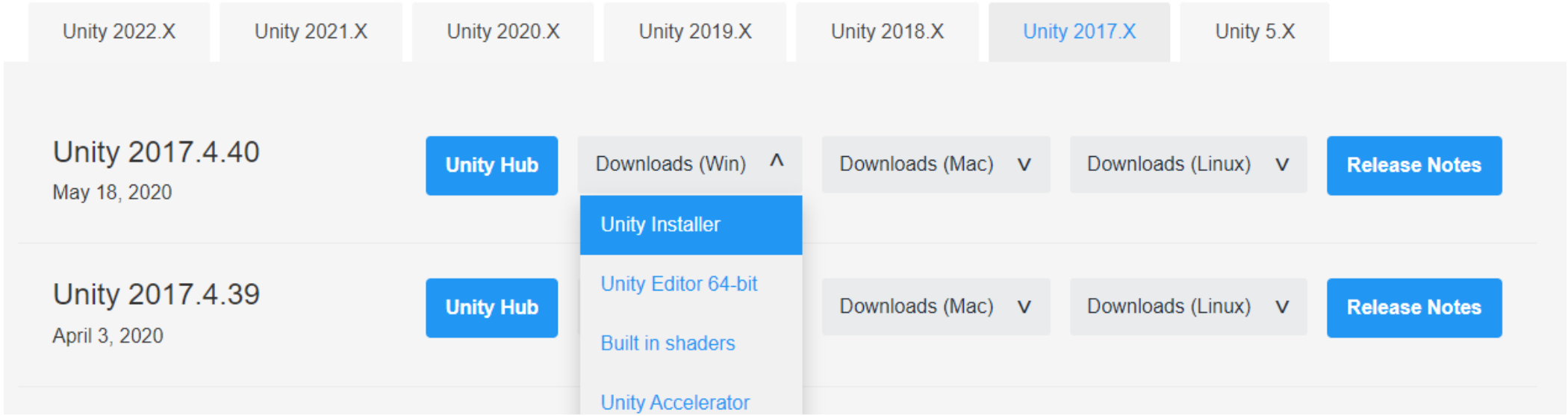
Depois de escolher uma pasta click em “Extrair”



Versão do Unity: 2017.4.40

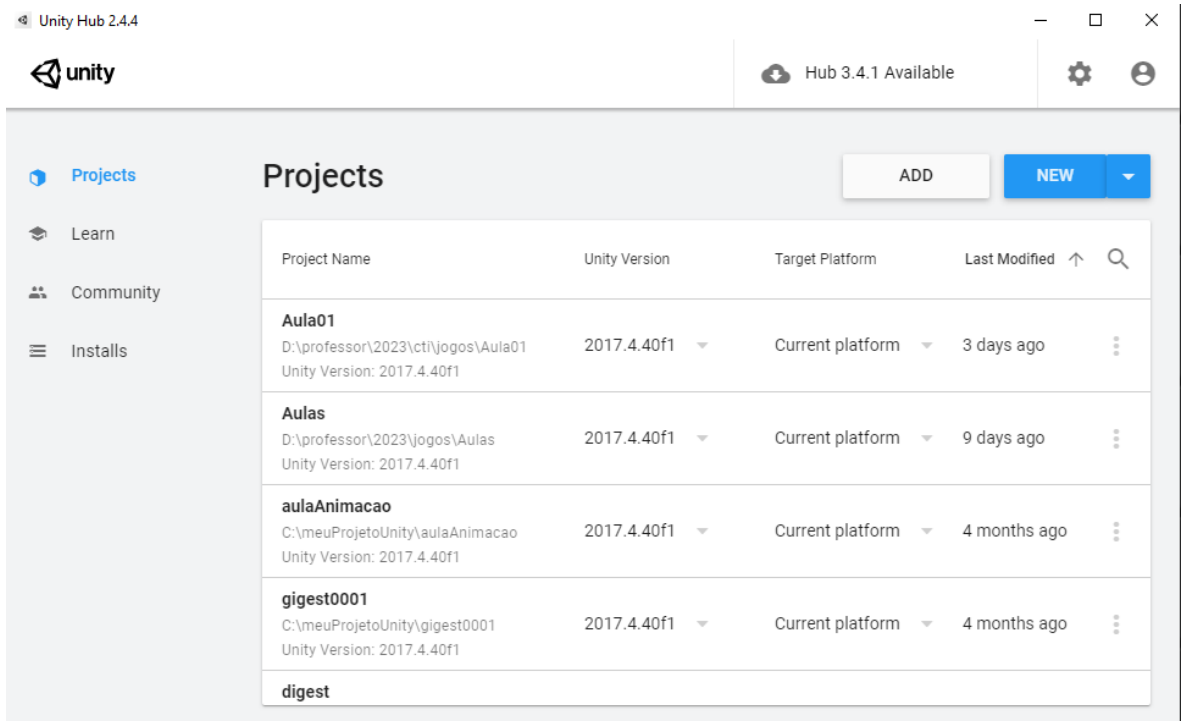
Link para download: https://unity.com/releases/editor/archive

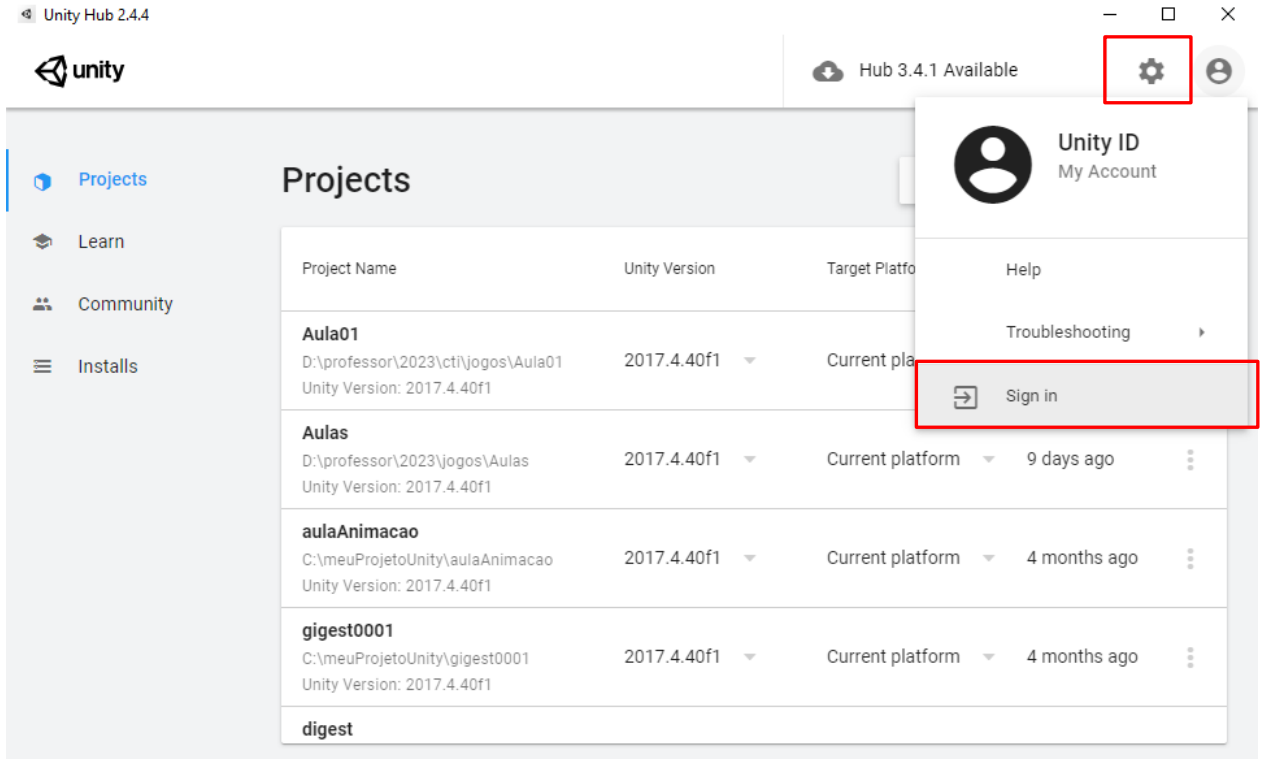
Instale o **Unity hub** e depois instale o **Unity installer**



Abra o Unity hub

Vá em **”SING IN”** e registre-se para adquirir uma licença gratuita





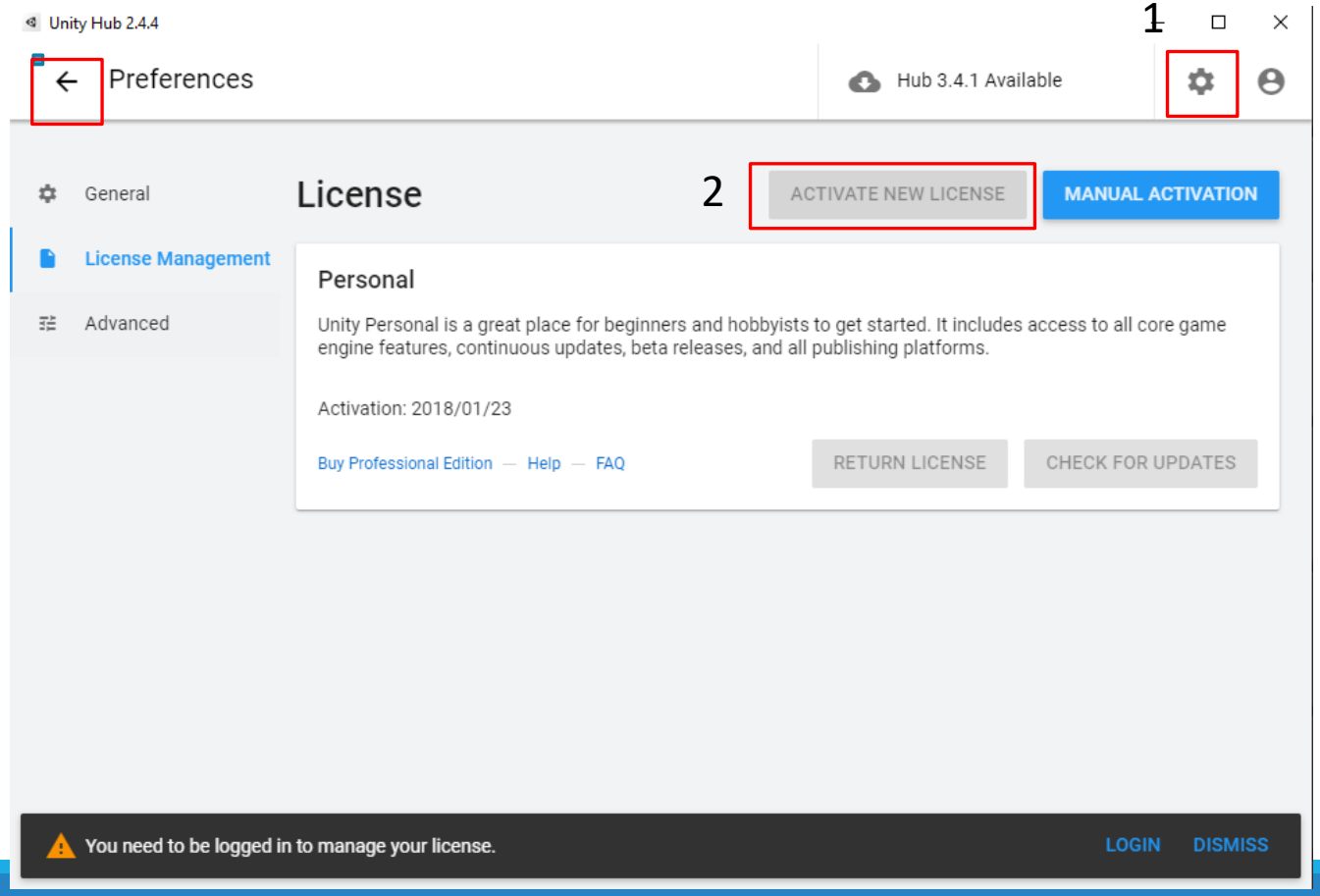
Após o registro

Click na **”ENGRENAGEM”** e depois no menu **“License Management”** >> **“Active new license”**.

