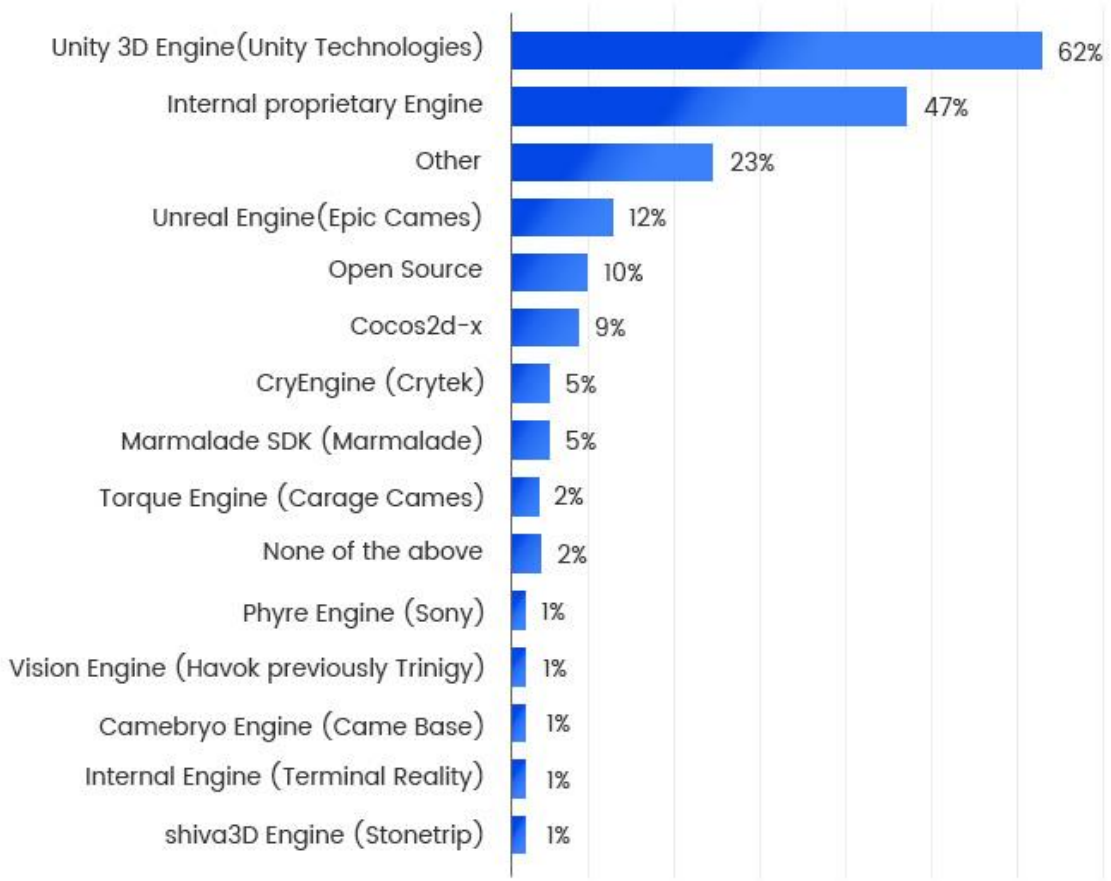
Introdução ao Unity3d, conceitos fundamentais de física e *Game Engine*

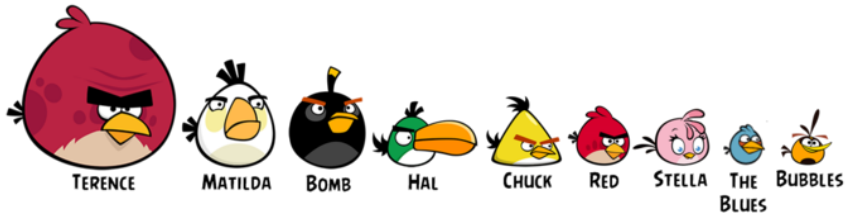
Prof. Me. Hélio Esperidião

Unity

- Fatia de mercado mundial para o desenvolvimento de jogos.
- 71% dos 1.000 jogos mobile mais distribuídos são feitos com Unity.



Unity	Unreal Engine
Mobile games made with Unity account for 71% of the top 1000 titles on the market	Over 2 million games powered by the engine
The user base (developers + users) amounts to over 2.7 billion people	Marketplace features over 16K assets
Asset Store features over 65K assets	Developer community exceeds 7 million people
Revenue increased by 43% in 2020	Market share is estimated at 13%
Market capitalization is approximately $27.8 billion.	Epic Games' (parent company) market cap is approximately $17 billion.



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

Mais jogos


AND THE BLIND


CITIES
SKYLINES

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

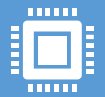
Geralmente possuem recursos para criação de funções gráficas, física aos objetos, trilhas sonoras, entre outras ações.

Game engine

Incorpora um motor gráfico para processar gráficos em 2D ou 3D

Incorpora motor de física para identificar colisões e realizar animações, além de oferecer suporte para áudio, inteligência artificial, gerenciamento de arquivos, programação, entre outras funcionalidades.

Isso implica que a criação de um jogo a partir do zero se torna mais acessível.



As engines de jogos possibilitam que uma pessoa, por conta própria, desenvolva um jogo que, nas décadas de 80, exigiria uma equipe completa.

Década de 80

- Na década de 80, os desenvolvedores de jogos para consoles populares como o Atari enfrentavam a árdua tarefa de codificar manualmente todos os elementos necessários para criar pixels, movimentos e efeitos físicos.
- Esse processo era extremamente trabalhoso e pouco intuitivo exigindo uma equipe considerável para criar os primeiros jogos em 2D.
- Devido às limitações de hardware, o código precisava ser o mais simples possível, e a manutenção da limpeza do código escrito à mão era um desafio adicional.
- Todo esse extenso trabalho tornava-se obsoleto rapidamente, pois a constante evolução do hardware exigia a reescrita de código para cada novo jogo.

Por que é o mais usado?

- Facilidade de aprendizado
- Muita documentação e fórum de discussões na internet
- Cria/portar games para várias plataformas
- Versão gratuita
- Crie jogos do Sistema Operacional que preferir
- Jogos em 2D e 3D

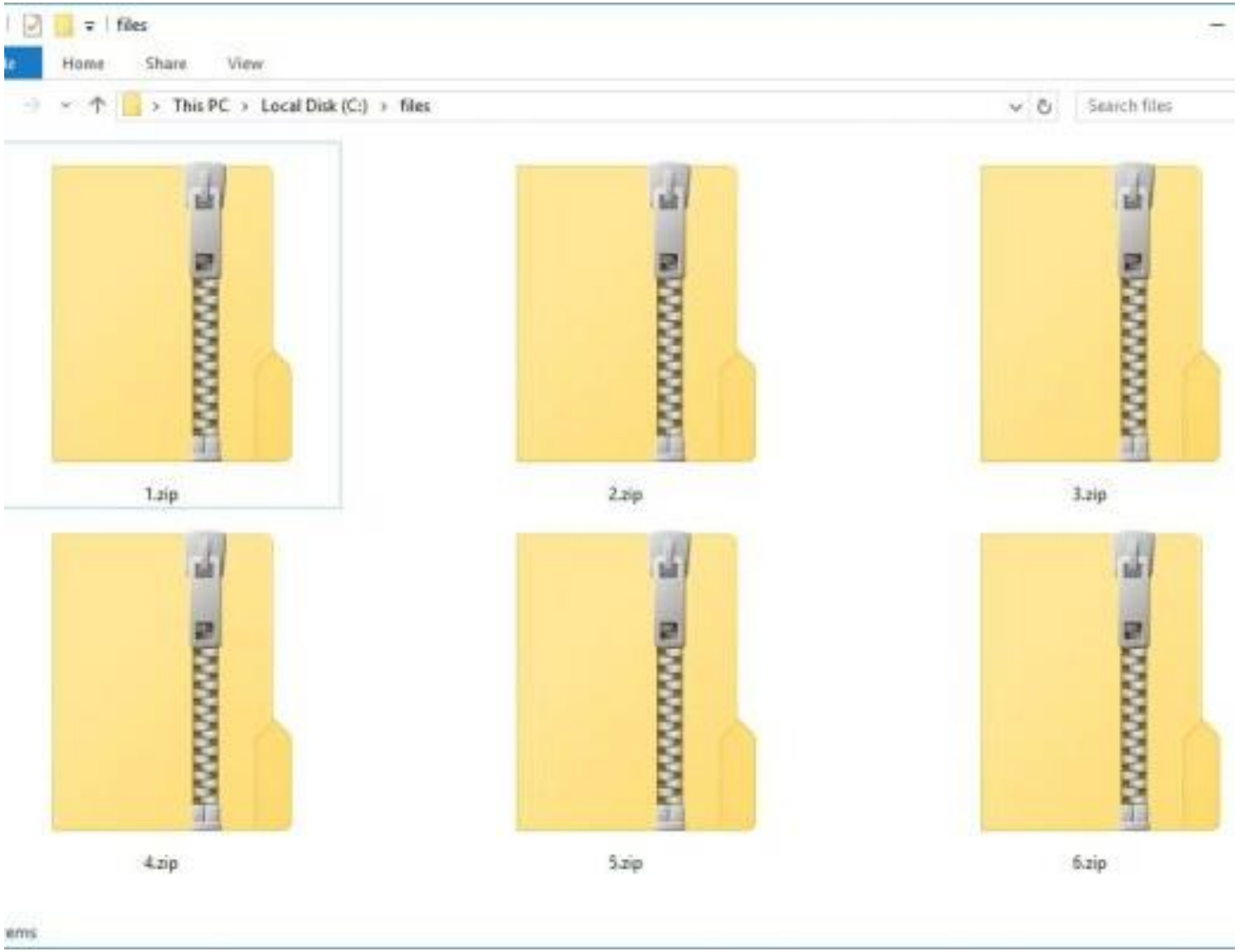
Assets

- **Asset significa: A**tivo.

- 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

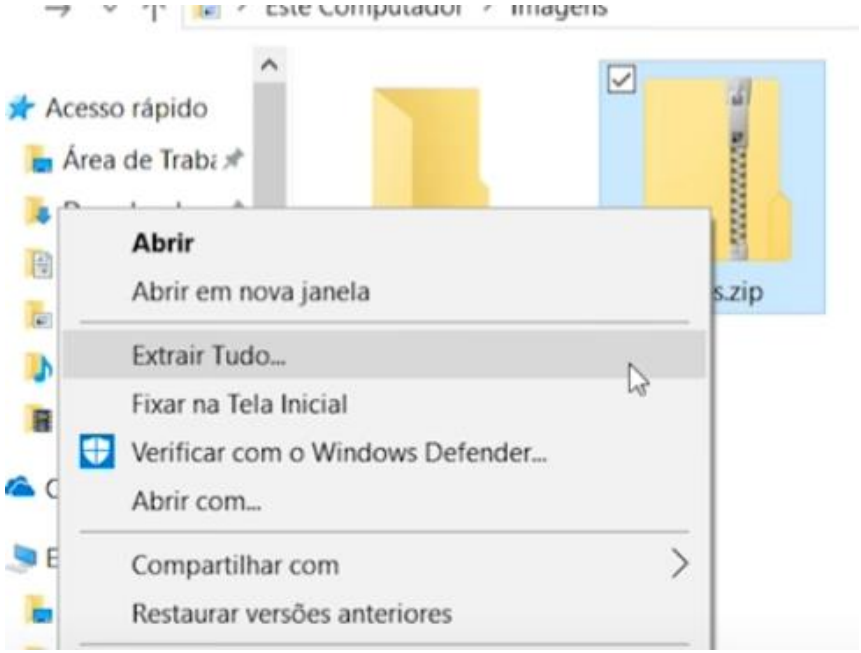


* Arquivos compactados (.zip)

- 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.

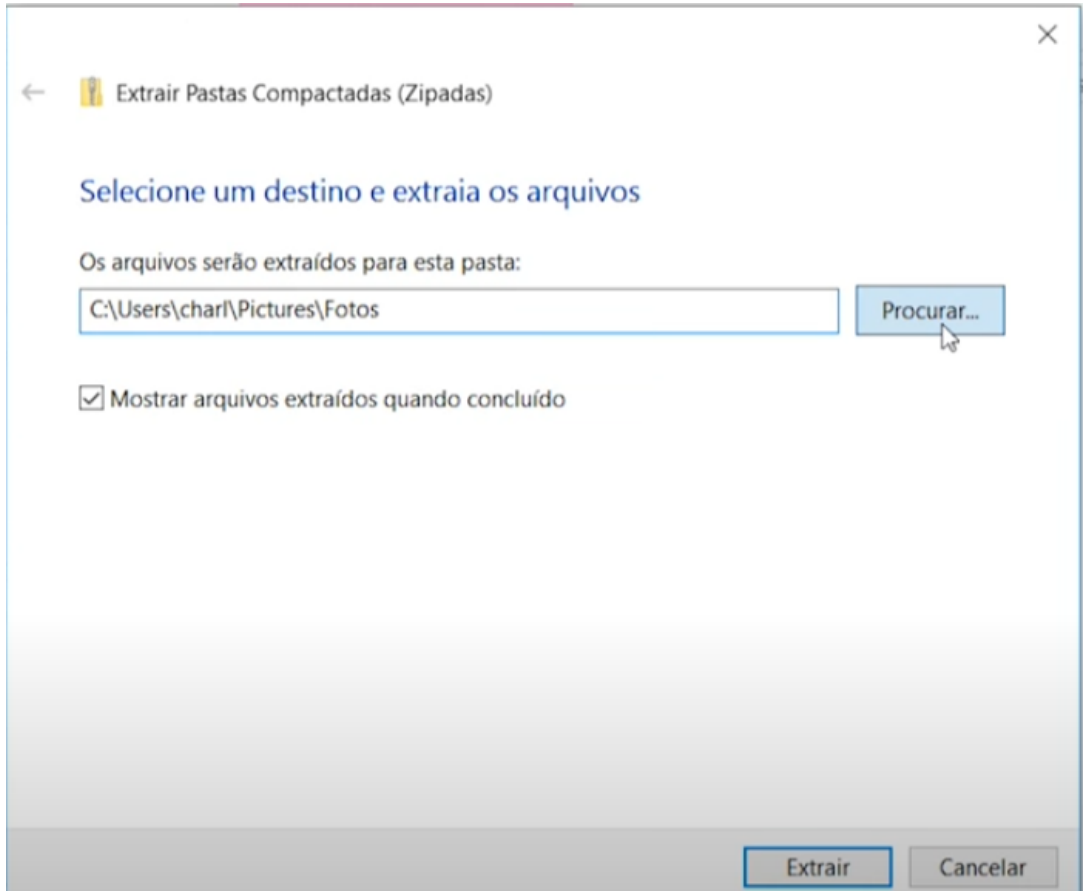
* Arquivos compactados (.zip)

- 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”