Você precisa estar “logado” na plataforma.

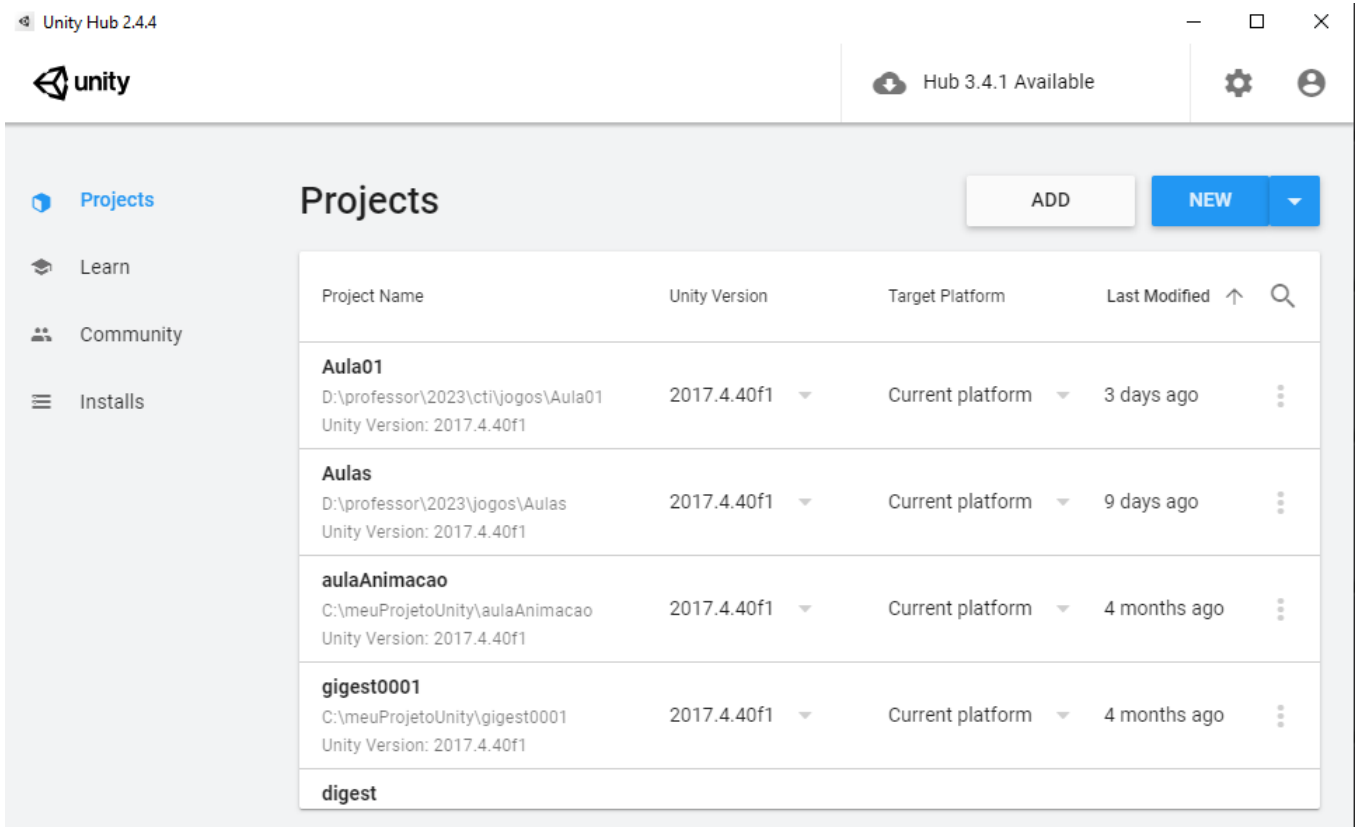
Escolha a licença de não uso profissional do unity

Depois de ativada a licença você deve clicar na seta direcional para esquerda.



Criar um novo projeto

Click no botão Azul “NEW”



Configurações do projeto

Escolha a opção 2D

Defina um nome para o projeto em “Project name”

“Location:” Defina onde todos os arquivos do projeto serão salvos.

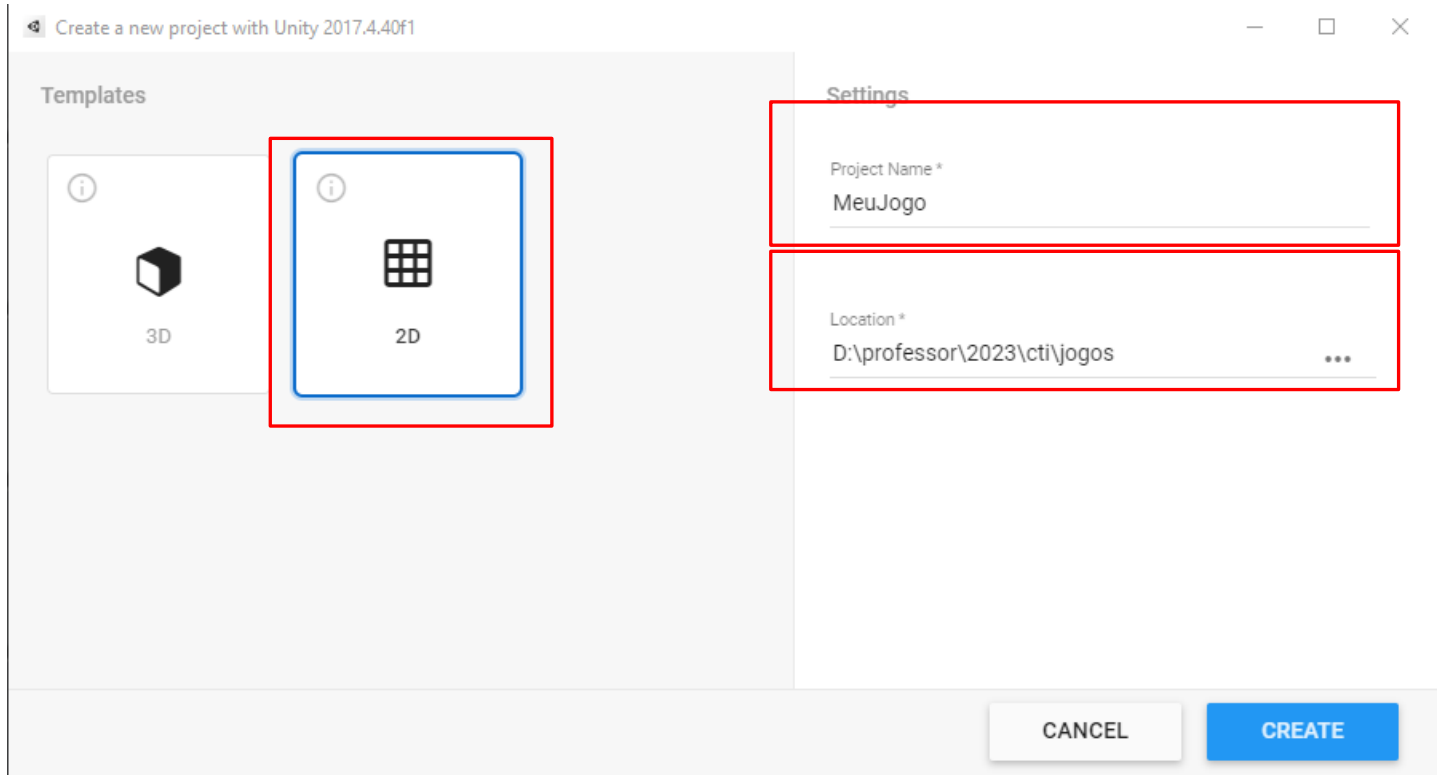
No exemplo os arquivos serão salvos em:

D:\professor\2023\cti\jogos\MeuJogo

Portanto salve o conteúdo inteiro da pasta “MeuJogo”

O projeto é composto de muitos arquivos

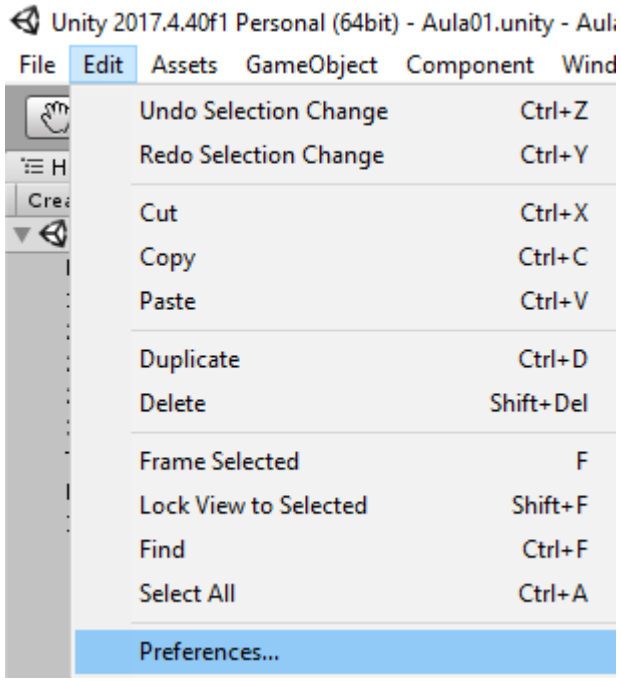
Todos os arquivos e pastas devem ser salvos



Configurar o Editor de código monoDevelop

É necessário realizar esse procedimento apenas uma vez.