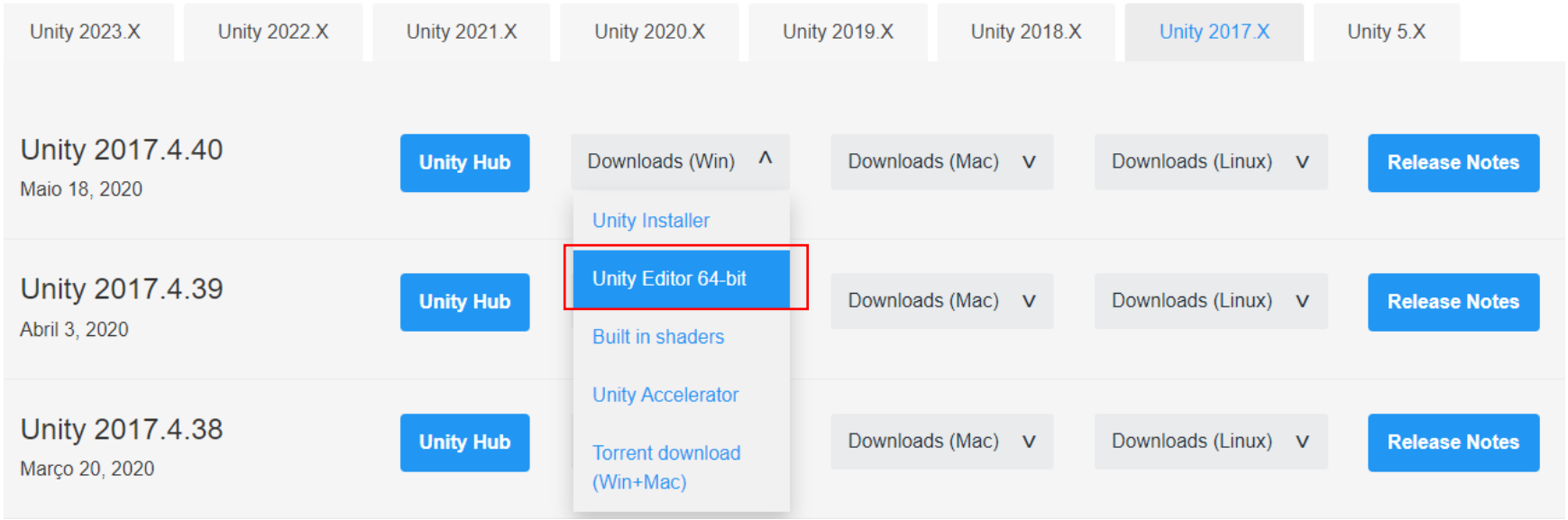


Instalação do unity

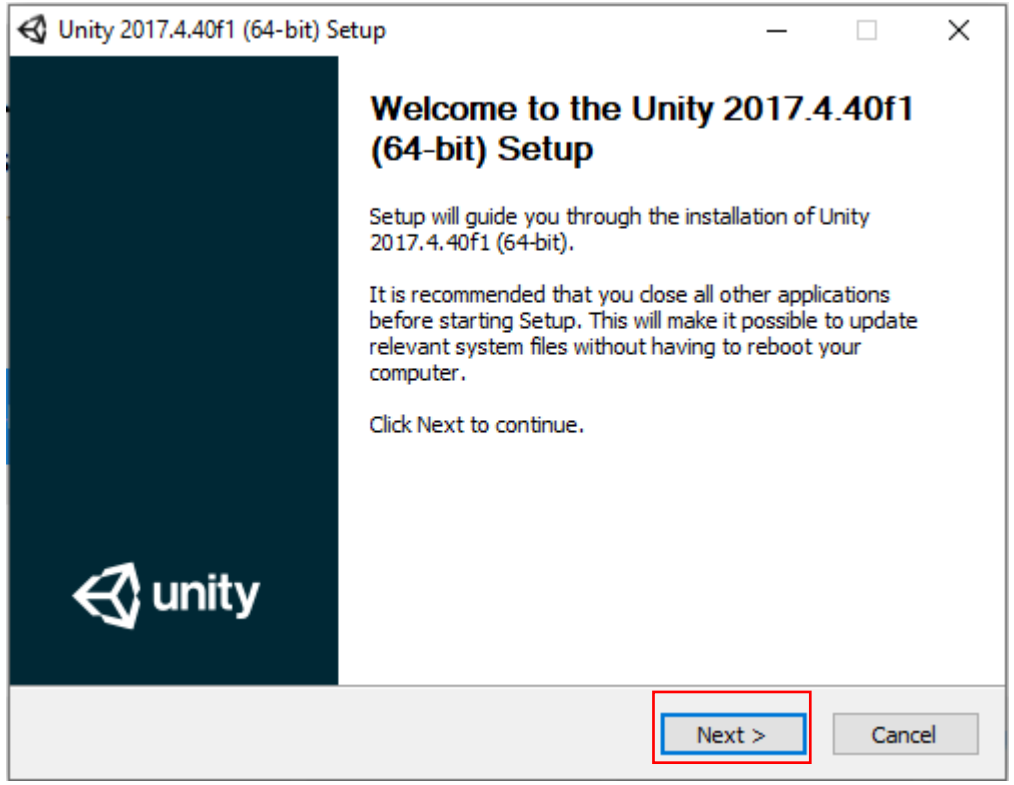
- Obrigatoriamente instale em sua casa a mesma versão que está instalada na escola:
  - **2017.4.40f1**
- Link para download: https://unity.com/releases/editor/archive
- Download direto: https://download.unity3d.com/download_unity/6e14067f8a9a/Windows64EditorInstaller/UnitySetup64-2017.4.40f1.exe

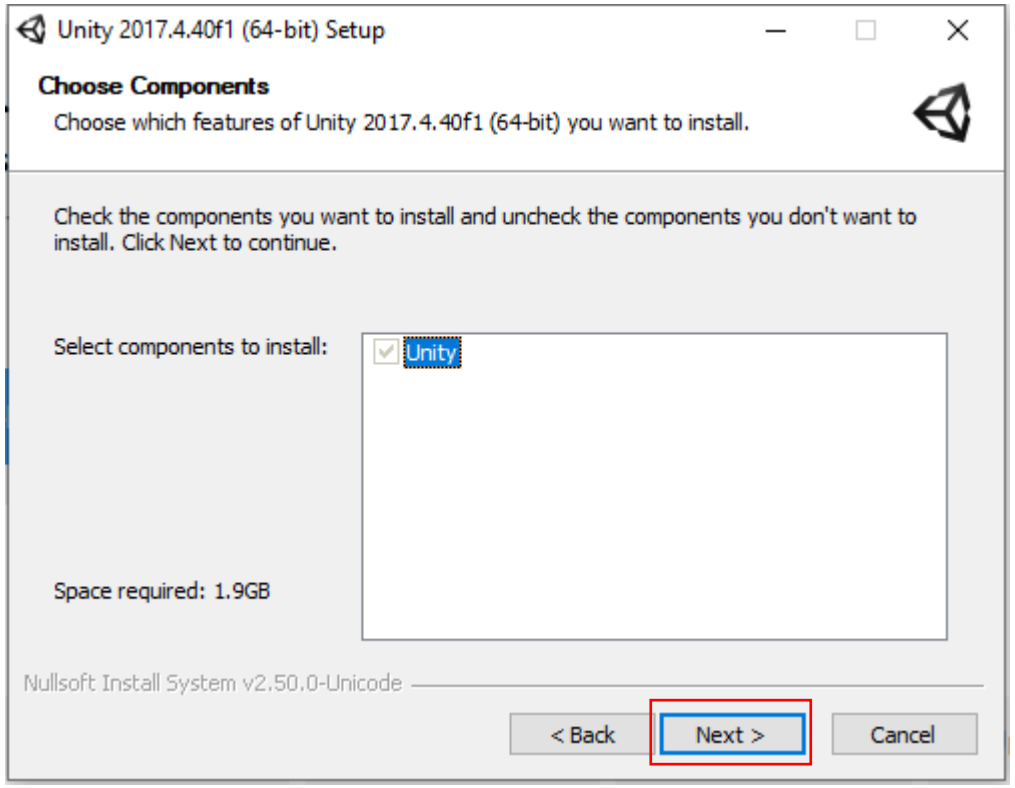
Versão do Unity: 2017.4.40f1

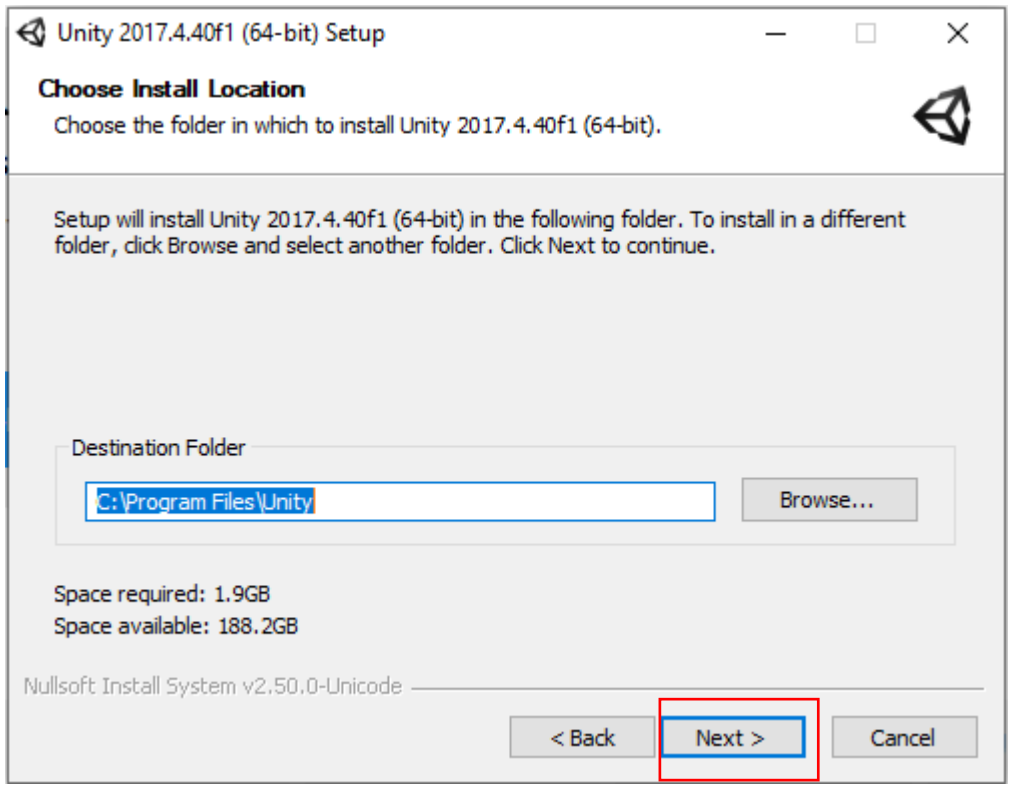
- É possível fazer o download diretamente do site
- Instale o **Unity Editor 64-bit**



Instalação Unity passo a passo



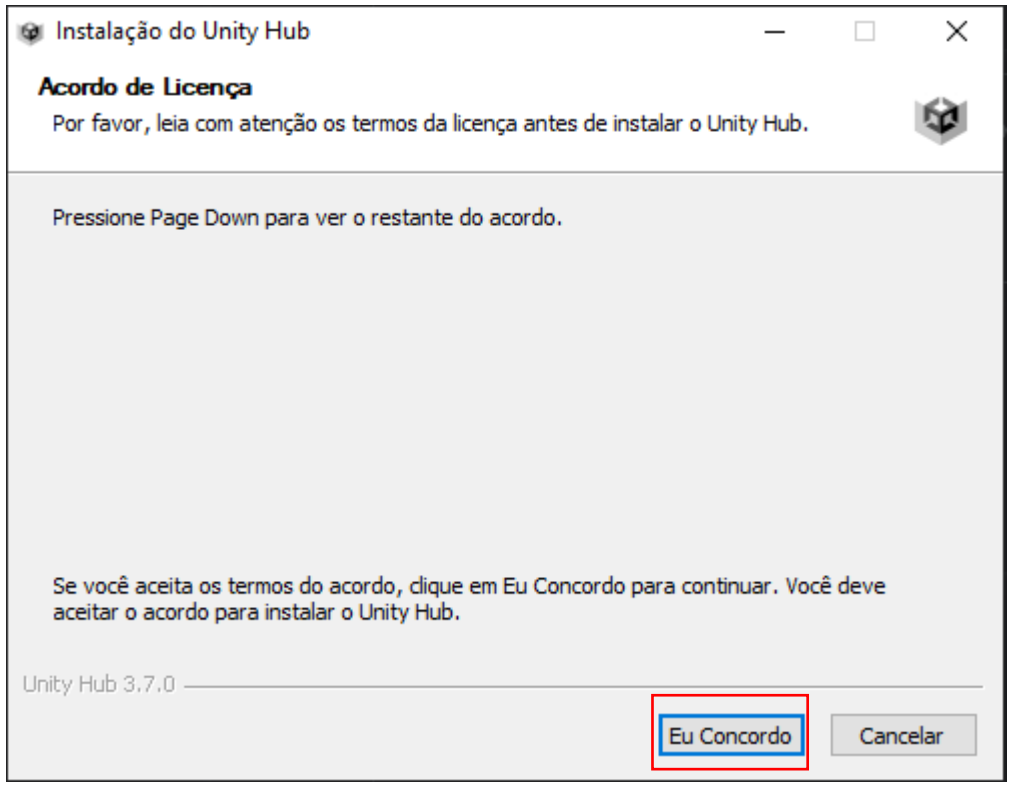

Unity 2017.4.40f1 (64-bit) Setup
Choose Components
Choose which features of Unity 2017.4.40f1 (64-bit) you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Unity
Space required: 1.9GB
Nullsoft Install System v2.50.0-Unicode
< Back
Next >
Cancel

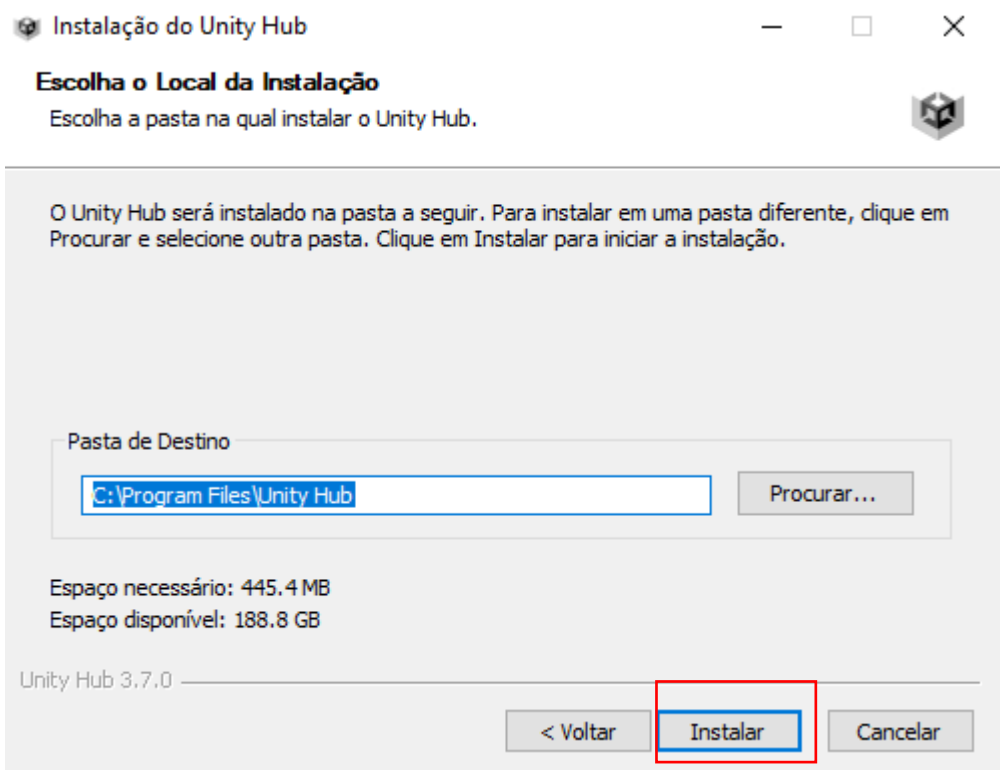

Unity 2017.4.40f1 (64-bit) Setup
Choose Install Location
Choose the folder in which to install Unity 2017.4.40f1 (64-bit).
Setup will install Unity 2017.4.40f1 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Program Files\Unity
Browse...
Space required: 1.9GB
Space available: 188.2GB
Nullsoft Install System v2.50.0-Unicode
< Back
Next >
Cancel

Instalar o Unity HUB(2024)

- Link para download: https://public-cdn.cloud.unity3d.com/hub/prod/UnityHubSetup.exe