Ao carregar o Unity vá o menu: **“Edit”**>>**”Preferences…”**



MonoDevelop

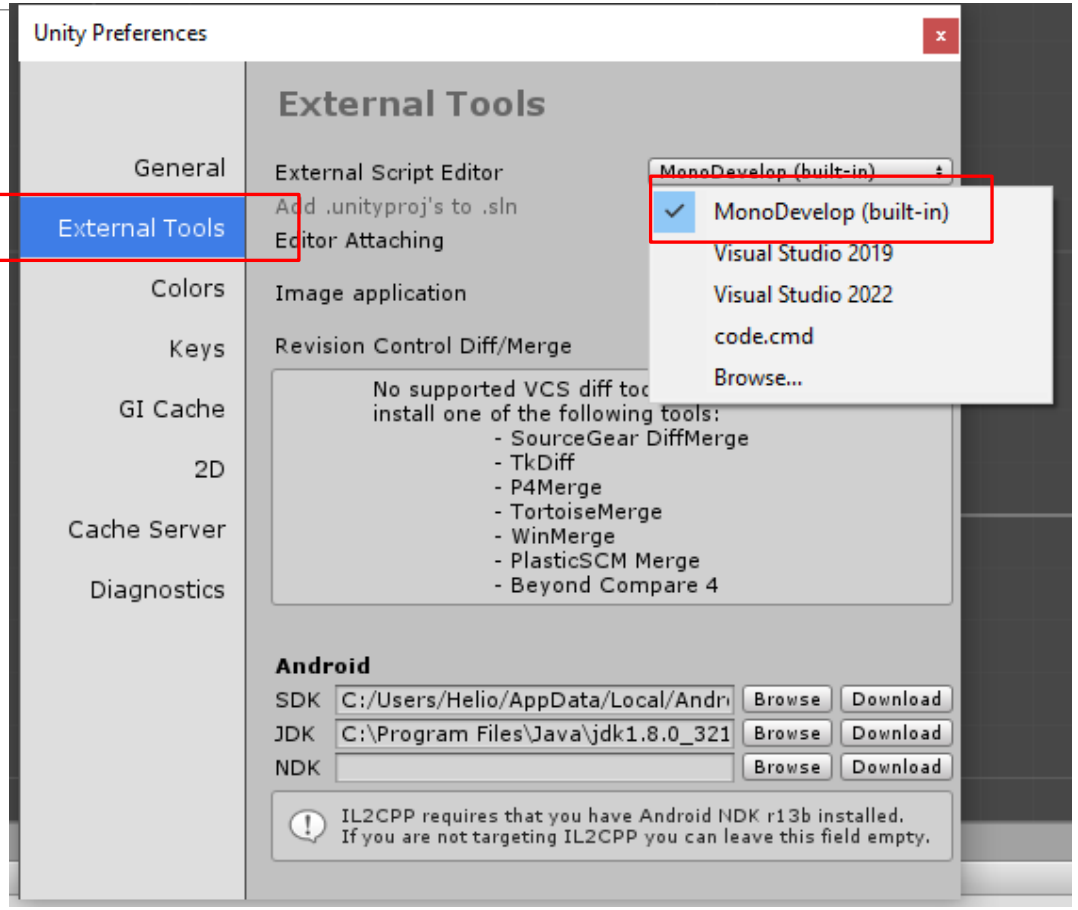
Click em “**External Tools**”

Escolha a opção “MonoDevelop (Built-in)”

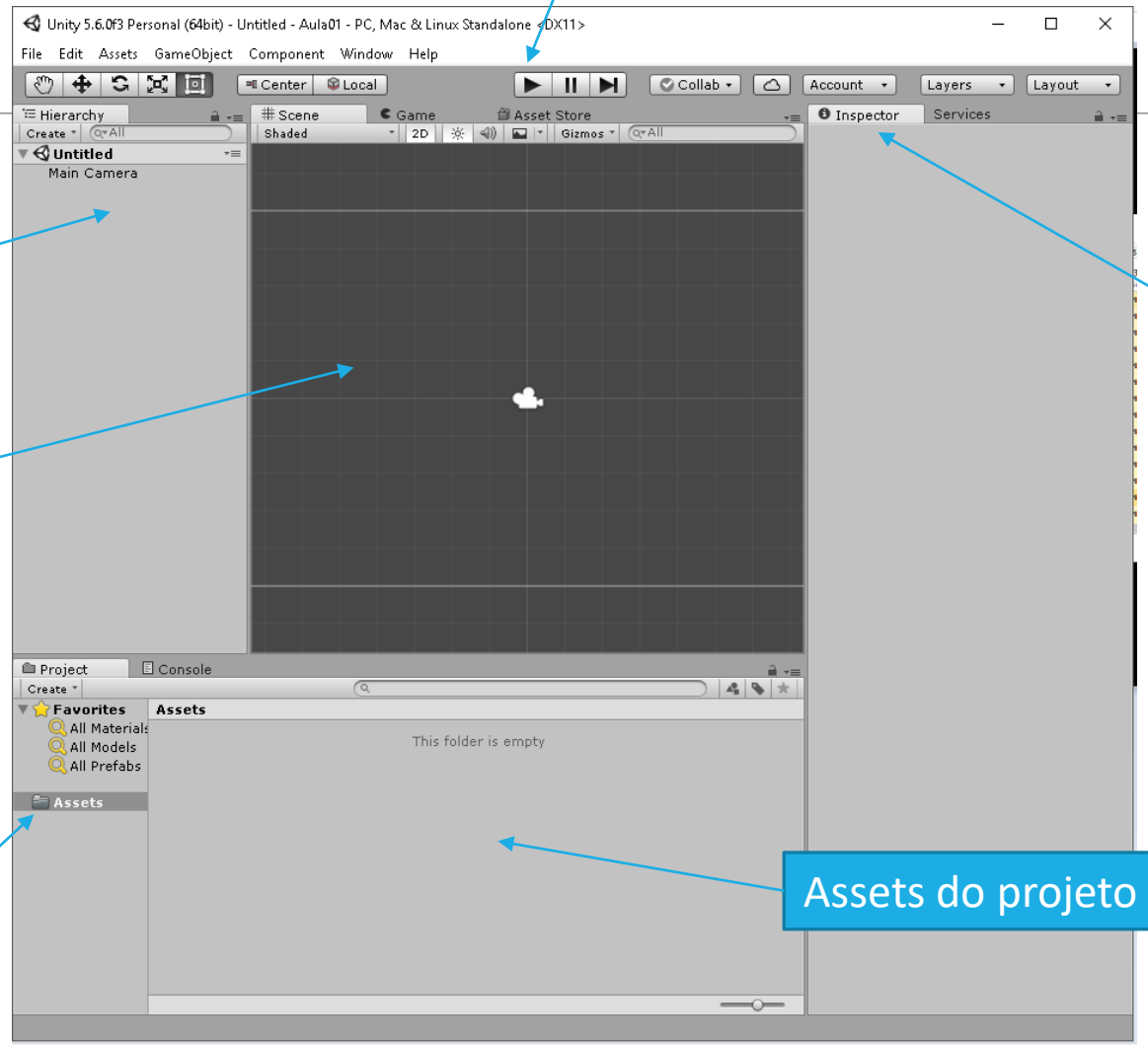
Caso seja necessário o unity indicará um link para download.

- Caso necessário faça o download e instale o **monoDevelop**.

Feche a janela “**Unity Preferences**”



Unity



Executa o jogo

Lista de elementos de cena

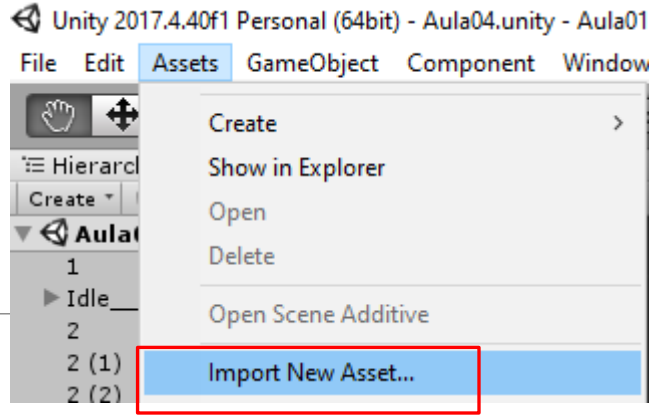
Cenário

Pastas do projeto

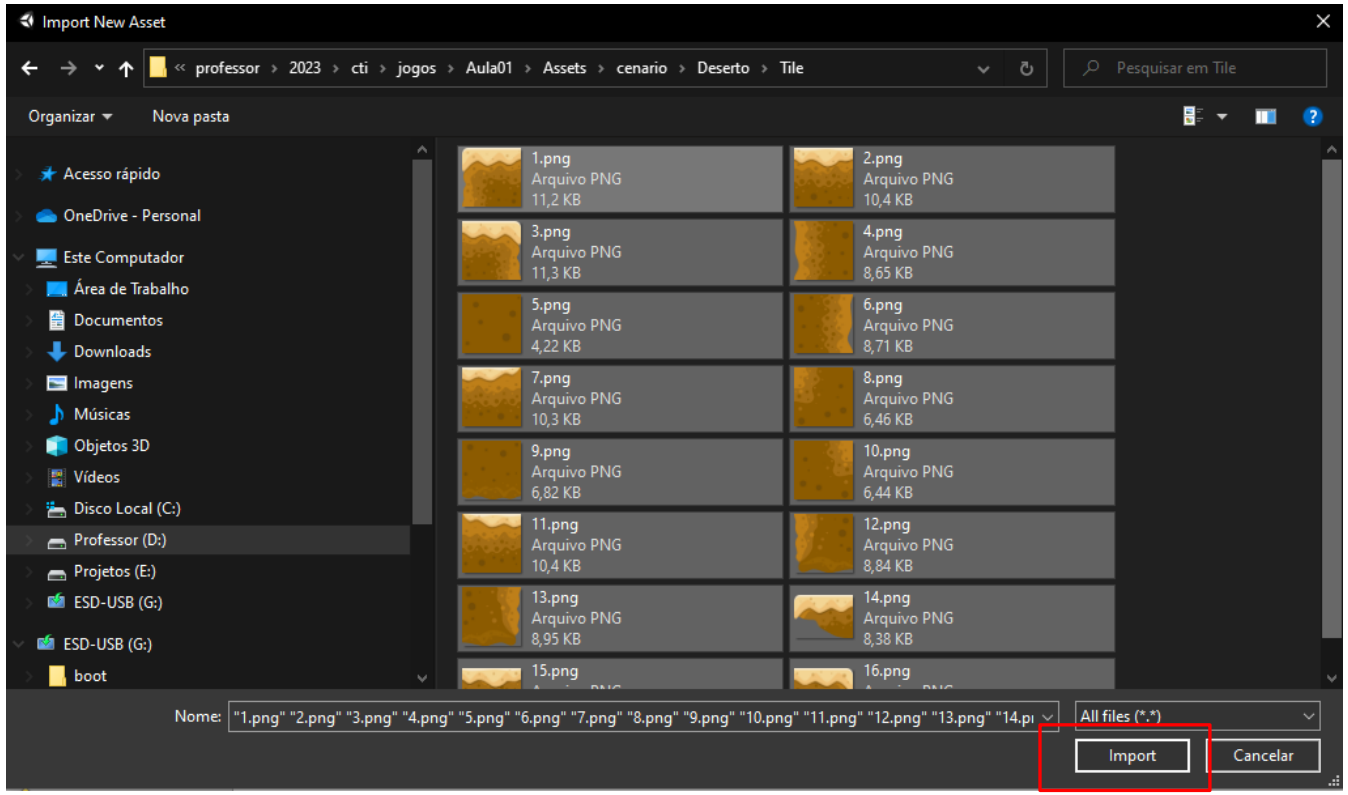
Propriedades de um elemento de cena

Assets do projeto

Acrescente seus Assets

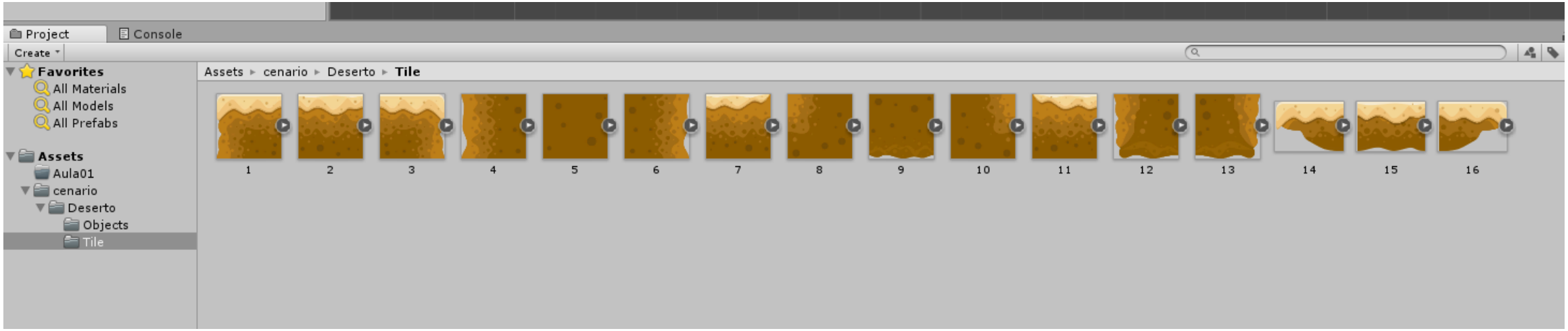


- Click no menu: **Assets** >> **Import new Asset**
- Selecione os arquivos de assets desejados e click em **Import**
- Caso possua muitos assets crie mais pastas dentro da pasta Assets para manter a organização



Após importar

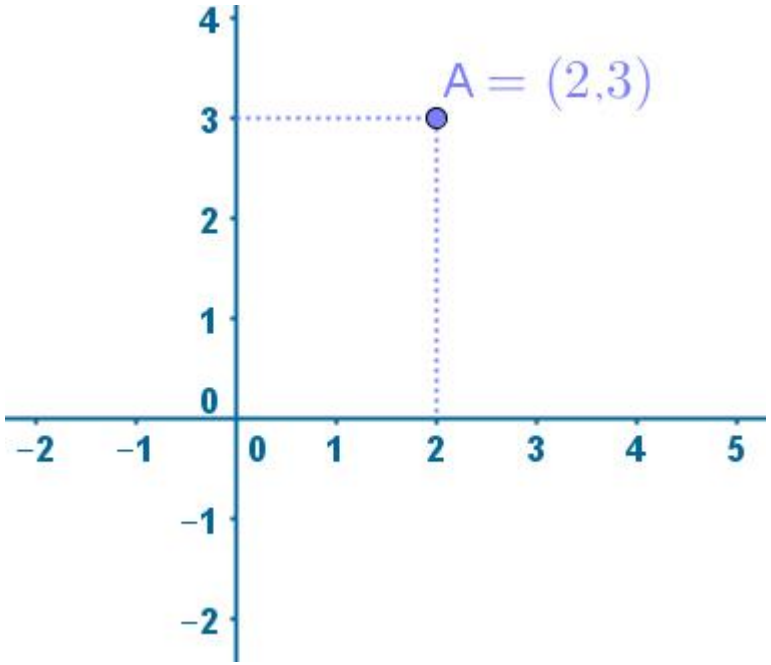
Na parte inferior da interface padrão do Unity é possível visualizar os assets que foram importados para a plataforma.



Posição no plano 2d

O Unity trabalha com conceitos de posicionamento no plano.

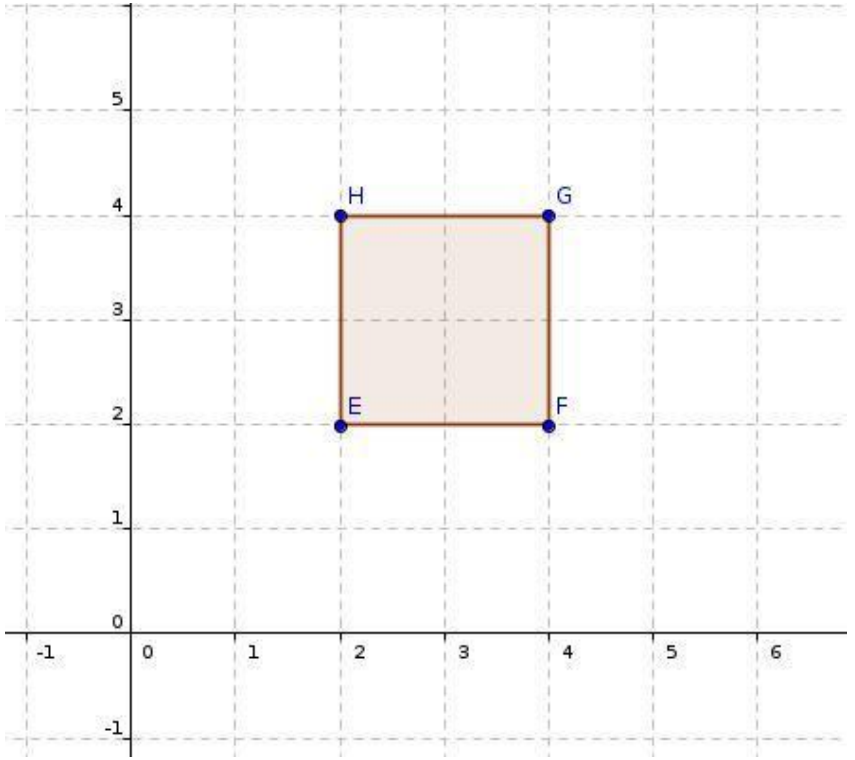
Observe que o ponto **A** possuí **x=2 e y=3**



Posição dos assets no cenário 2d

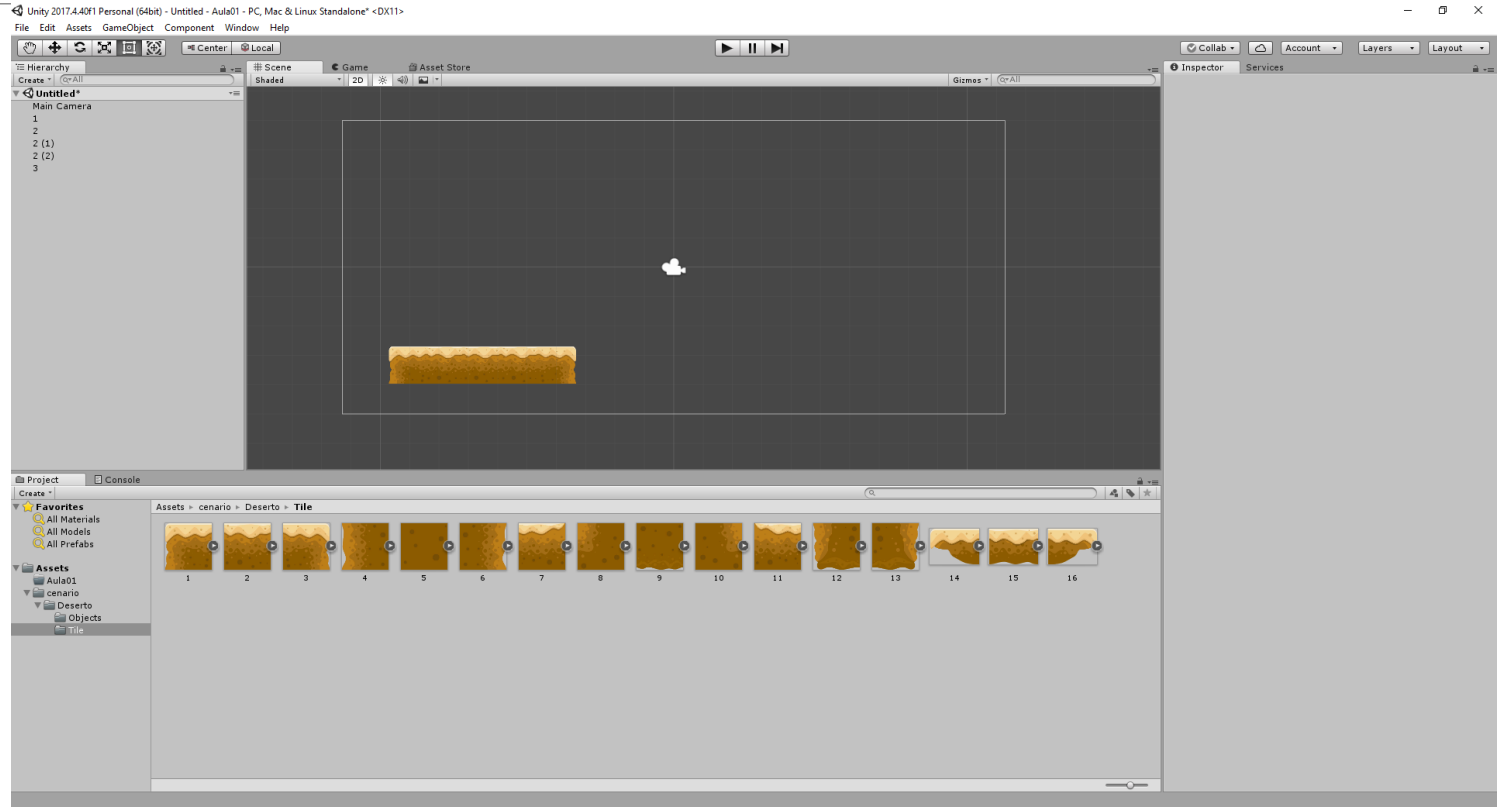
Todo asset posicionado no cenário é arranjado conforme as posições x e y.