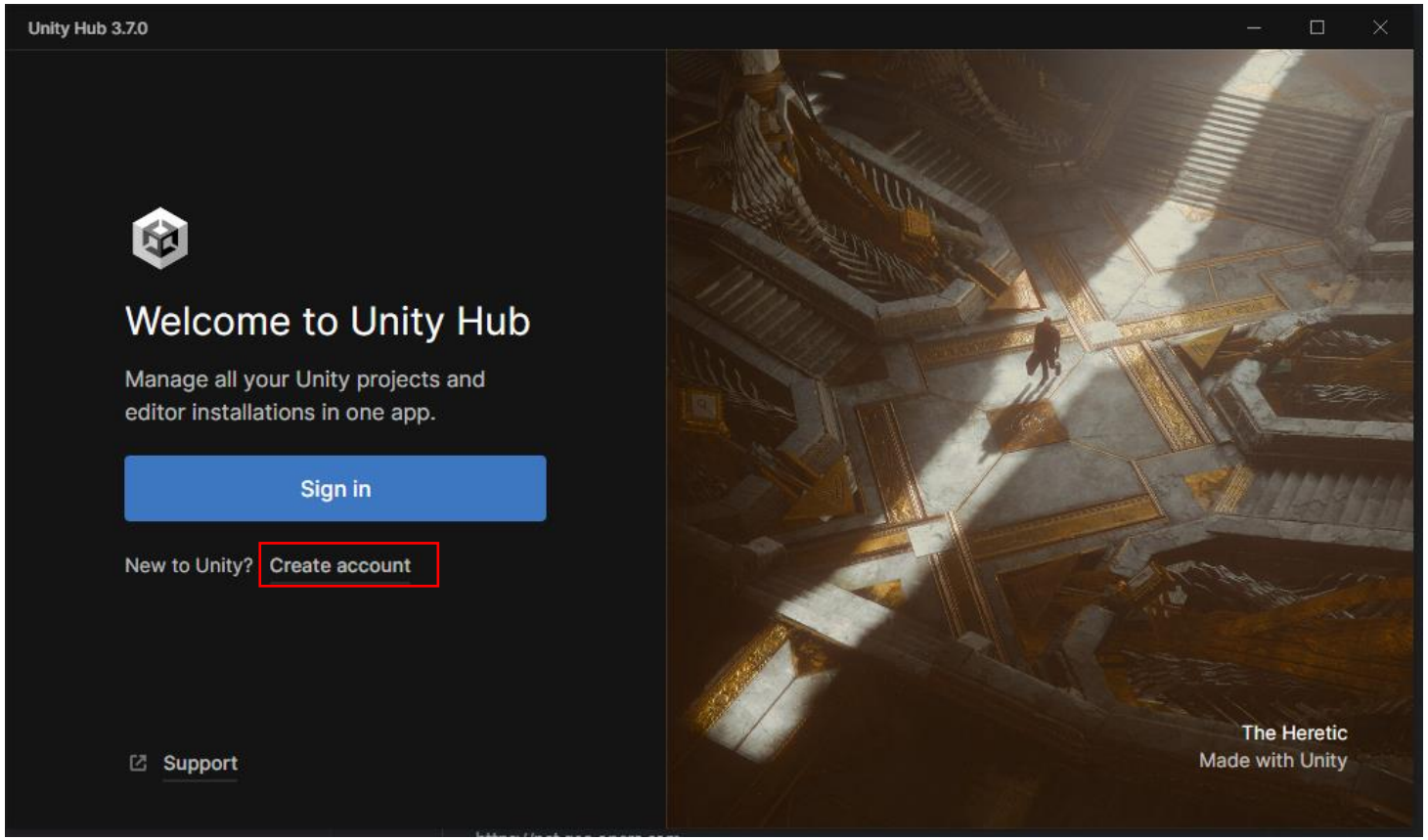
Instalação do Unity hub





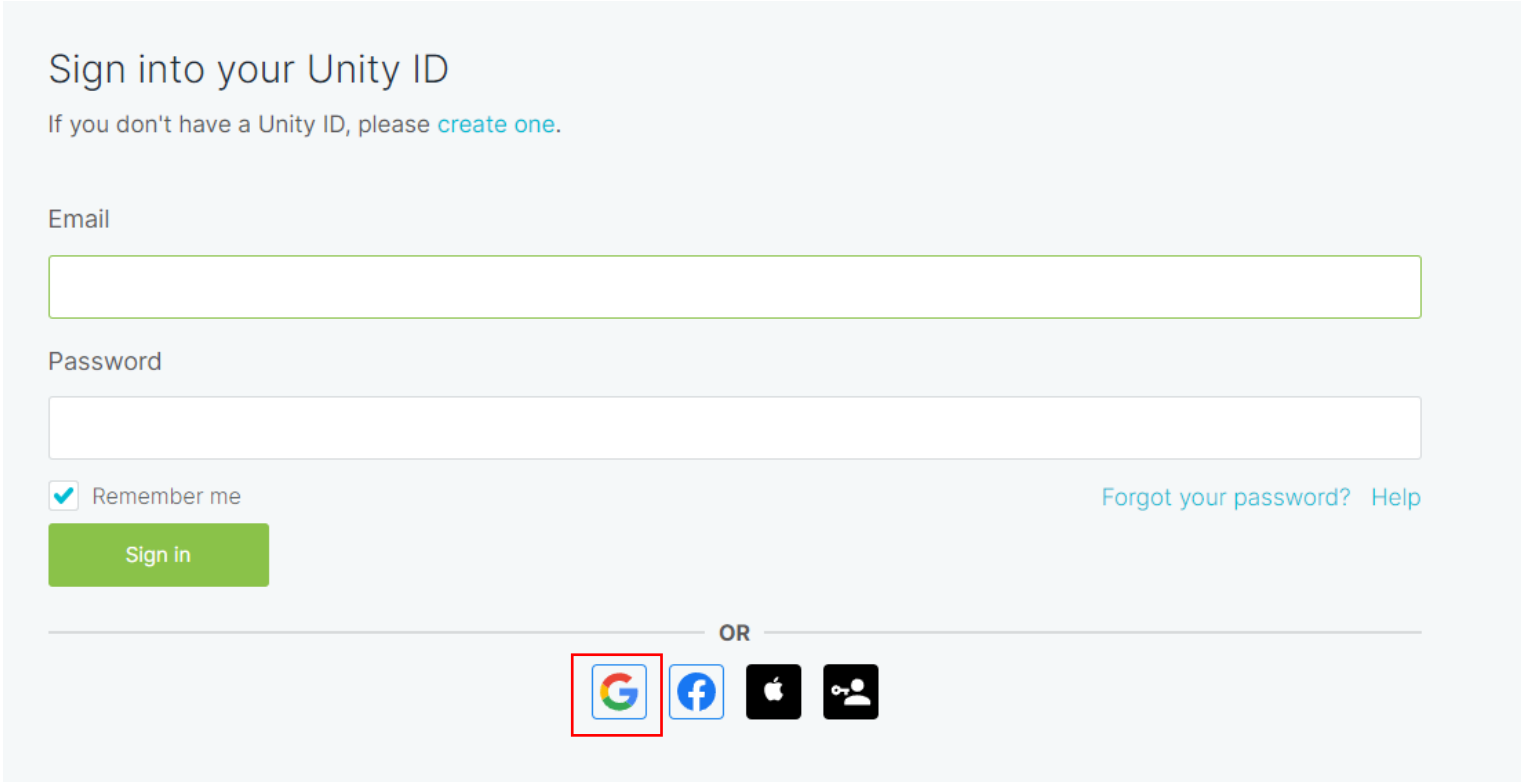
Faça login

- Procedimento necessário para executar o Unity NO SEU COMPUTADOR.
  - Faça login ou crie uma conta
- Os laboratórios da escola já estão configurados.

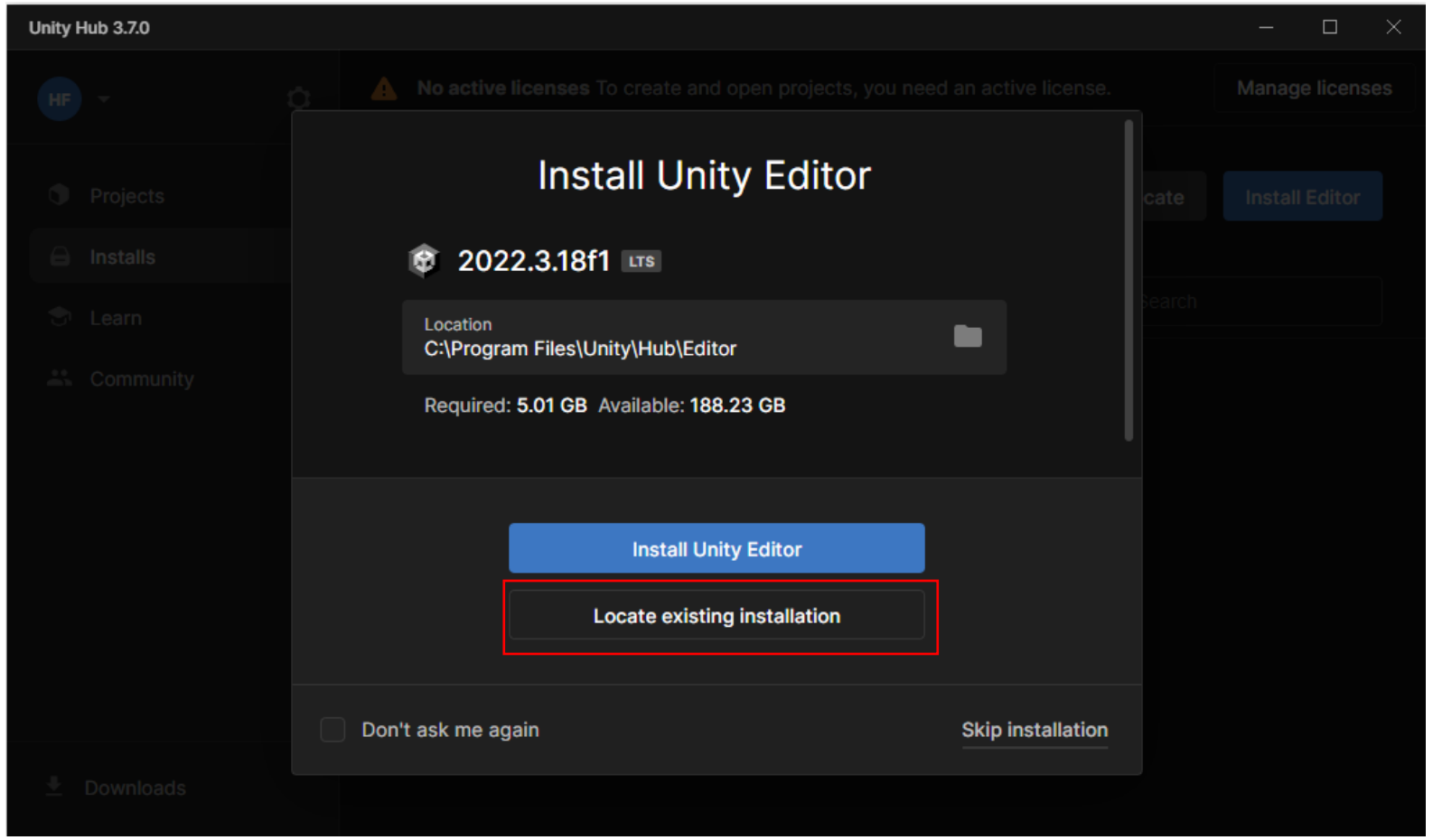


Só é necessário para instalar e utilizar o Unity no seu computador.

Faça login ou crie uma conta

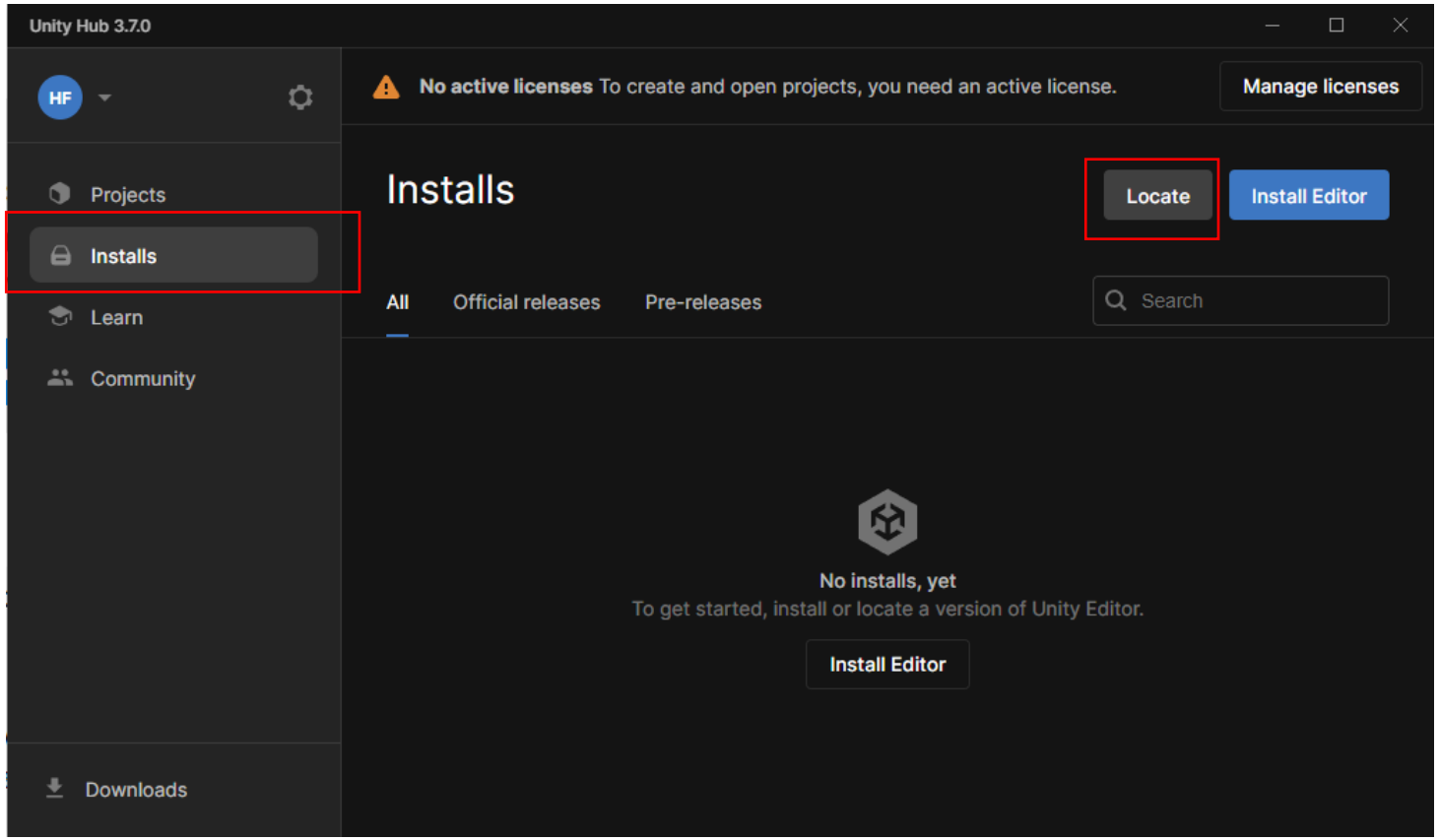


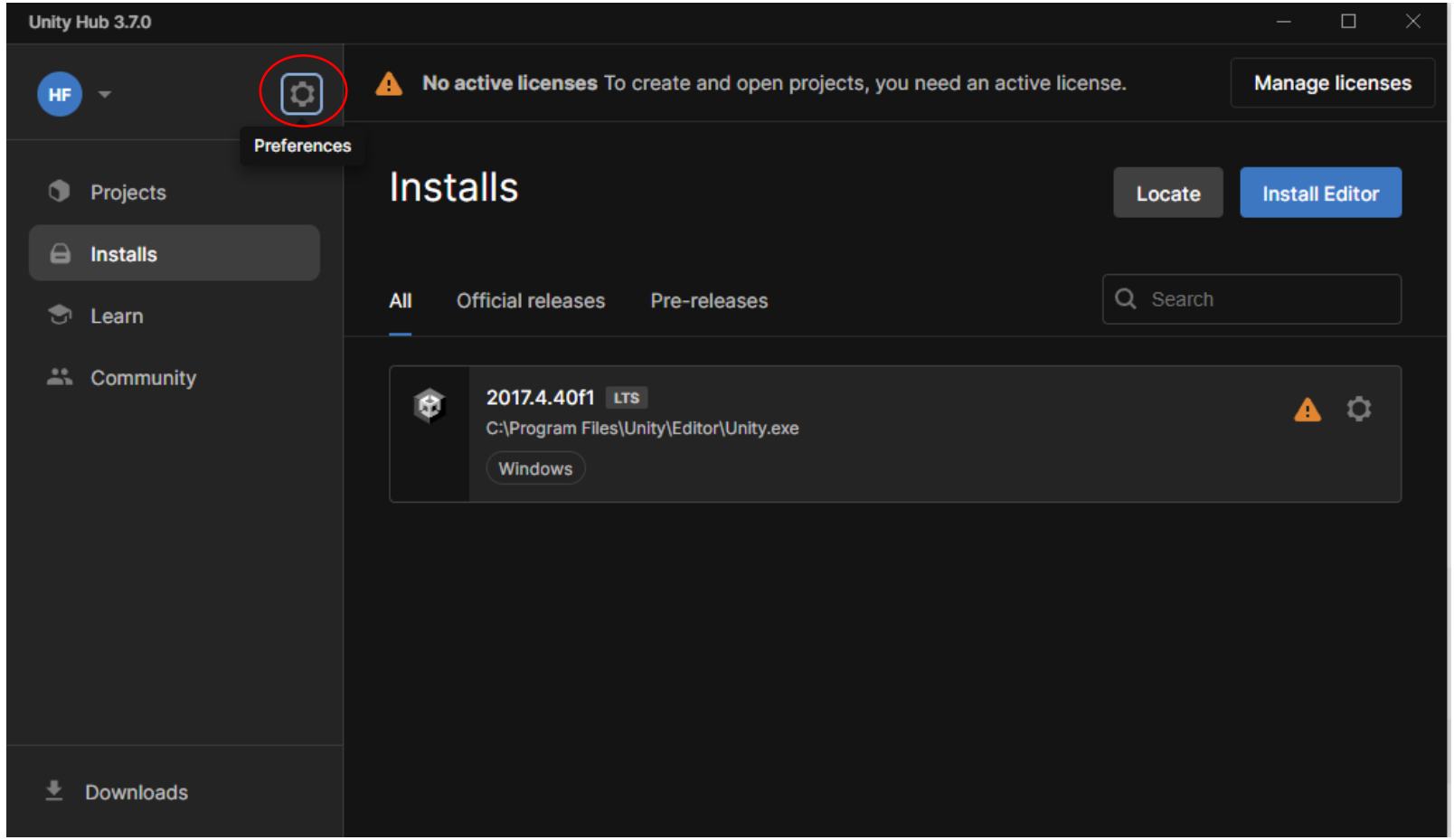
Locate existing installation



Versão Unity hub 2024

- Vá no menu installs.
- Click no botão Locate
  - Selecione o arquivo “C:\Program Files\Unity\Editor\unity.exe”