Observe ao lado um bloco posicionado no plano.

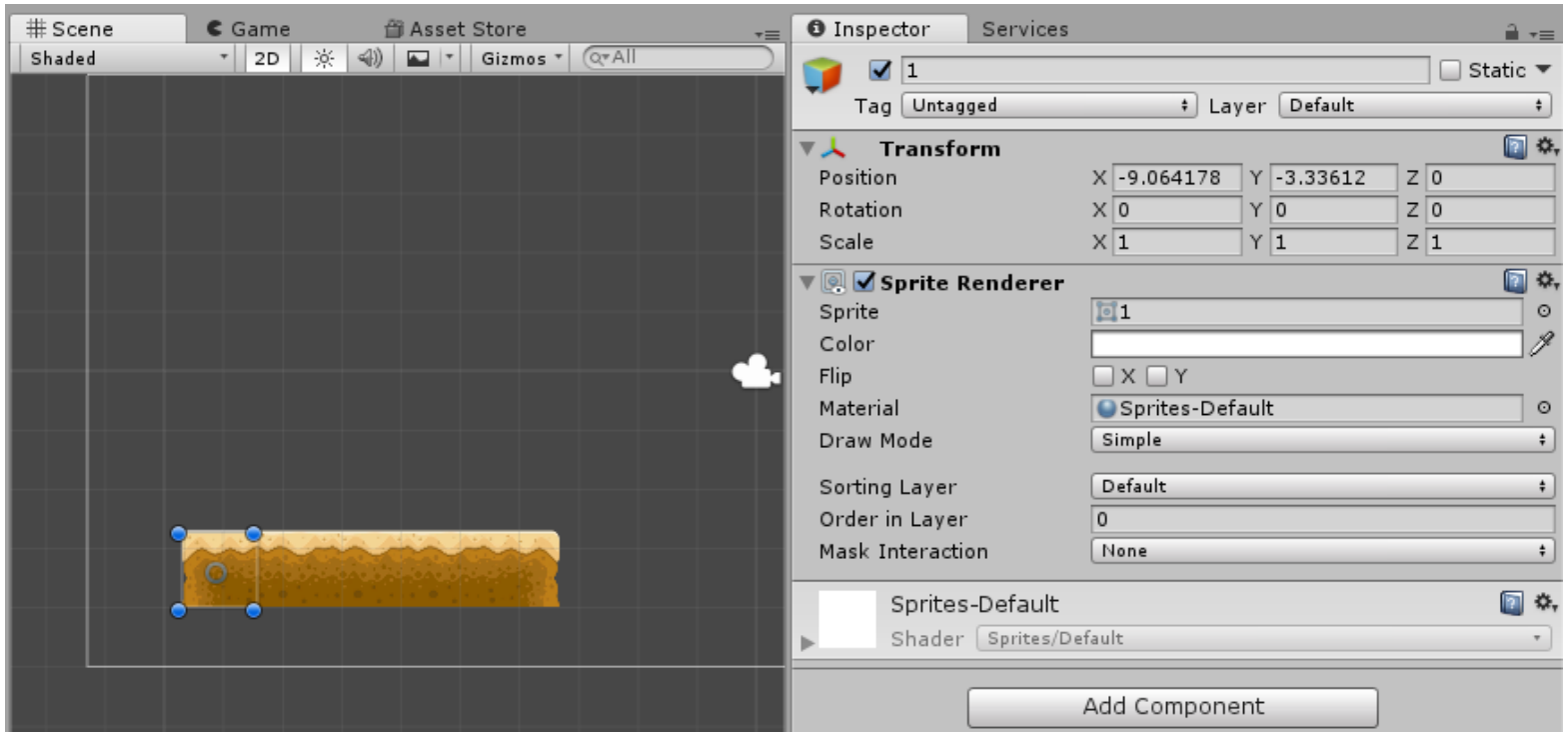


Posicione os Assets no seu cenário

Arraste os assets para dentro do cenário e os posicione no plano.



Propriedade: Position

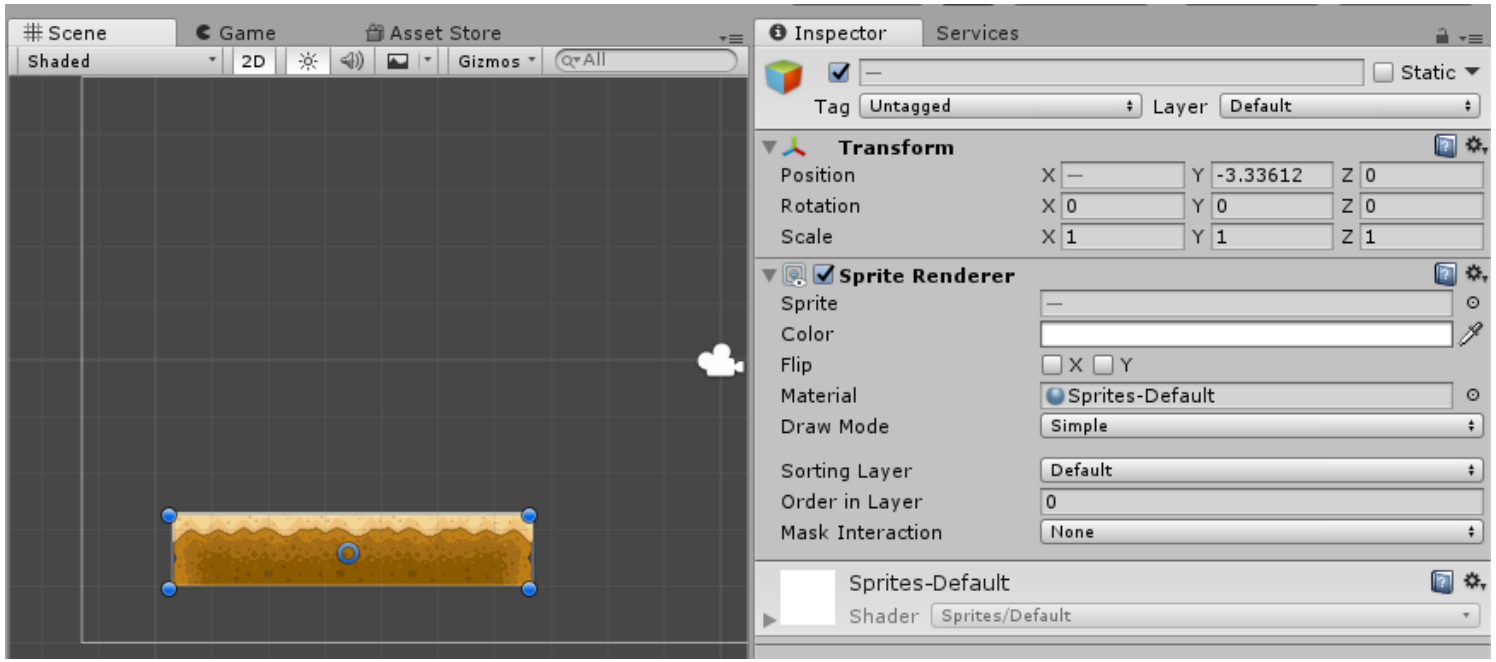


Cada elemento posicionado no cenário possui uma série de propriedades

Ao clicar no objeto de cena conforme figura ao lado é possível verificar todas as suas propriedades.

Exemplo de propriedade: **Position: x,y e z** equivalem a posição de cada elemento dentro do cenário.

Propriedade: Position

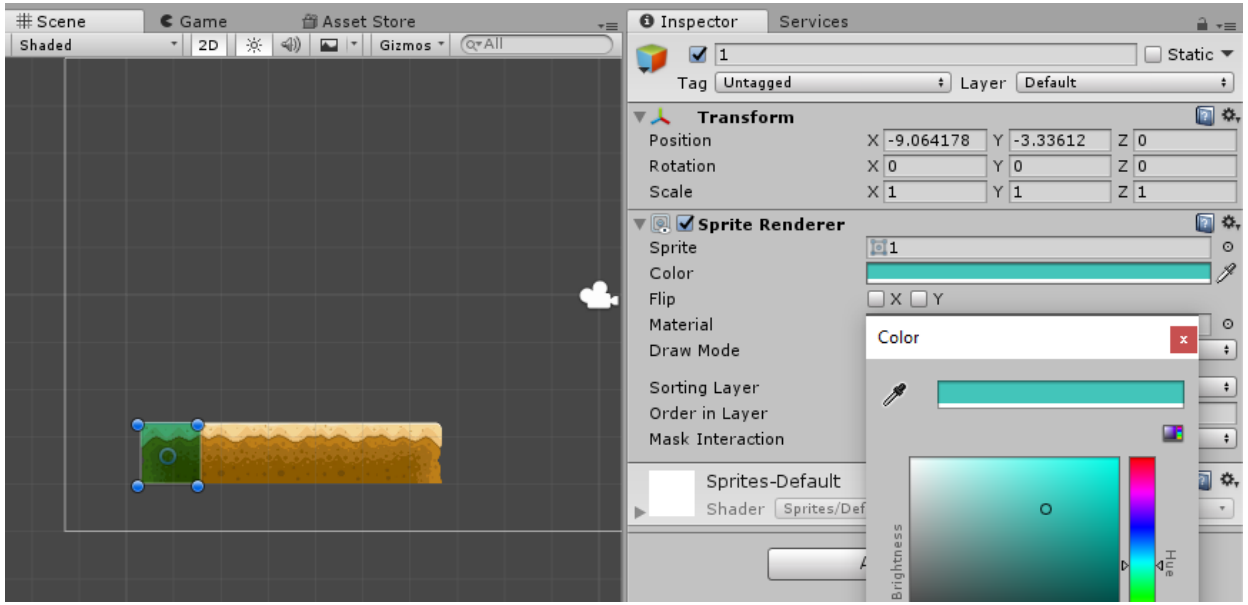


A imagem ao lado representa a seleção de um conjunto de objetos de jogo.

É possível verificar que a propriedade Position Y de todos eles é igual.

Isso permite que aconteça uma alinhamento em **y**, ou seja não há um elemento com alinhamento mais alto que o outros, todos possuem a mesma altura de alinhamento.