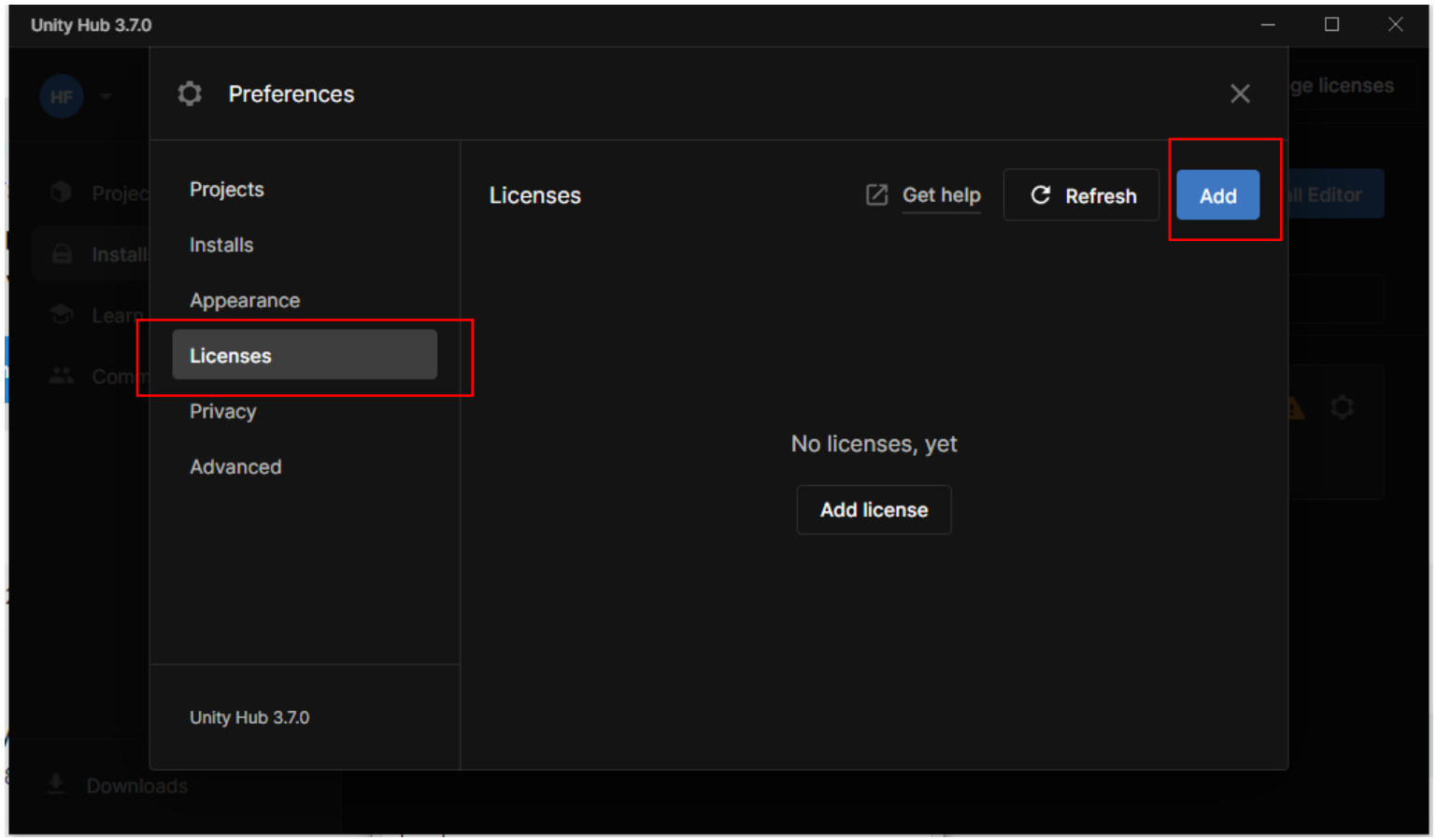


Ative uma licença do unity

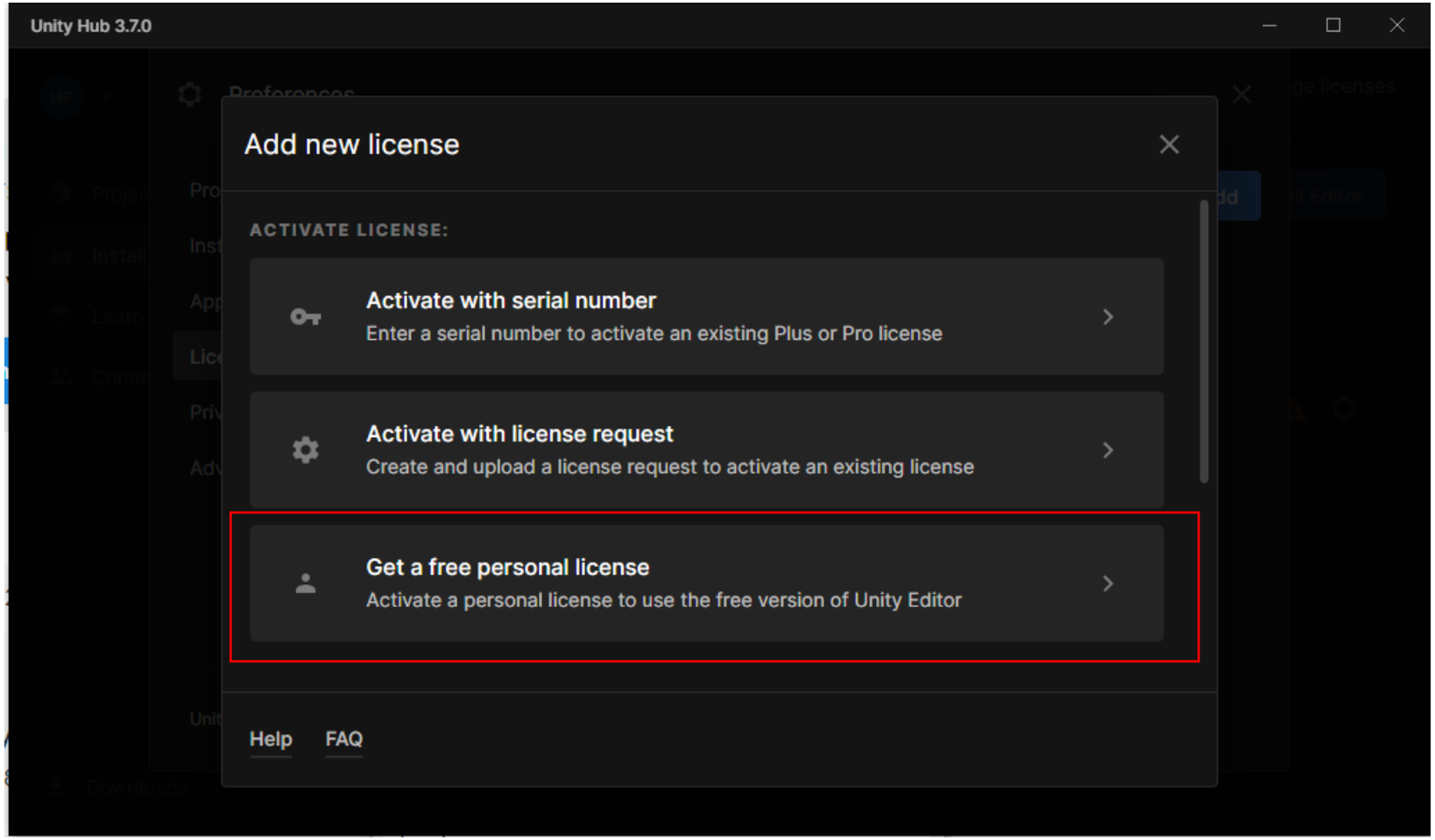
Clique na engrenagem

Escolha a opção “Licenses”

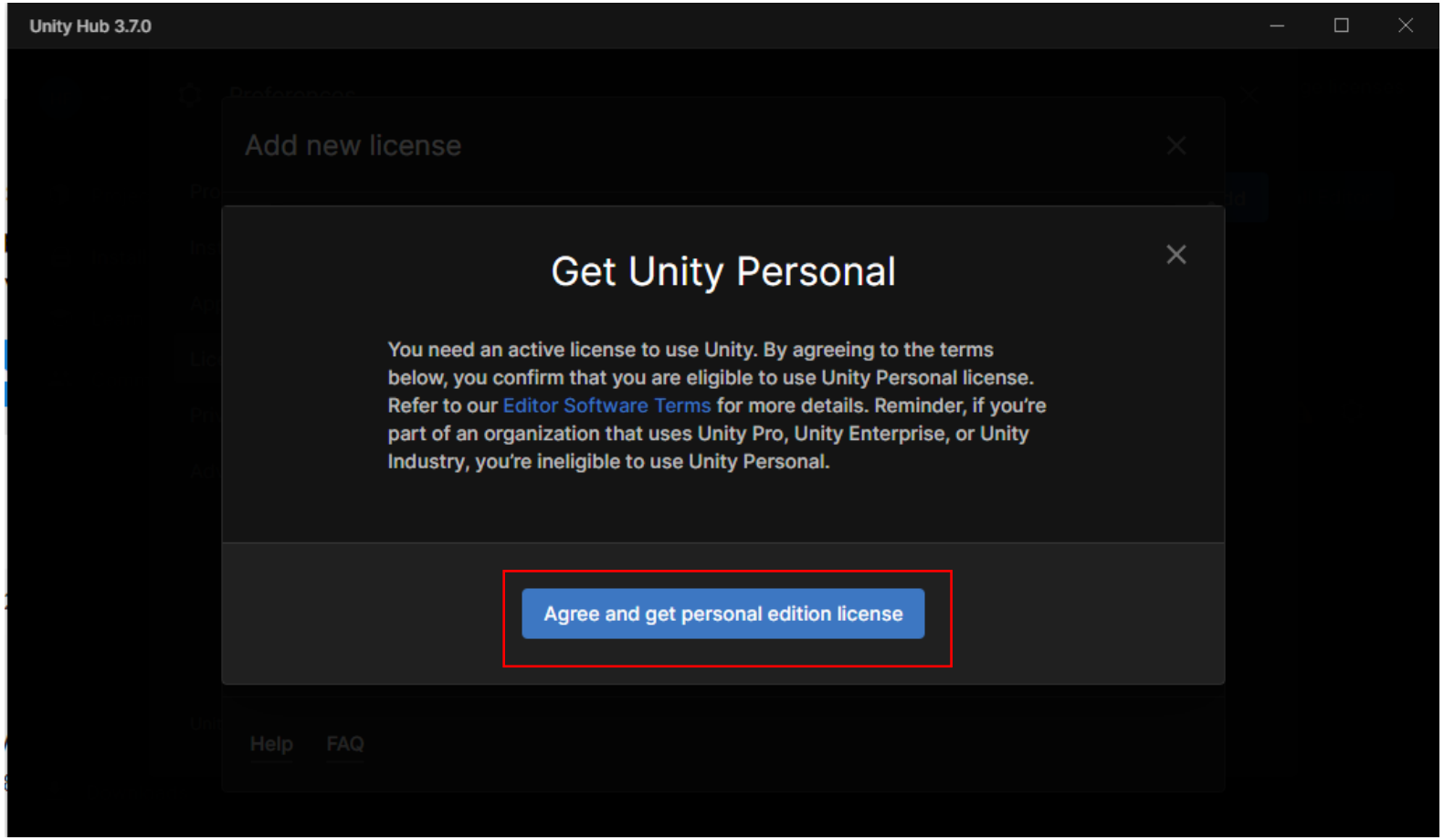
- Clique no botão “add”



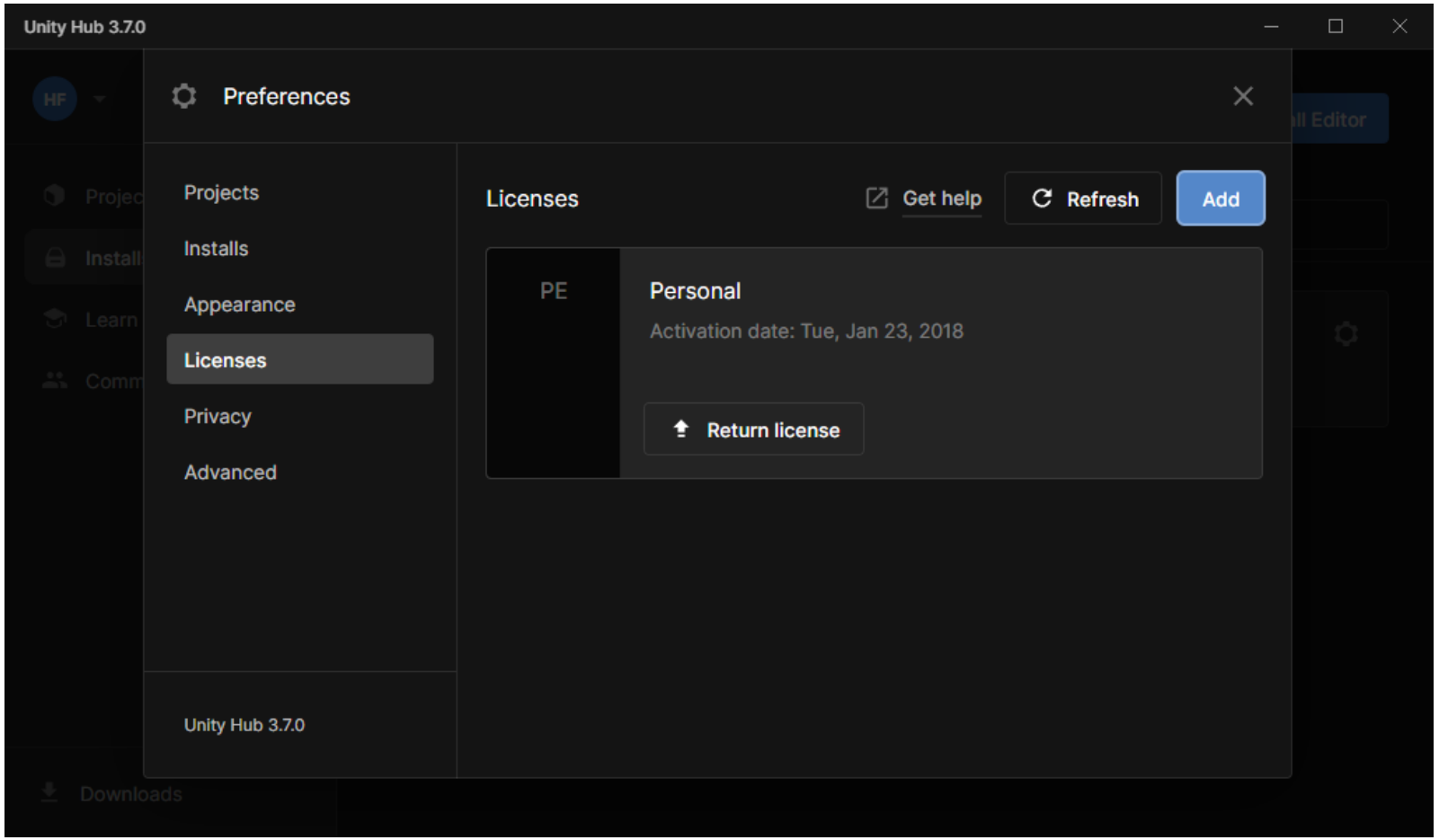
Escolha:
“Get a free personal license”