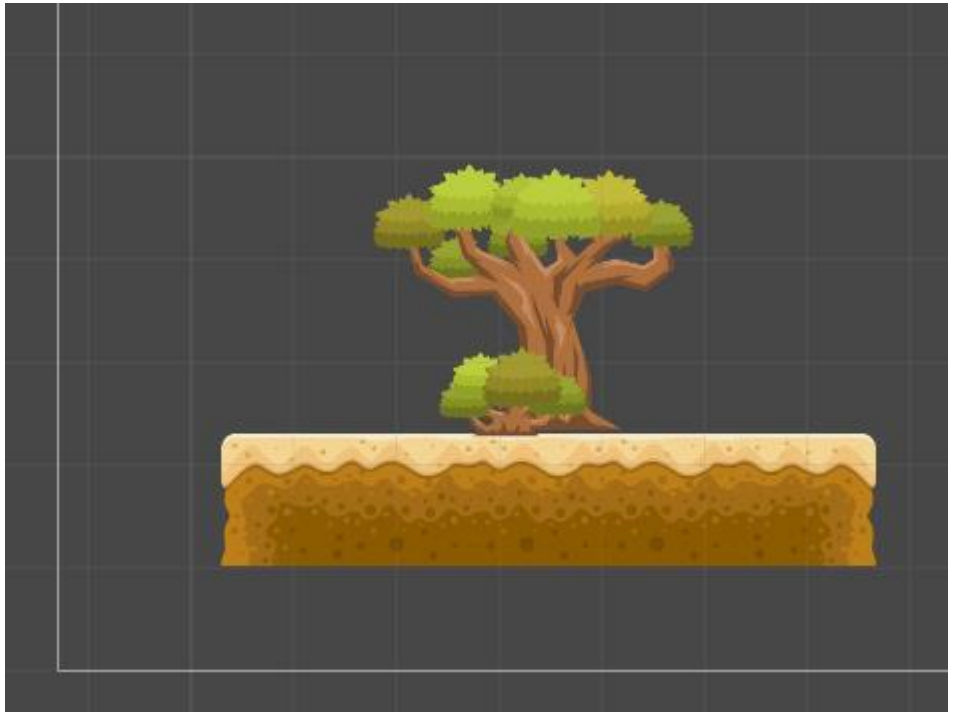
Propriedade: Color



Existem outras propriedades além das propriedades de posicionamento.

A propriedade **color** permite aplicar um filtro de cor a um determinado **Asset** conforme figura ao lado.

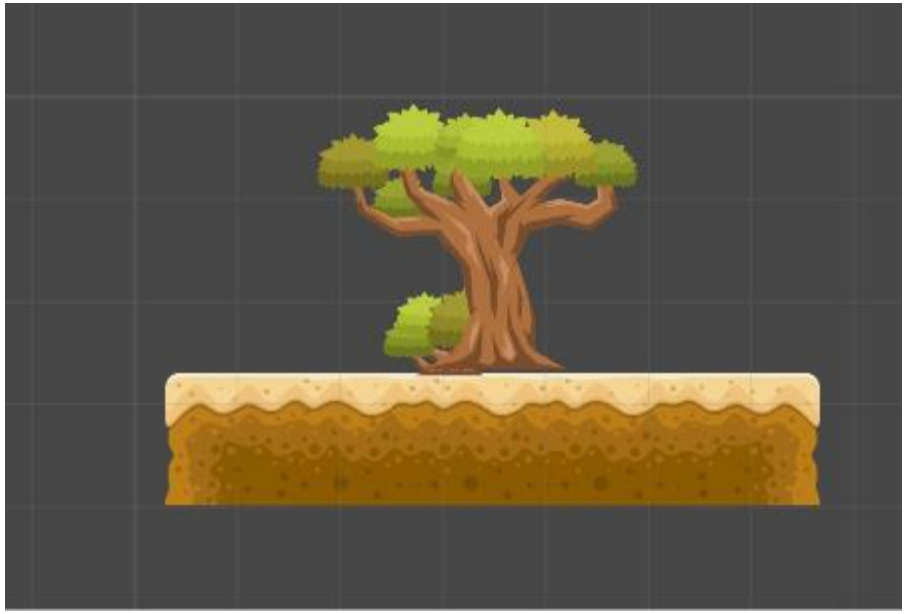
Propriedade: **Order Layer**



Essa propriedade permite definir quando um objeto é posicionado a frente ou atrás de outro objeto.

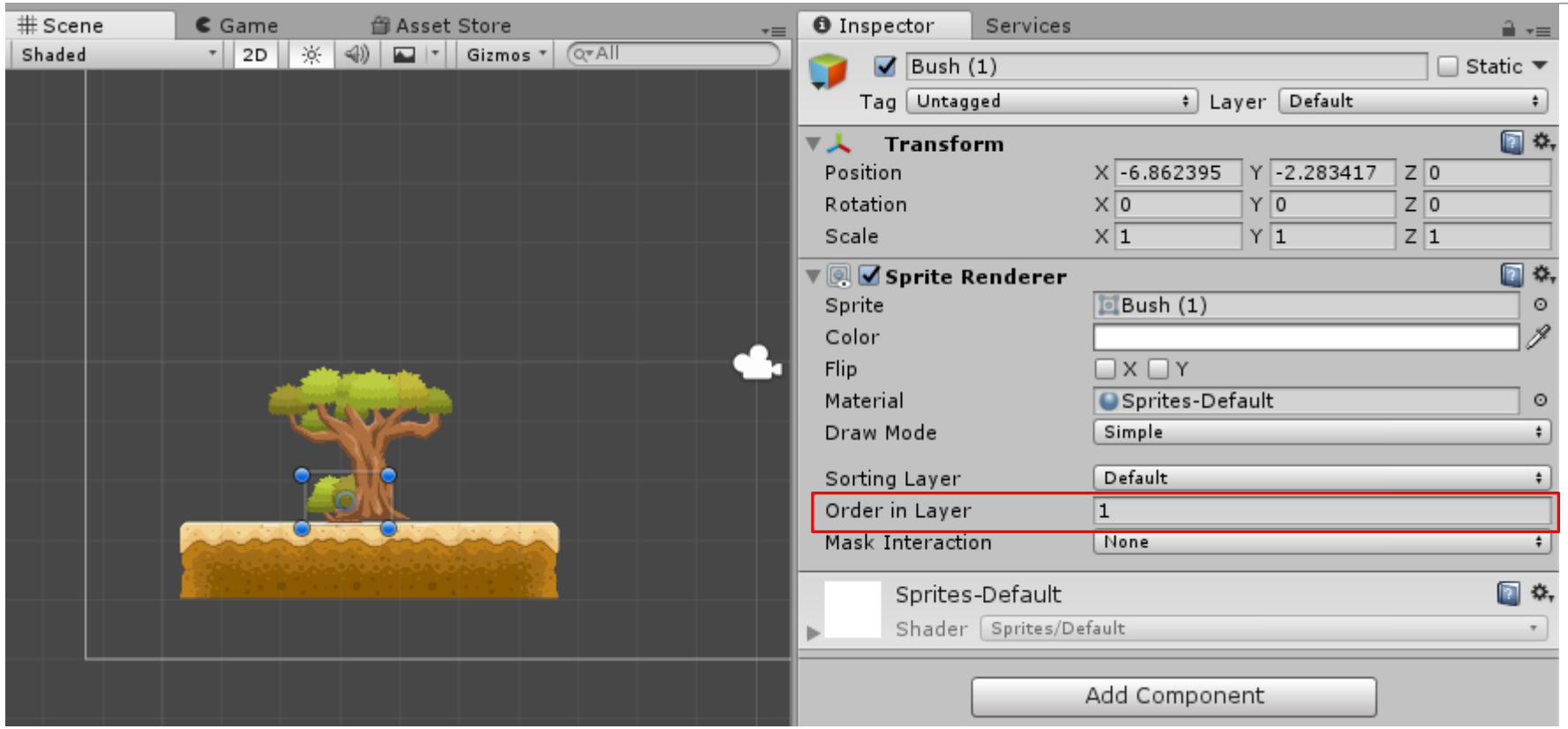
Observe a figura ao lado, o arbusto deve possuir um **order layer** maior que o **order layer** da árvore, pois o arbusto está a frente da árvore.

Propriedade: Order Layer



Observe a fígura ao lado, o arbusto deve possuir um order layer menor que o order layer da árvore, pois o arbusto está atrás da árvore.

Como mudar o order layer?



Para mudar o order layer click sobre o elemento no cenário e procure a propriedade order layer

Conceitos sobre física

Física em jogos se refere a uma simulação controlada pela própria engine.

No Unity, existem duas engines de física disponíveis, uma 2D, baseada na Box2D, e outra em 3D, chamada PhysX.

Collider

Componentes que marcam o volume físico de um objeto, assim como o material físico (com atrito e elasticidade) que define parte de seu comportamento.

Existem vários tipos de colisores, cada um com formato diferente (por exemplo, BoxCollider, SphereCollider).

Também mantem informações sobre o tipo de interação que tem com outros objetos (colisão ou sobreposição). Esse componente tem métodos e eventos que utilizamos para criar lógica para interações físicas, como OnCollisionEnter e OnCollisionExit. Para que um script possa utilizar essas funções-evento, o objeto a que está atrelado tem que ter um Collider.

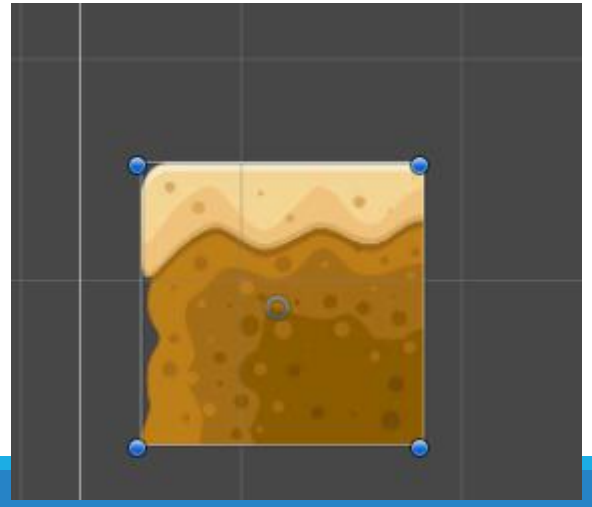
Colisão de elementos

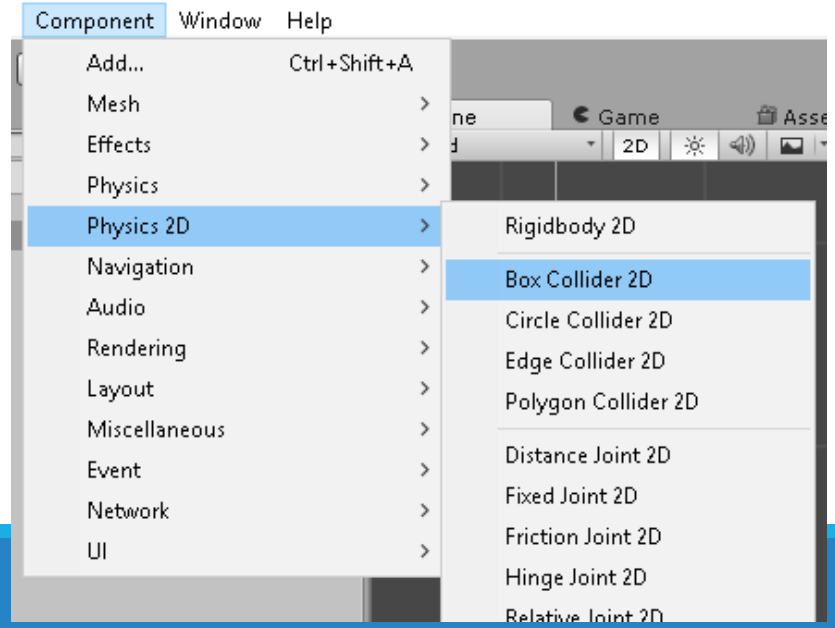
Acrescente o chão do cenário.