Aceite os termos de uso

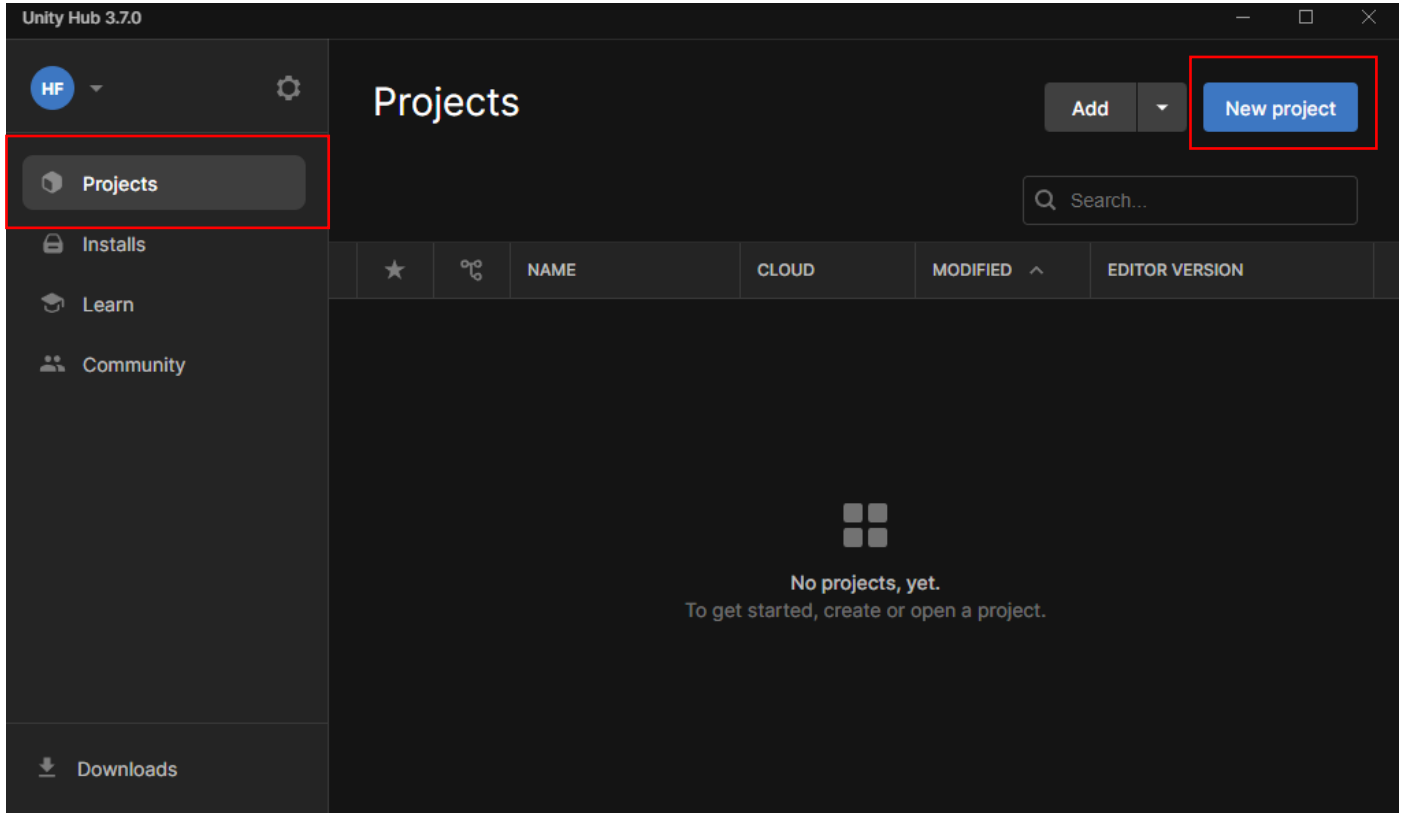


Licença ativada



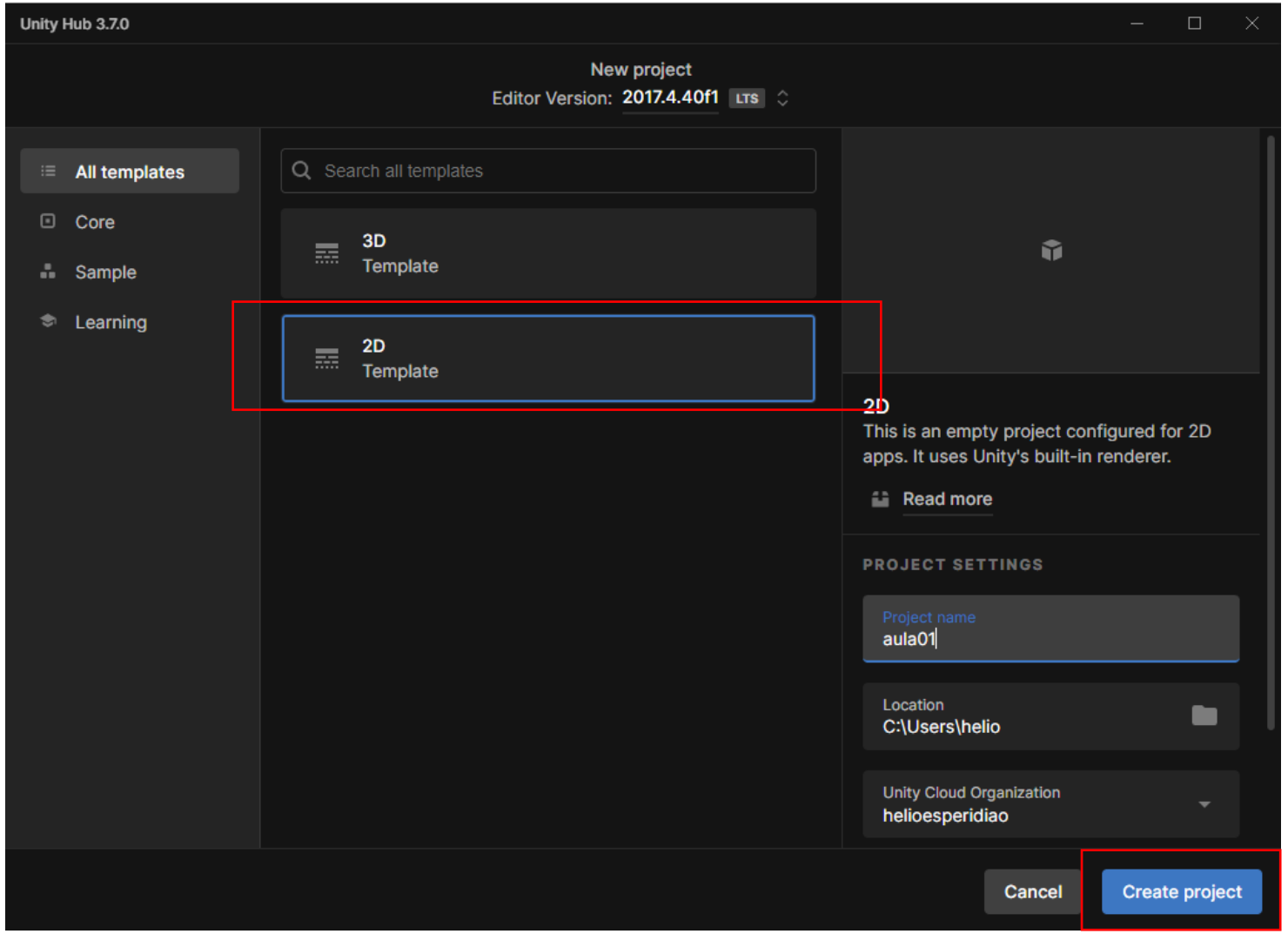
Criar um novo projeto

- Selecione “Projects”
- Clique no botão “new Project”



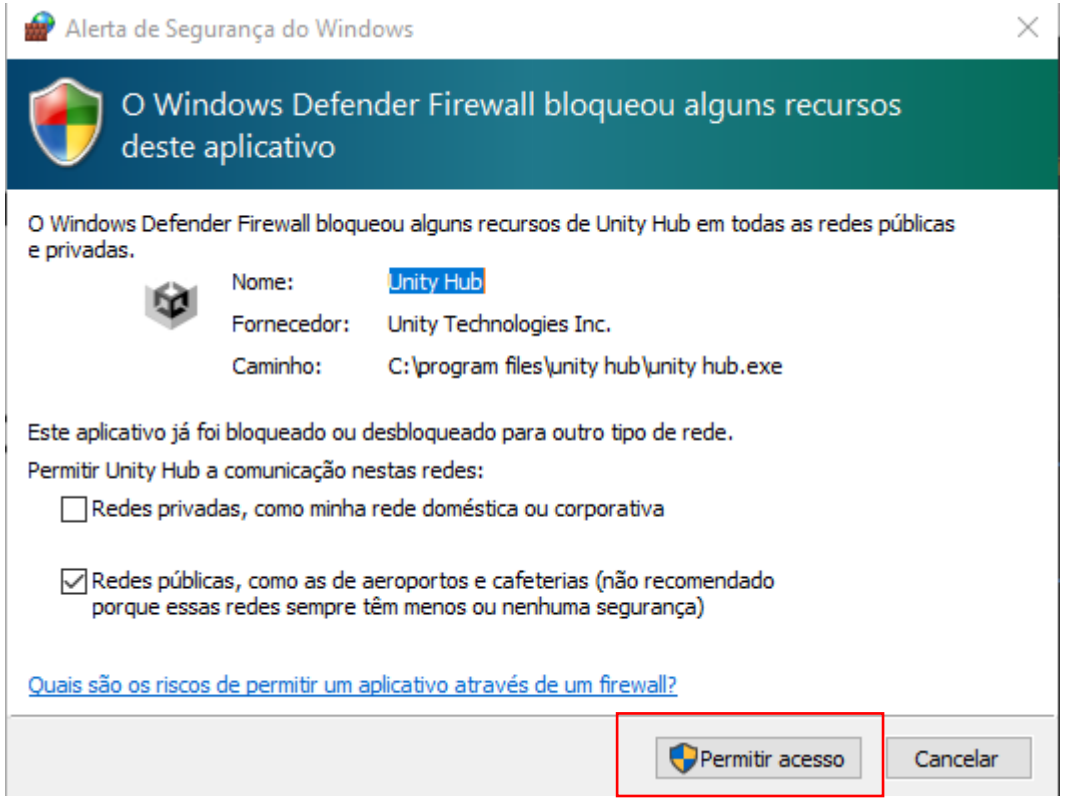
Defina o template 2d

- Escolha o nome para o projeto
- Escolha o local dos arquivos



Libere o acesso no firewall.

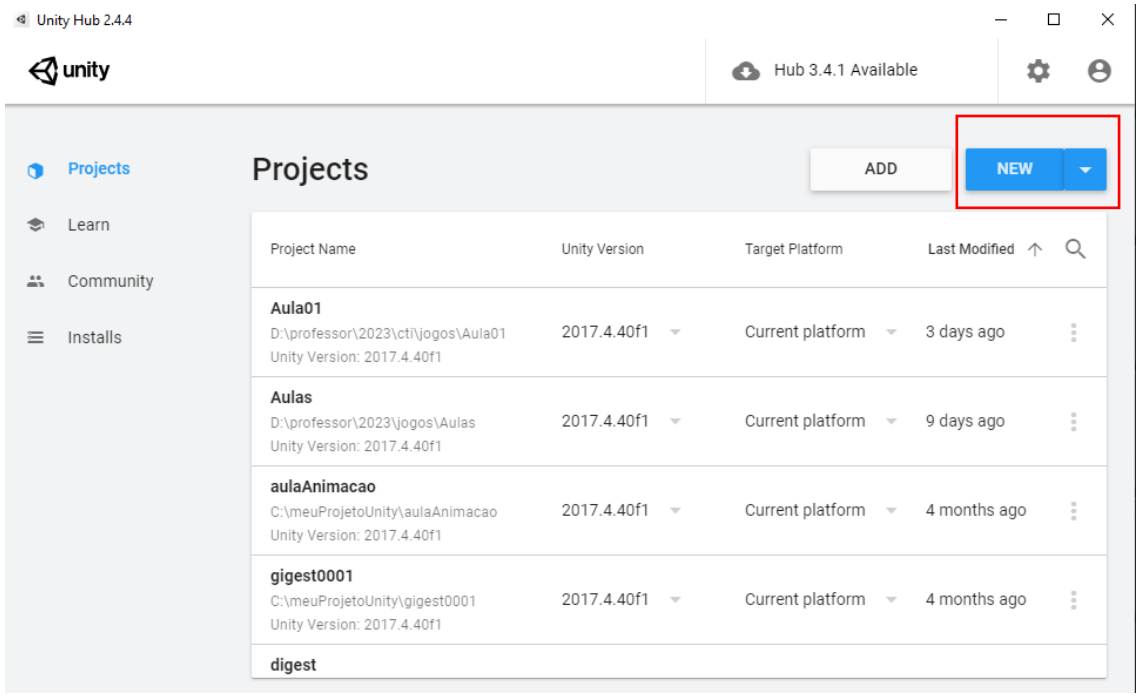
- Permita se for solicitado.



FAÇA NO LABORATÓRIO:

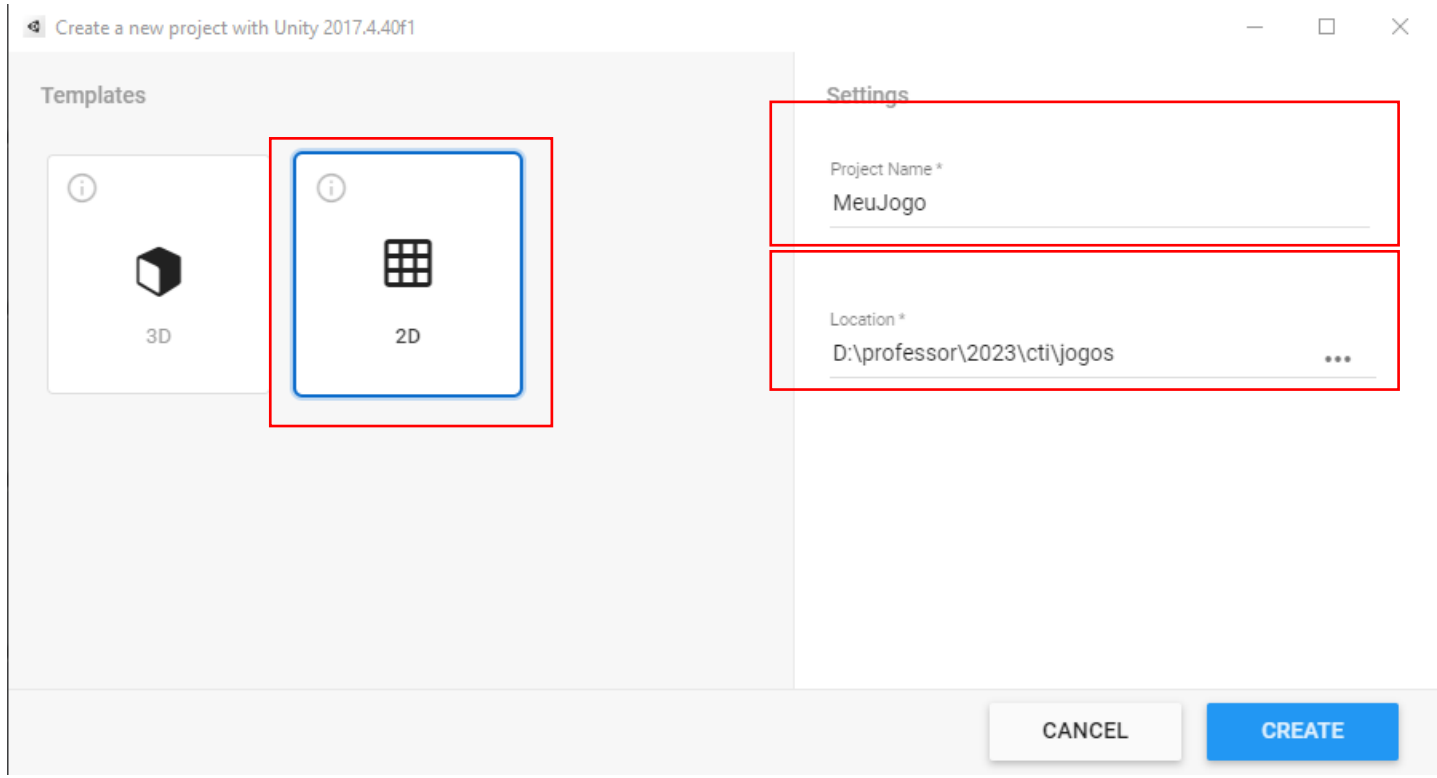
Abra o Unity hub
Criar um novo projeto

- A versão do Unity hub instalada nos laboratórios e ligeiramente diferente
- 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



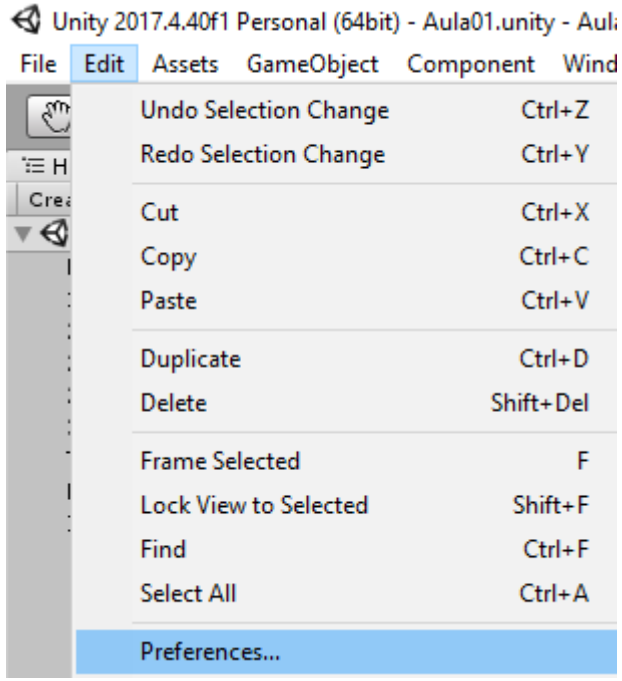
Configurando um editor de código

- Não é preciso fazer essa configuração no laboratório
- Você pode escolher seu editor favorito.
  - Editores mais comuns:
    - Mono Develop
    - Visual Studio
    - Visual studio code.