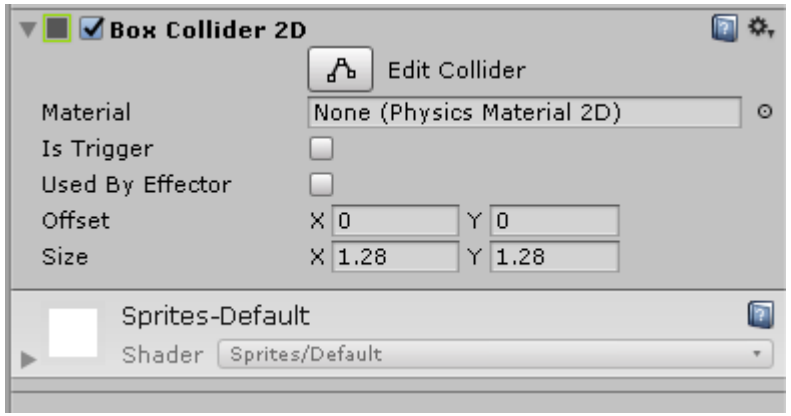
Clique sobre o componente.

Vá ao menu Component >> Physic2D>>Box Collider 2D

Faça para todos os elementos onde pode ocorrer colisão.







Colisão de elementos

Quando é adicionada a caixa de colisão a um elemento de cena, a caixa de colisão torna-se uma propriedade do elemento.

Além das propriedades de **posicionamento** e **cor** agora o elemento possui propriedades de colisão.

É possível configurar o tamanho da caixa de colisão (**hitbox**) clicando em edit Collider como pode ser viso na figura ao lado.

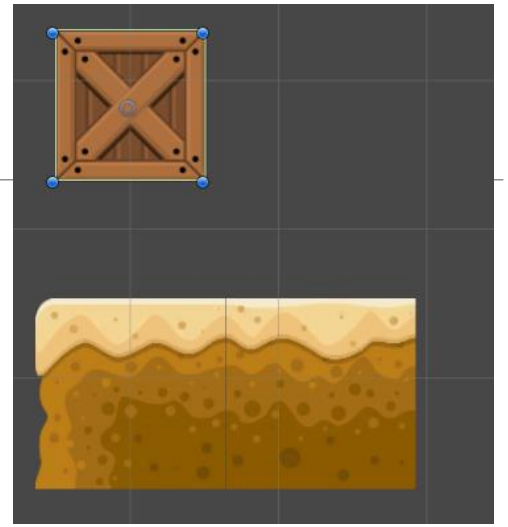
Corpos rígidos

O principal tipo de física usado em jogos é a física de corpos rígidos, que considera que objetos não deformam ou alteram seu volume e material quando afetados por interações como colisões.

RigidBody2D

- Componentes que marcam um objeto como parte da simulação física. Configura propriedades como massa, velocidade e arrasto, e permite o controle de como esse objeto se movimenta e gira em resposta a forças.
- O componente também guarda métodos para a aplicação de forças sobre o objeto.

RigidBody 2D



Acrescento o caixote

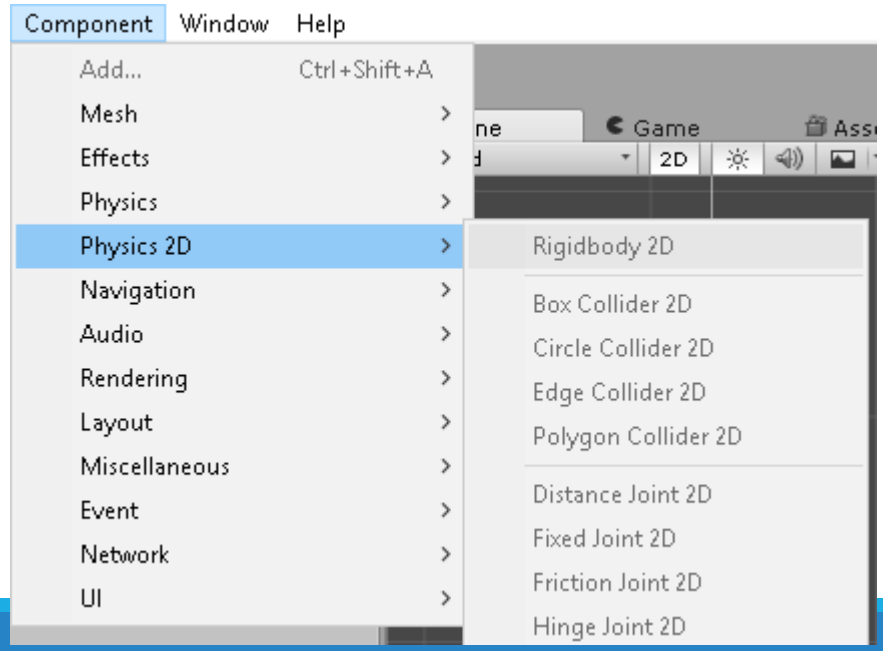
Defina o como caixa de colisão 2D

- Vá ao menu Component >> Physic2D>>Box Collider 2D

Acrescente Física ao de corpo Rígido

- Vá ao menu menu **Component** >> **Physic2D**>>**RigidBody 2D**

Como a caixa é um corpo rígido ao rodar o jogo ela vai cair se se colidir com o chão que por sinal é uma caixa de colisão.



Faça suposições

o que acontece se o cawboy e o chão forem configurados com colisor?

o que acontece se o cawboy for configurado como corpo rígido e o chão for configurado como colisor?

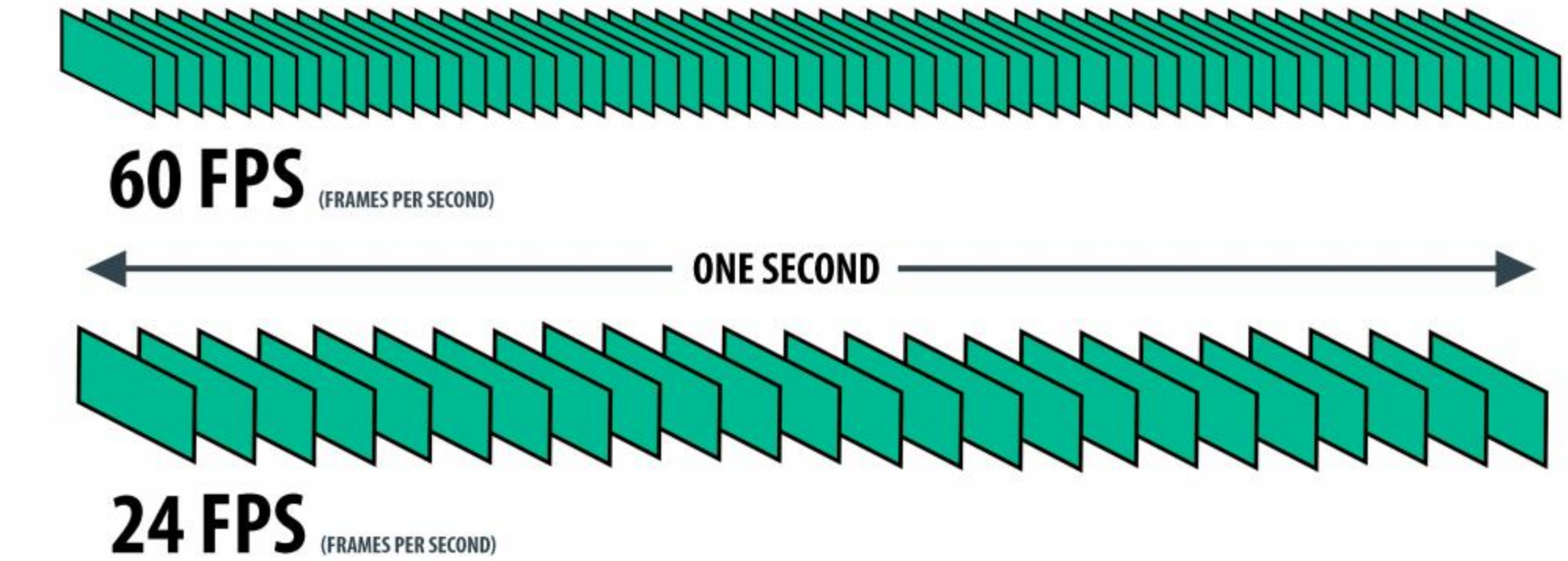
O que acontece se o chão for configurado como corpoRigido?