Configurar o editor de códigos: monoDevelop

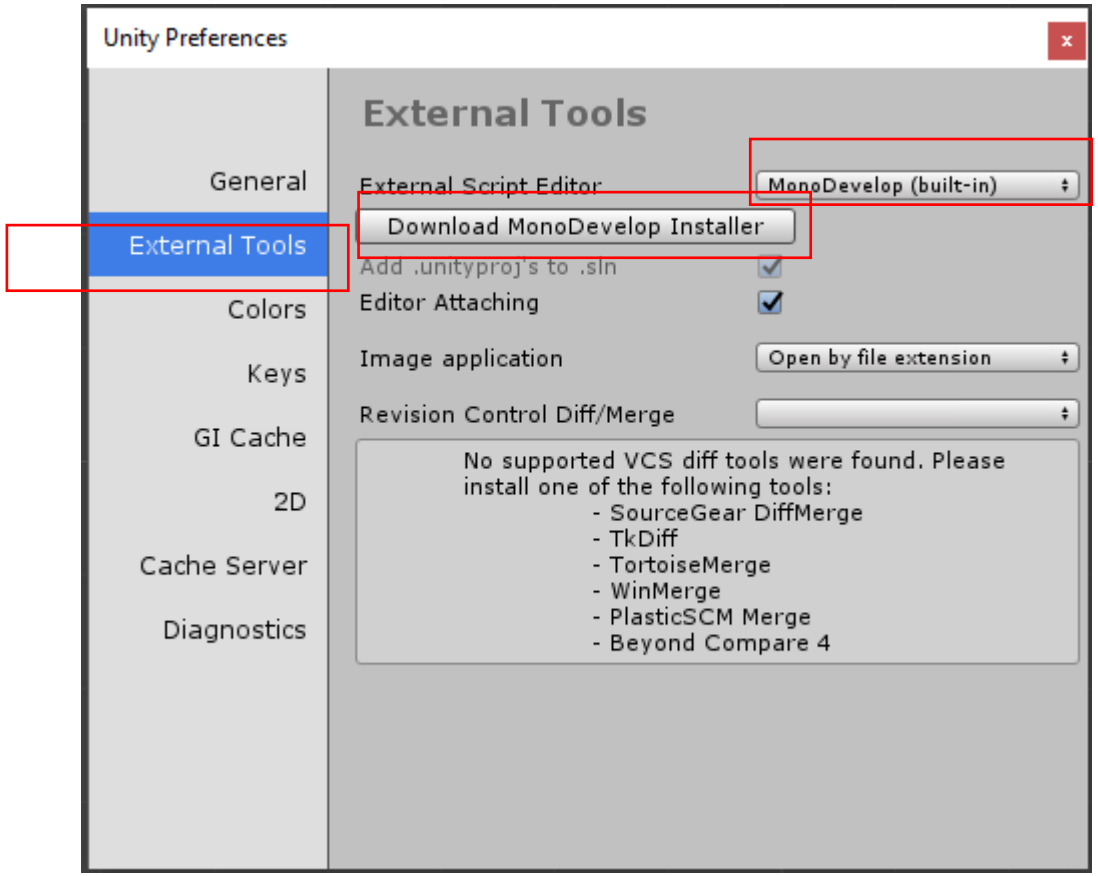
Configurar o Editor de código monoDevelop

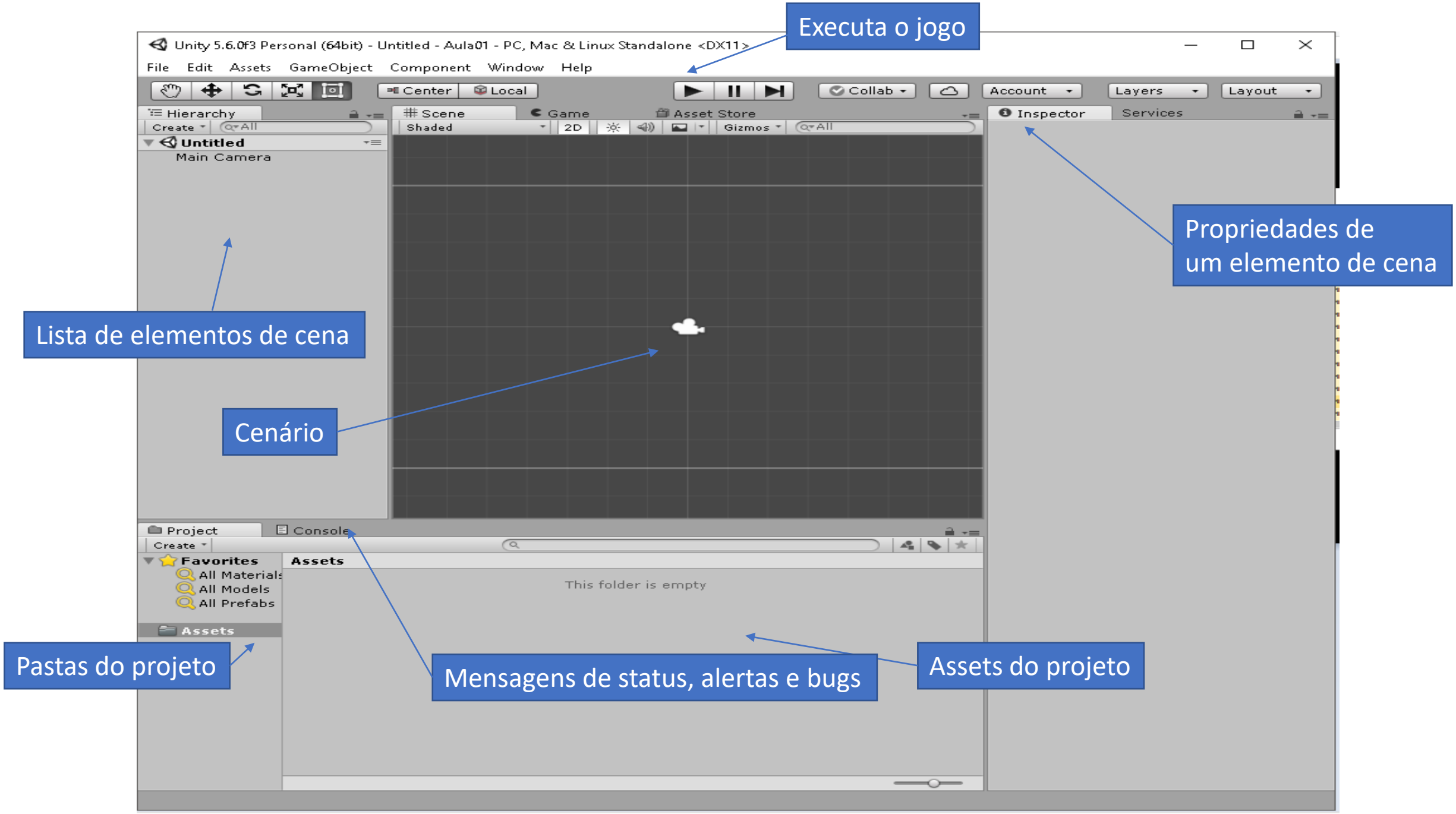
- É necessário realizar esse procedimento apenas uma vez.
- Ao carregar o Unity vá o menu: **“Edit”>>”Preferences…”**



MonoDevelop

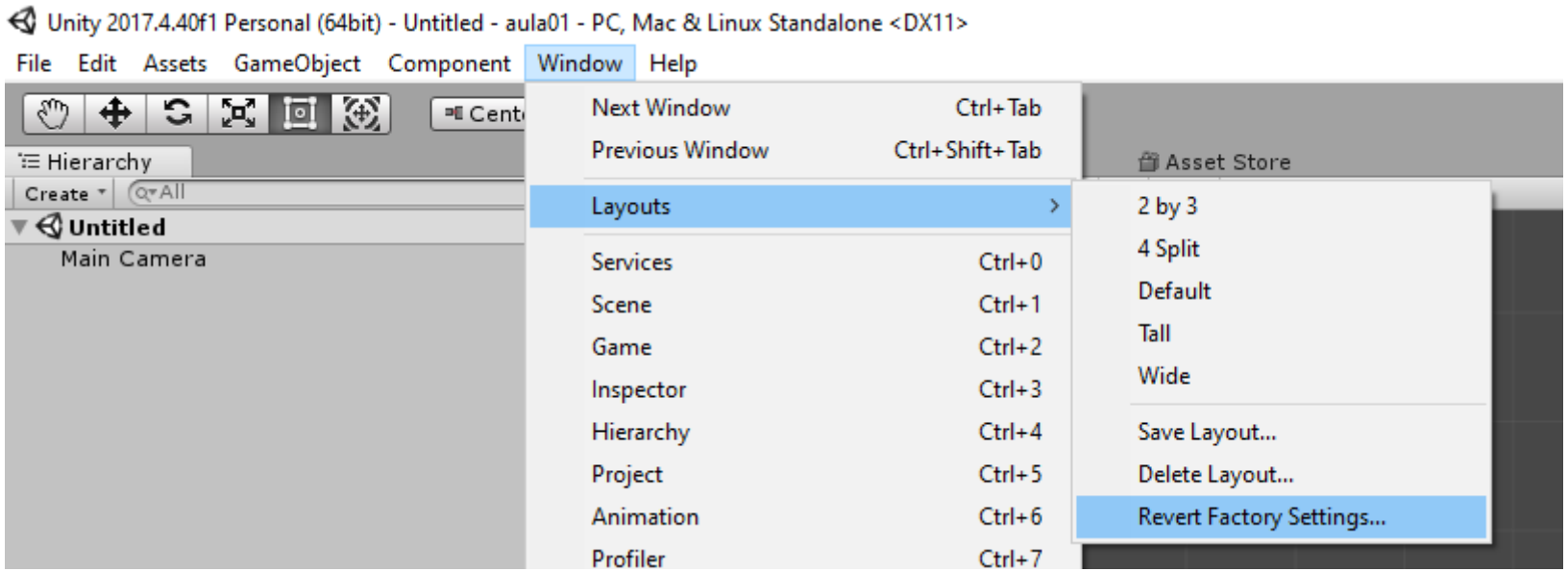
- Click em “**External Tools**”
- Escolha a opção “MonoDevelop (Built-in)”
- Caso seja necessário o unity indicará um link para download.
  - Caso necessário faça o download e instale o **monoDevelop**.
- Feche a janela “**Unity Preferences**”




Executa o jogo
Unity 5.6.0f3 Personal (64bit) - Untitled - Aula01 - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Untitled
Main Camera
Scene Game Asset Store
Shaded 2D Gizmos All
Inspector Services
Propriedades de um elemento de cena
Lista de elementos de cena
Cenário
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
Assets
Assets
This folder is empty
Pastas do projeto
Mensagens de status, alertas e bugs
Assets do projeto

Restaurando o Layout Original

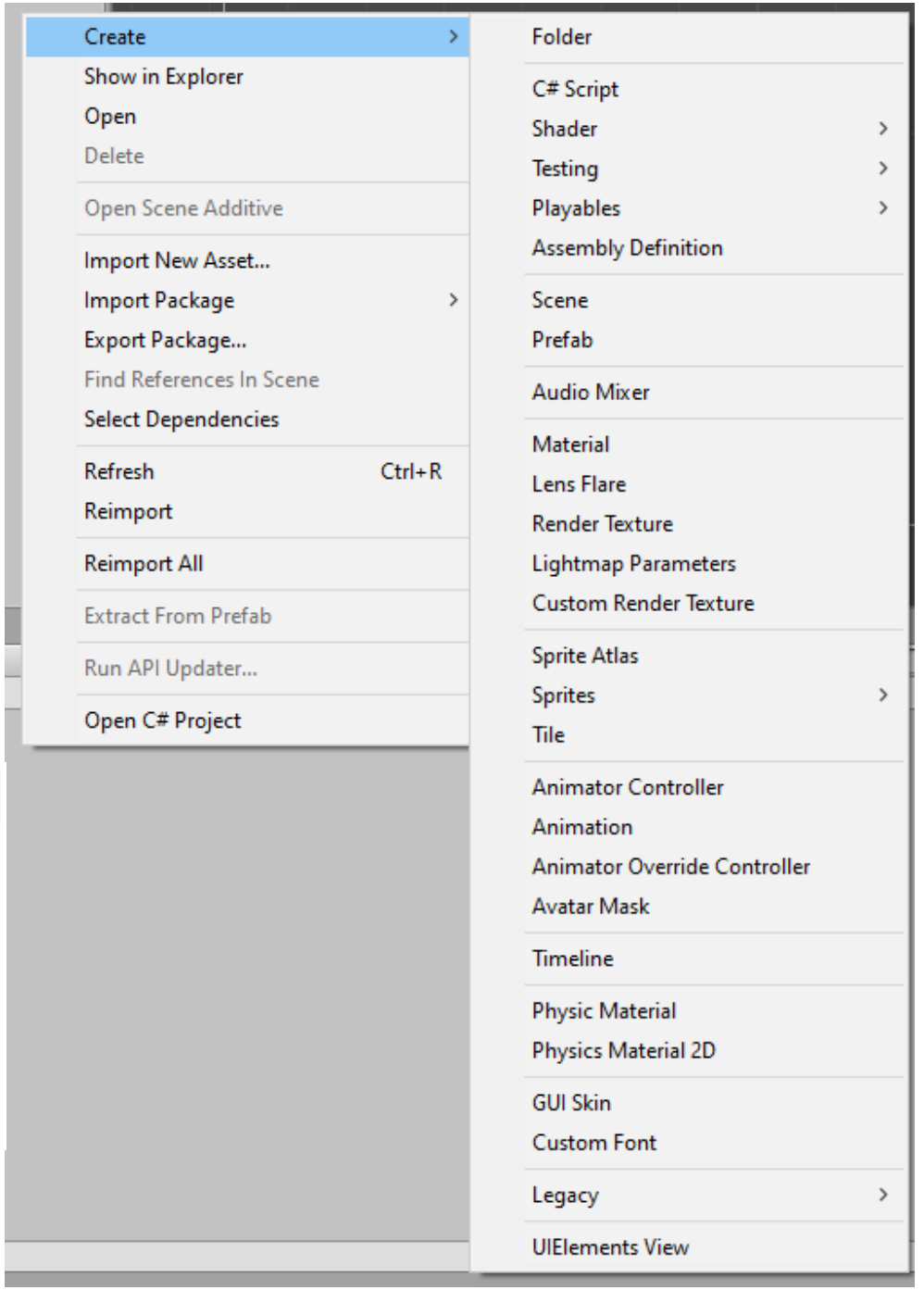
- Se perder ou fechar algum dos painéis e quiser restaurar o layout original do programa, acesse a opção Window > Layouts > Revert Factory Settings



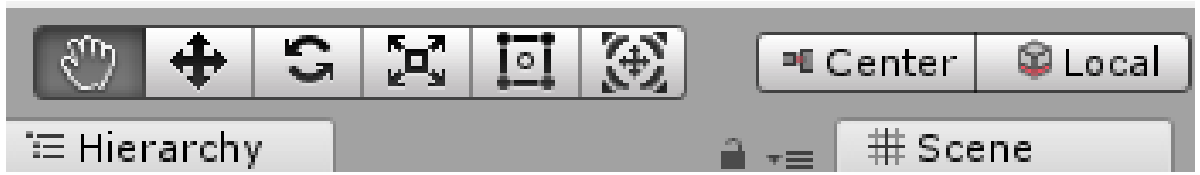
Estrutura do Projeto

Podemos ignorar todas as pastas e nos atentar apenas à pasta Assets, que é onde ficarão todos os arquivos e recursos do nosso jogo (modelos 3D, sons, gráficos, telas e botões).

Nome	Data de modificação	Tipo	Tamanho
Assets	11/08/2022 09:10	Pasta de arquivos	
Library	11/08/2022 09:10	Pasta de arquivos	
ProjectSettings	11/08/2022 09:10	Pasta de arquivos	
Temp	11/08/2022 09:10	Pasta de arquivos	
UnityPackageManager	11/08/2022 09:10	Pasta de arquivos	
Aula01.sln	11/08/2022 09:10	Visual Studio Solu...	1 KB

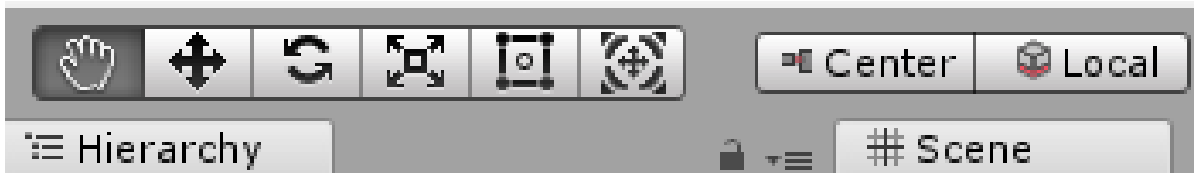


Editor de Cena e Navegação 3D



Atalho de Navegação	Função
Alt + clique Esquerdo	Orbit
Click do Scroll	Pan
Scroll	Zoom
F	Zoom no objeto selecionado
Z	Alinhamento do Pivot (Object/Center)
X	Orientação do Pivot (Local / World)

Atalho útil

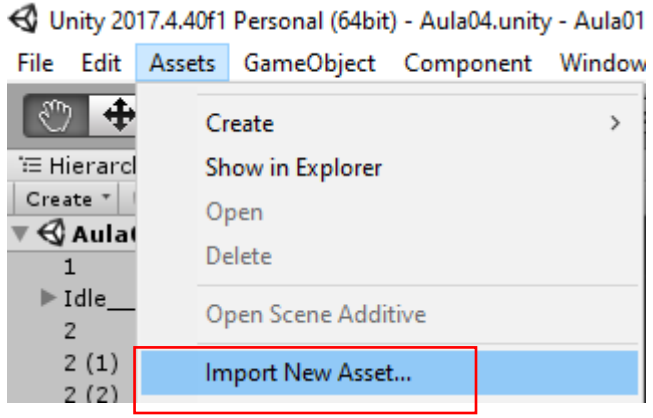


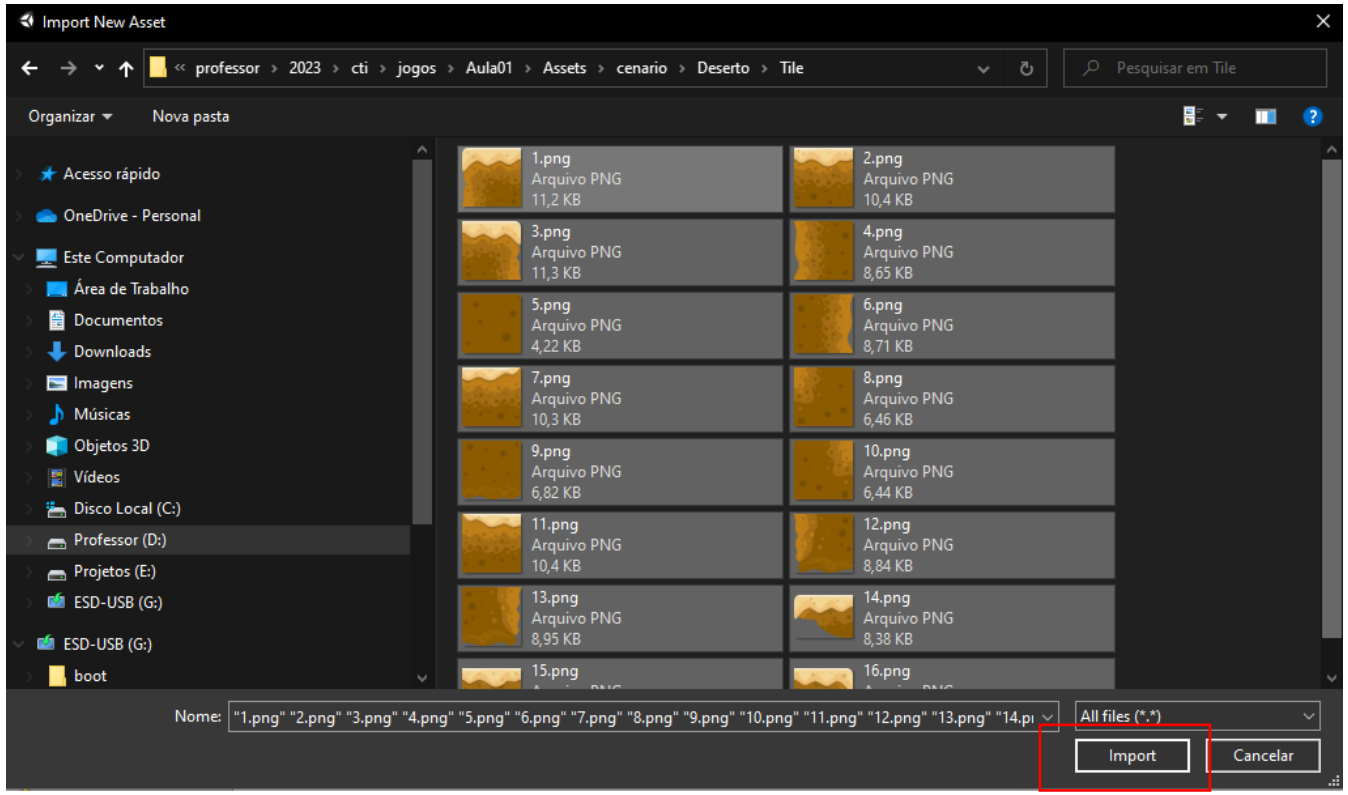
Gizmos de Manipulação	Função
Q	Pan
W	Move
E	Rotate
R	Scale
T	Rect Tool
Y	Move Rotate Scale (segure Shift para View Align)

Outros comandos	Função
Ctrl + P	Executar Aplicação (Play/Stop)
Shift+ Espaço	Tela Cheia na janela ativa (em foco)
Ctrl + D	Duplicar
Ctrl + N	Novo Node – Empty

Acrescente seus Assets

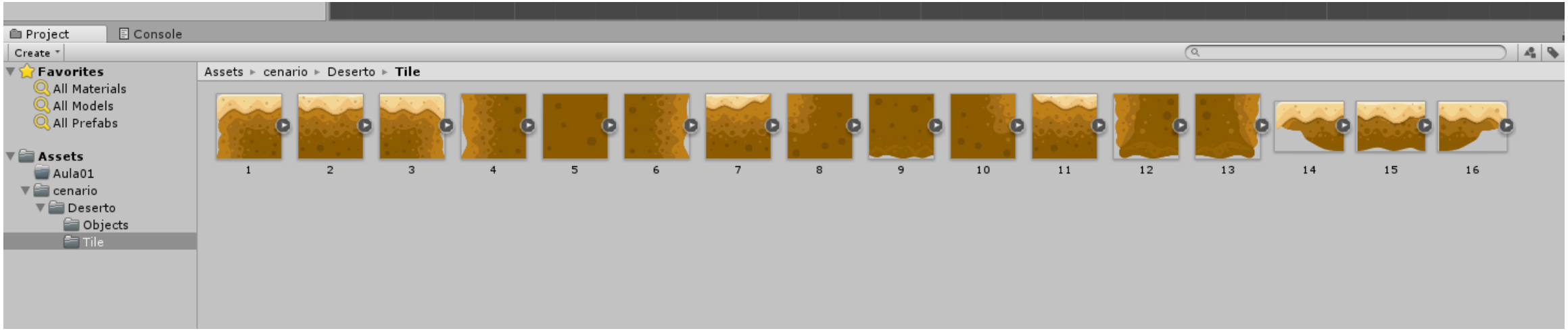
- Click no menu: **Assets** >> **Import new Asset**
- Selecione os arquivos de assets desejados e click em **Import**
- Caso possua muitos assets crie mais pastas dentro da pasta Assets para manter a organização





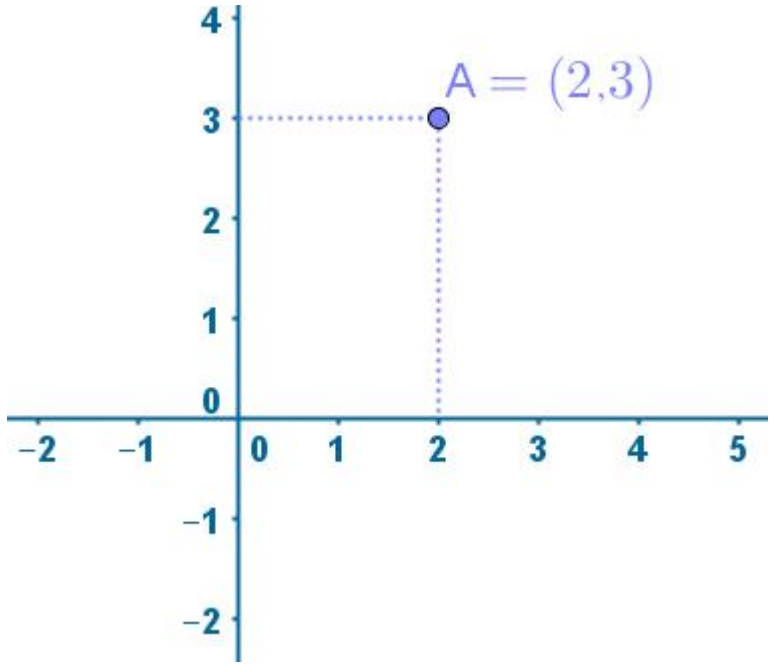
Após importar

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



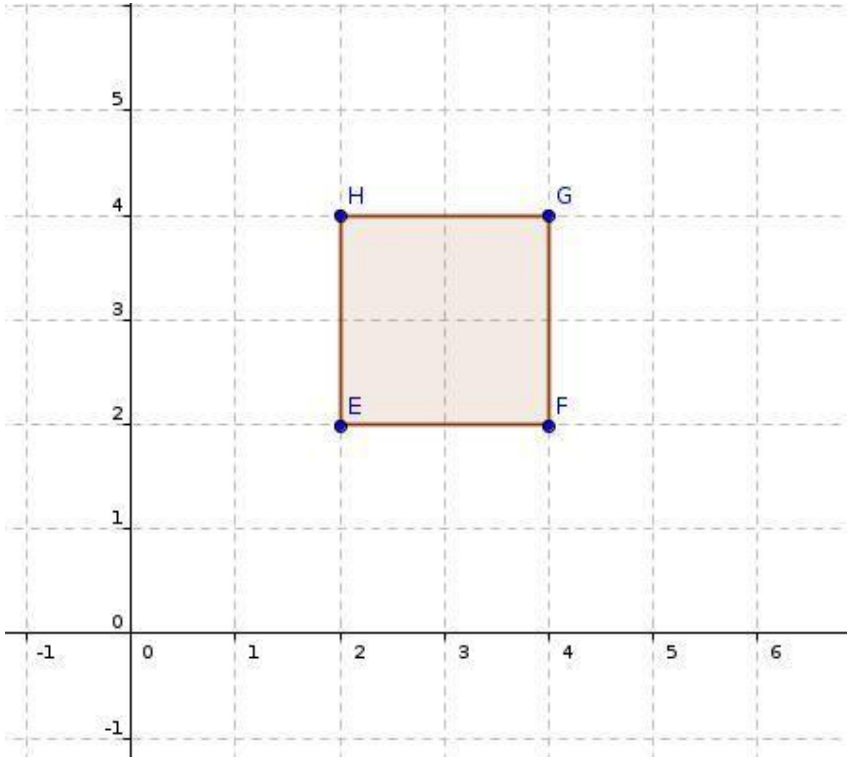
Posição no plano 2d

- O Unity trabalha com conceitos de posicionamento no plano.
- Observe que o ponto **A** possuí **x=2 e y=3**



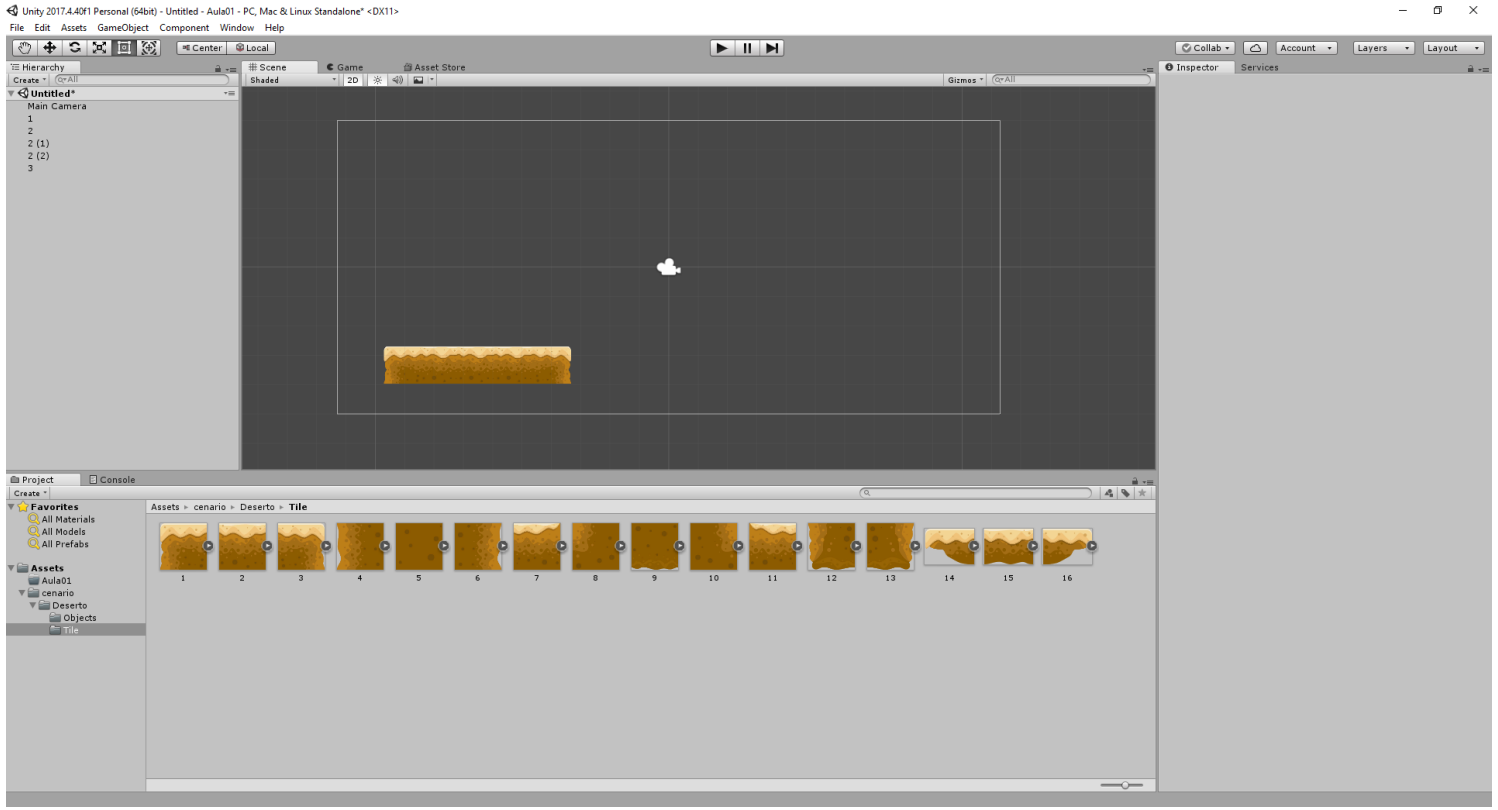
Posição dos assets no cenário 2d

- Todo asset posicionado no cenário é arranjado conforme as posições x e y.
- Observe ao lado um bloco posicionado no plano.