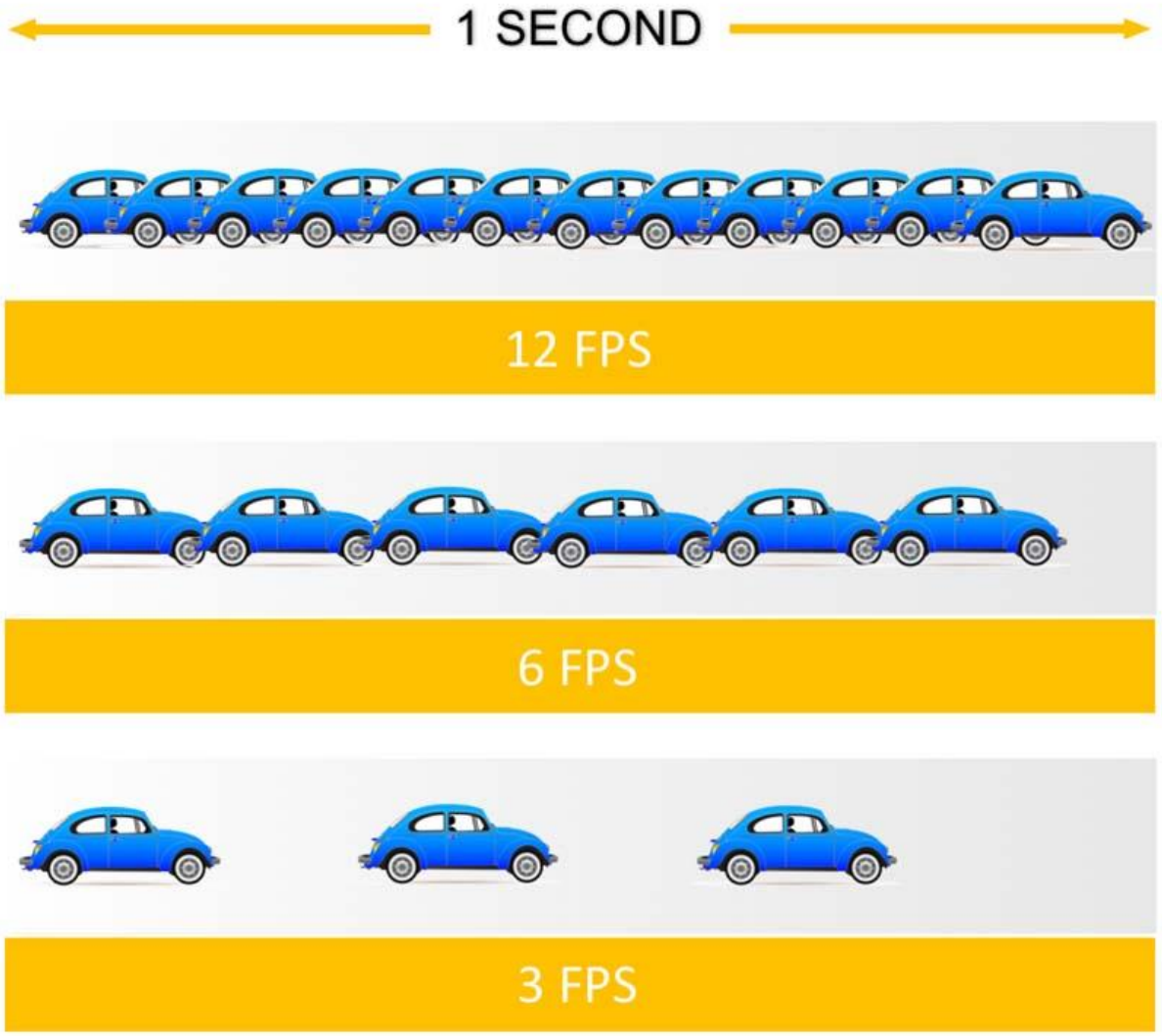
O que é FPS(frame per second)?

O FPS é a sigla que determina a quantidade quadros por segundo em um filme, uma animação ou uma cena de um jogo.

Quanto mais quadros, mais fluída será a animação e mais detalhado será o movimento

O cérebro humano de transformar imagens em sequência em um “filme” dentro de nossas cabeças.


60 FPS (FRAMES PER SECOND)
ONE SECOND
24 FPS (FRAMES PER SECOND)


1 SECOND
12 FPS
6 FPS
3 FPS

Script na computação

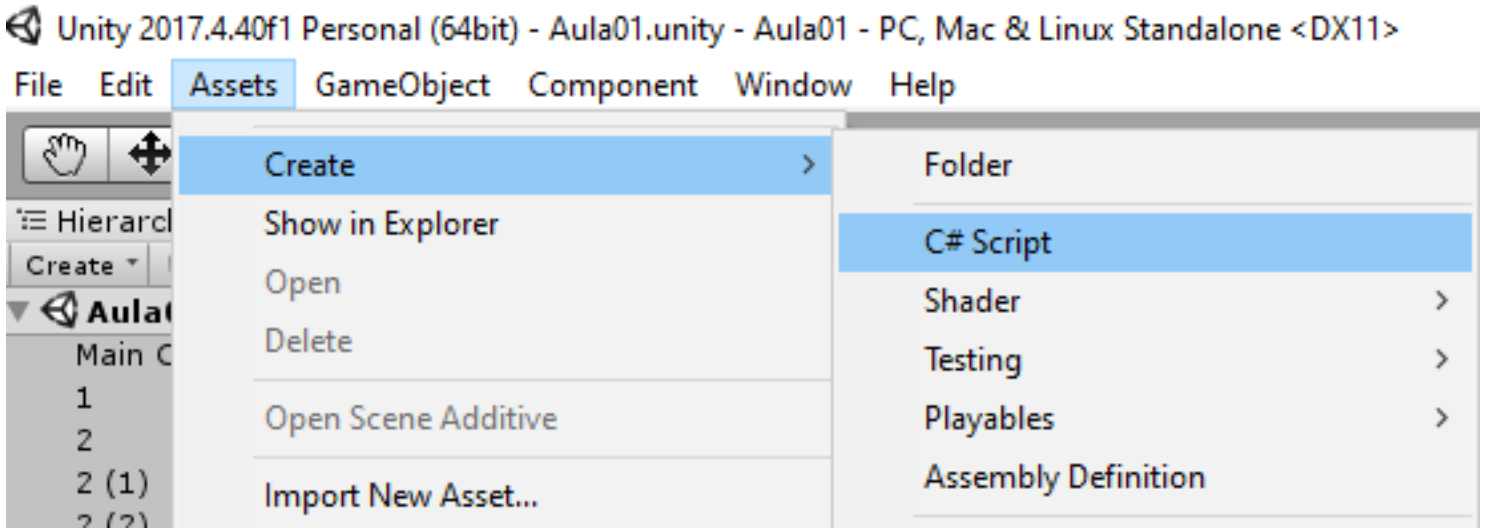
Script é um texto com uma série de instruções escritas para serem seguidas por um computador.

O termo é uma redução da palavra inglesa manuscript, que significa “manuscrito”, “escrito à mão”.

Criar um Script para o jogo

Click no meu “Assets”>>”C# Script”

Mude o nome do arquivo criado para : **Jogador** e efetue dois cliques para editar o script



Entenda o Script

Nome do Arquivo

Características de comportamento individual.

```
using UnityEngine;
using System.Collections;

public class MovimentoJogador : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Executa somente quando O jogo é iniciado

Executa uma vez a cada frame

Atributos

Os atributos são características.

São utilizados para descrever alguma coisa.

Quais os atributos da sala?

Quais os atributos da mesa?

Quais os atributos de um personagem de jogo?

Atributos do Personagem

Os atributos neste momento do curso podem ser considerados variáveis que devem ser utilizadas para representar características do personagem.

X: posição em x do personagem

Y: posição em Y do personagem

VelocidadeMovimento: Velocidade de movimentação do personagem.

DirecaoHorizontal: Se o personagem deve se mover para esquerda ou direita.

CorpoRigido: Variável que contem todas as funções de física do personagem.

Por que do personagem?

Porque o Script será posicionado e utilizado pelo personagem.

```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {
        float X;
        float Y;
        float VelocidadeMovimento;
        float DirecaoHorizontal;
        Rigidbody2D CorpoRigido;

    void Start () {

    }

    void Update () {

    }

}
```

Inicialização de Variáveis

```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {
        float X;
        float Y;
        float VelocidadeMovimento;
        float DirecaoHorizontal;
        Rigidbody2D CorpoRigido;
        void Start () {
                VelocidadeMovimento = 5;
                DirecaoHorizontal = 0;
                CorpoRigido = GetComponent<Rigidbody2D> ();
                CorpoRigido.freezeRotation = true;
        }
        void Update () {


        }
}
```

O método **void Start()** deve ser utilizado para iniciar variáveis e propriedades do Personagem.

Nesse momento o Corpo Rígido que foi adicionado ao personagem (Component>>Physic2D>>RigidBody2D) é inserido na variável **CorpoRigido**, assim é possível modificar propriedades de física utilizando a variável CorpoRigido.

Observe que a propriedade freezeRotation é modificada para true, isso impede que o personagem rotacione quanto tocar em quinas.

Criar movimento horizontal

```
void Update () {
    DirecaoHorizontal = Input.GetAxis ("Horizontal");
    X = VelocidadeMovimento * DirecaoHorizontal;
    Y = CorpoRigido.velocity.y;
    Vector2 vetorMovimento = new Vector2 (X, Y);
    CorpoRigido.velocity = vetorMovimento;
}
```

DirecaoHorizontal Recebe até 1 para direita e até -1 para esquerda.
Quando parado Recebe 0.

= 5 * DirecaoHorizontal.

Determina o módulo, sentido e direção do movimento

Código Completo

```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {
        float X; //Posição em X do Personagem
        float Y; //Posição em Y do Personagem
        float VelocidadeMovimento;  //Velocidade de movimento do personagem
        float DirecaoHorizontal;    // direção que o personagem deve ser mover (-1,0,1)
        Rigidbody2D CorpoRigido;
        void Start () { //INICIALIZA VARIÁVEIS E ATRIBUTOS
                VelocidadeMovimento = 5;   //Inicializa a variável VelocidadeMovimento
                DirecaoHorizontal = 0;    //Inicializa a variável DirecaoHorizontal
                // Inicializa a variável com o corpo rígido do personagem.
                // É necessário que o corpo rígido tenha sido inserido no personagem.
                CorpoRigido = GetComponent<Rigidbody2D> ();
                CorpoRigido.freezeRotation = true; //Impede que o personagem rotacione no eixo.
        }
        void Update () { // É EXECUTADO UMA VEZ POR FRAME
                //DirecaoHorizontal Recebe até 1 para direita e  até -1 para esquerda. Quando parado Recebe 0.
                DirecaoHorizontal = Input.GetAxis ("Horizontal");
                X = VelocidadeMovimento * DirecaoHorizontal; // Gera uma nova posição em x
                Y = CorpoRigido.velocity.y; //Recupera a velocidade em y que já existe
                Vector2 vetorMovimento = new Vector2 (X, Y); //Cria um vetor de velocidade
                // O vetor de velocidade é adicionado a velocidade do corpo rígido do personagem
                CorpoRigido.velocity = vetorMovimento;
        }
}
```

Melhorando o código - Organização

Conforme o andar do desenvolvimento de um jogo é possível verificar que o método Update ganha mais linhas de código conforme o Personagem ganha funcionalidade.

É possível que ao final do desenvolvimento o jogo possua centenas de linhas de código ou mais.

Para melhor organizar o que é programado em **void Update()** você pode criar blocos de código separados, onde cada bloco possuí uma funcionalidade diferente.

Melhorando o código - Organização

```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {
		float X;
		float Y;
		float VelocidadeMovimento;
		float DirecaoHorizontal;
		Rigidbody2D CorpoRigido;
		void Start () {
				VelocidadeMovimento = 5;
				DirecaoHorizontal = 0;
				CorpoRigido = GetComponent<Rigidbody2D> ();
				CorpoRigido.freezeRotation = true;
		}
		void Update () {
				//chama a todo frame O código dentro de MovimentoHorizontal()
				MovimentoHorizontal();
		}
		void MovimentoHorizontal(){
				DirecaoHorizontal = Input.GetAxis ("Horizontal");
				X = VelocidadeMovimento * DirecaoHorizontal;
				Y = CorpoRigido.velocity.y;
				Vector2 vetorMovimento = new Vector2 (X, Y);
				CorpoRigido.velocity = vetorMovimento;
		}
}
```

Nesse momento do curso trataremos como um bloco de código que pode ser chamado quando necessário