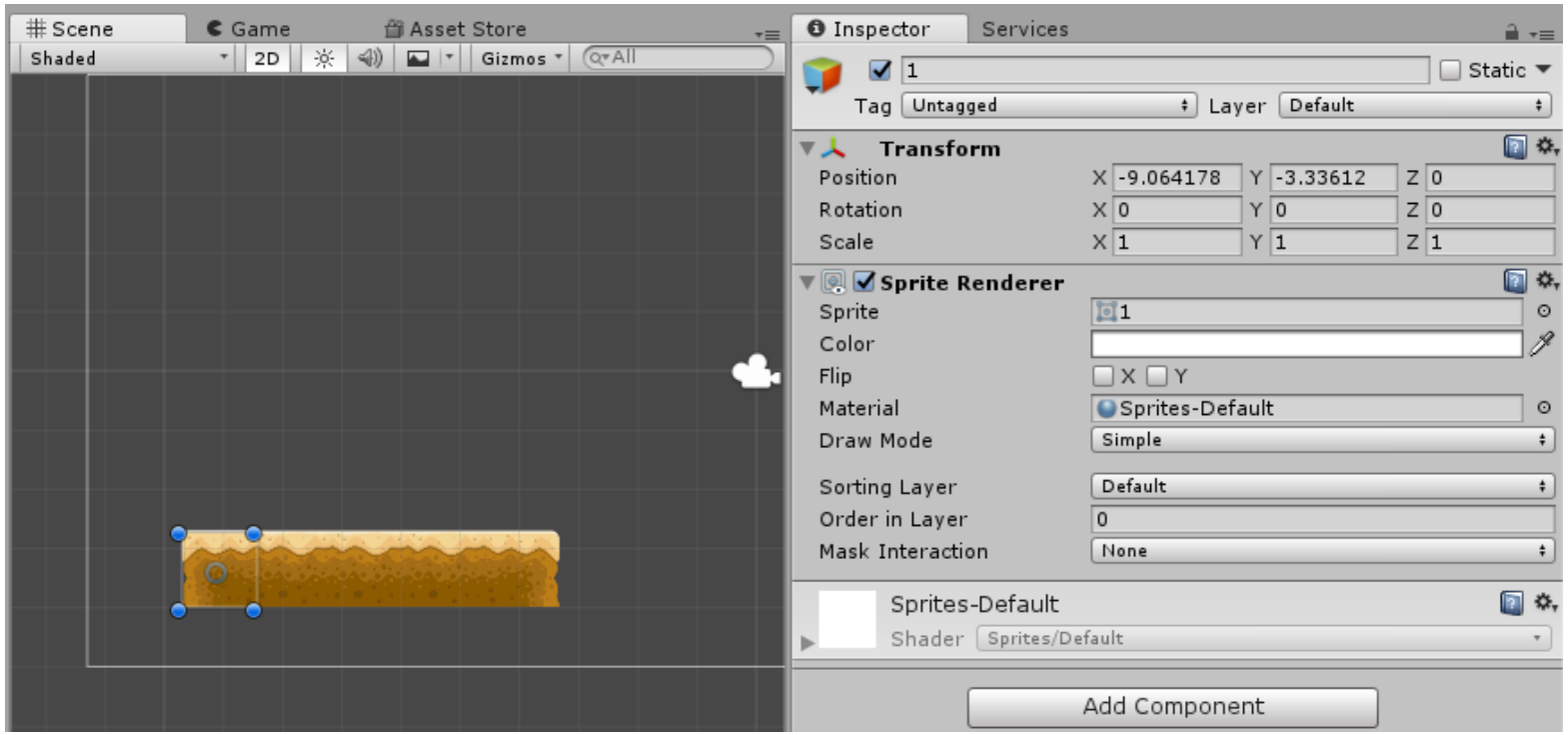


Posicione os Assets no seu cenário

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

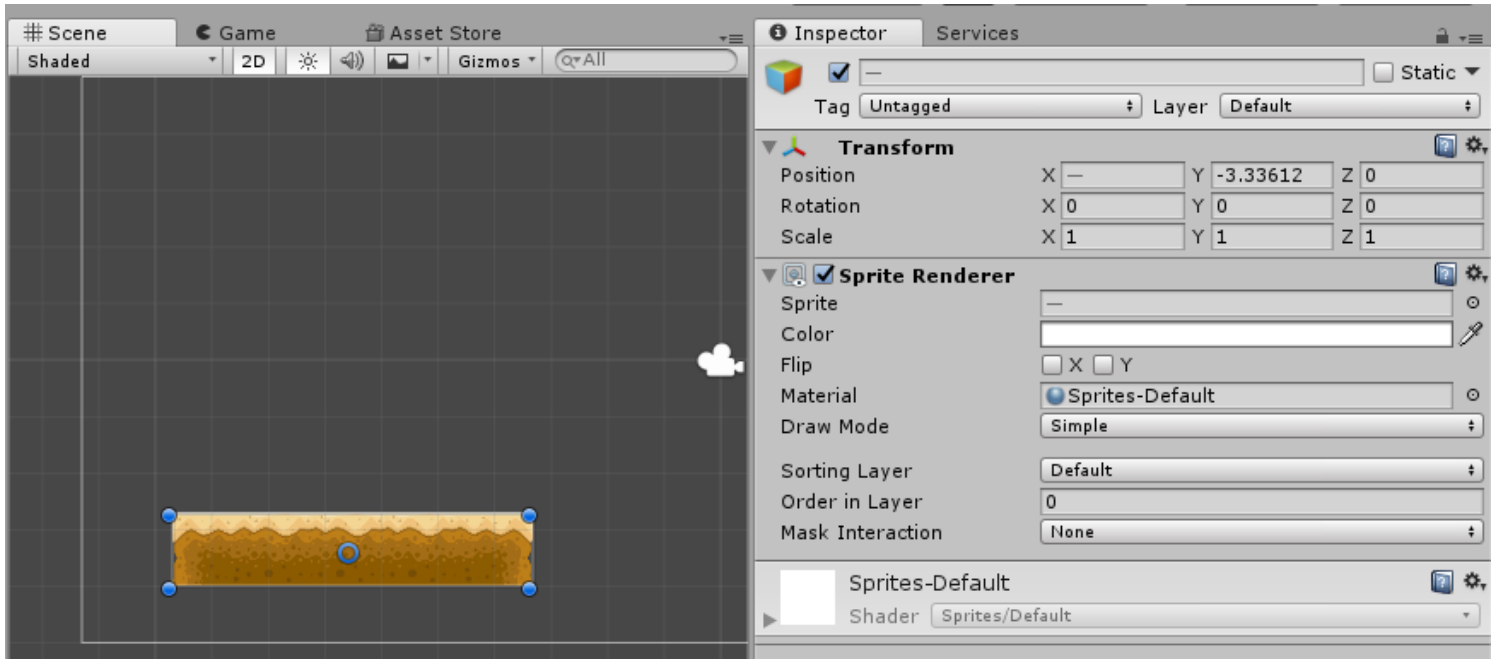


Propriedade: Position



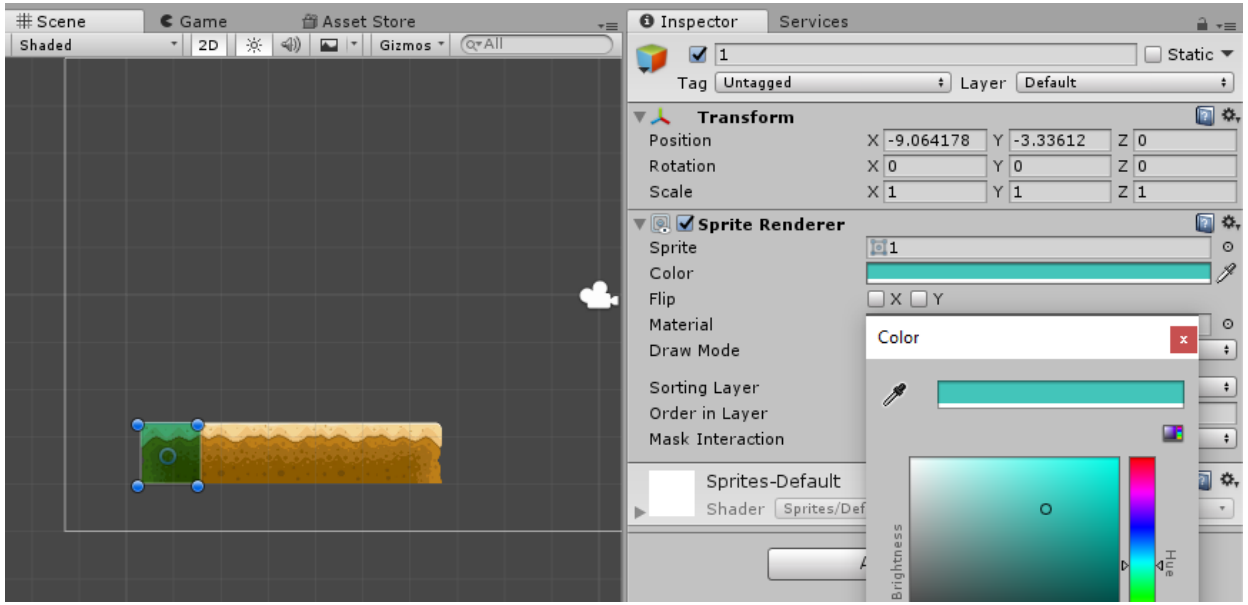
- Cada elemento posicionado no cenário possui uma série de propriedades
- Ao clicar no objeto de cena conforme figura ao lado é possível verificar todas as suas propriedades.
- Exemplo de propriedade: **Position: x,y e z** equivalem a posição de cada elemento dentro do cenário.

Propriedade: Position



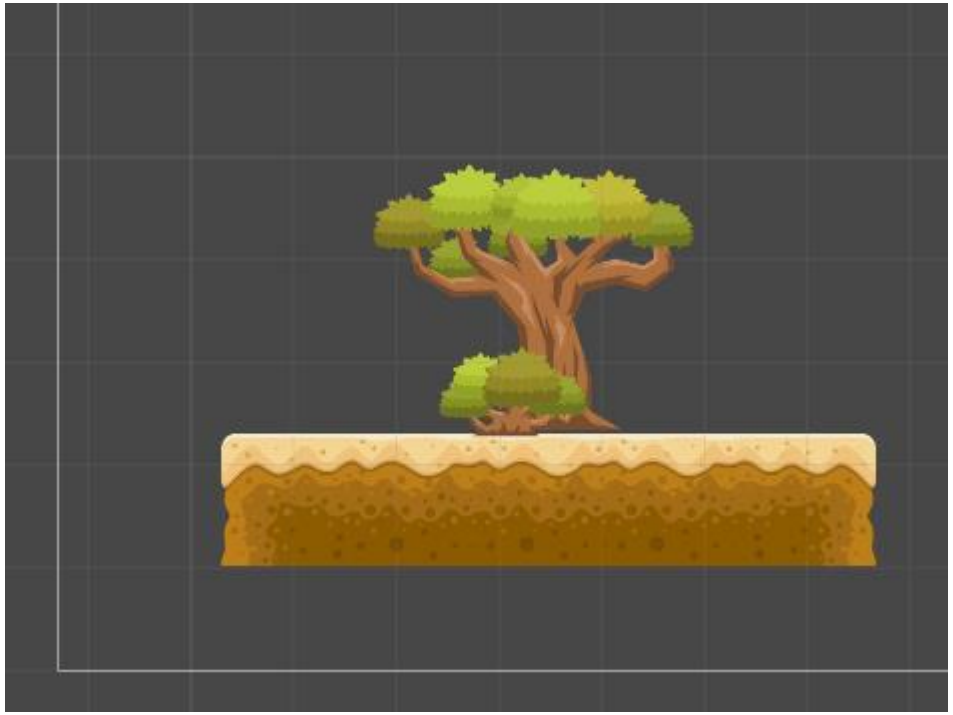
- A imagem ao lado representa a seleção de um conjunto de objetos de jogo.
- É possível verificar que a propriedade Position Y de todos eles é igual.
- Isso permite que aconteça uma alinhamento em **y**, ou seja não há um elemento com alinhamento mais alto que o outros, todos possuem a mesma altura de alinhamento.

Propriedade: Color



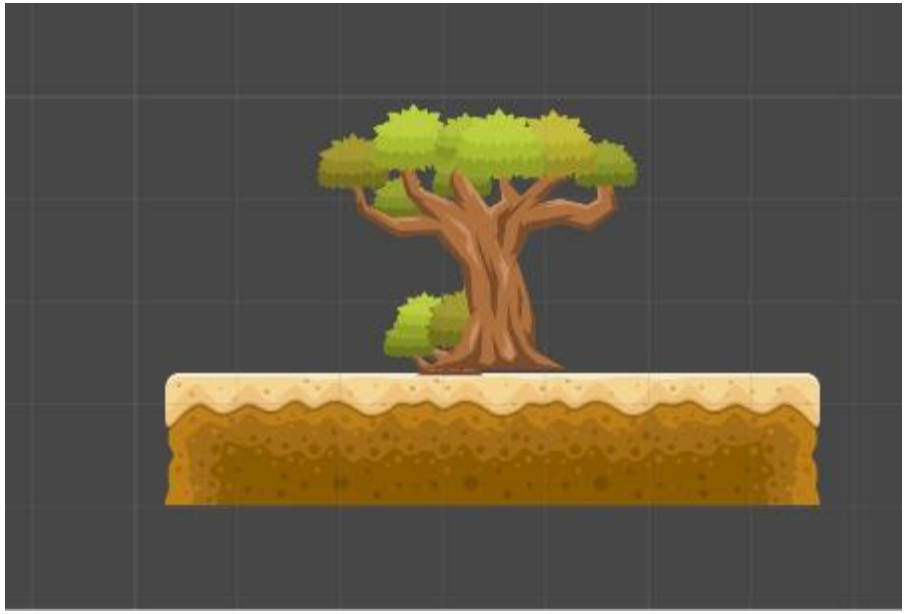
- Existem outras propriedades além das propriedades de posicionamento.
- A propriedade **color** permite aplicar um filtro de cor a um determinado **Asset** conforme figura ao lado.

Propriedade: Order Layer



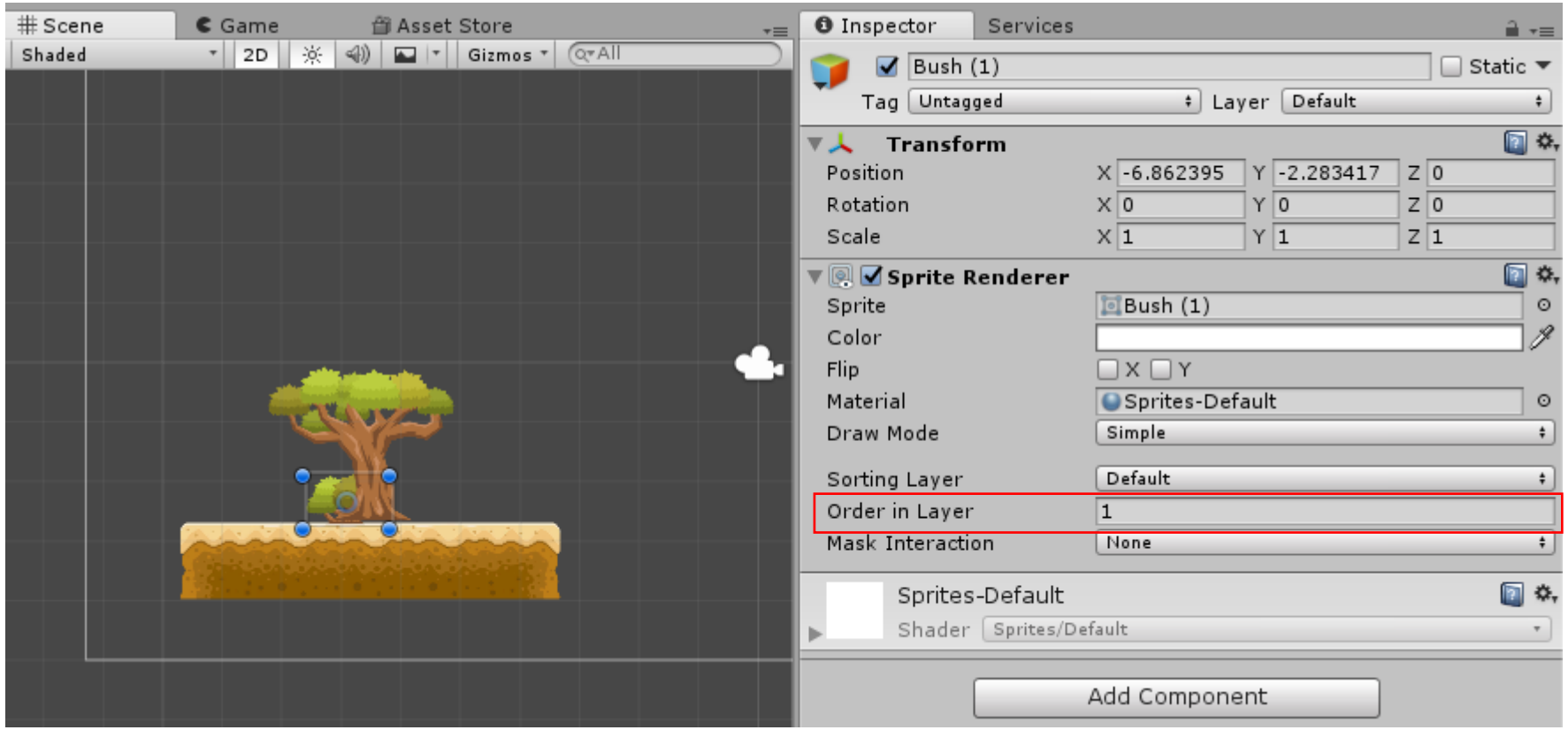
- Essa propriedade permite definir quando um objeto é posicionado a frente ou atrás de outro objeto.
- Observe a figura ao lado, o arbusto deve possuir um **order layer** maior que o **order layer** da árvore, pois o arbusto está a frente da árvore.

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

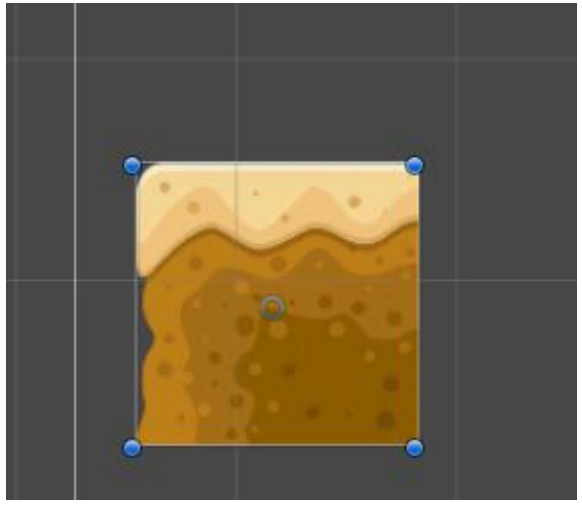
Componentes que marcam o volume físico de um objeto, assim como o material físico (com atrito e elasticidade) que define parte de seu comportamento.

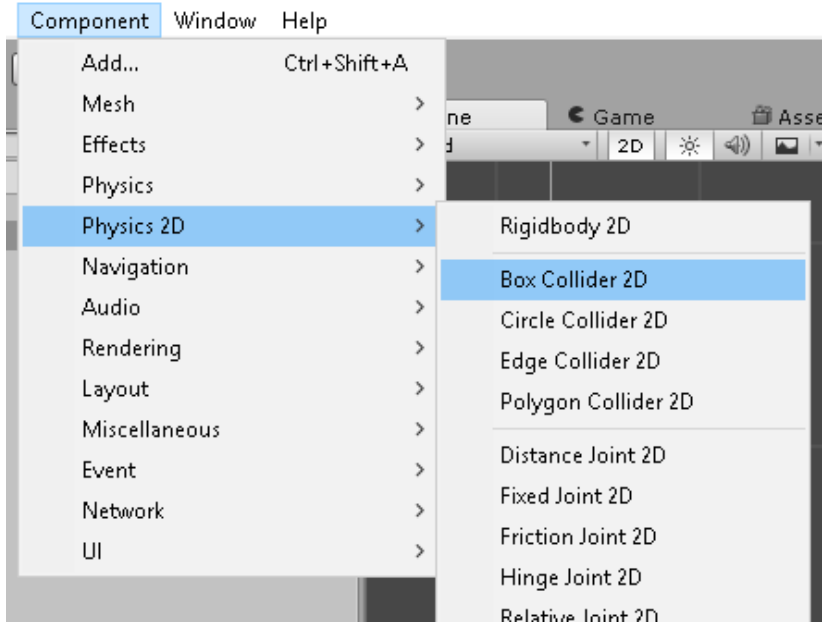
Existem vários tipos de colisores, cada um com formato diferente (por exemplo, BoxCollider, SphereCollider).

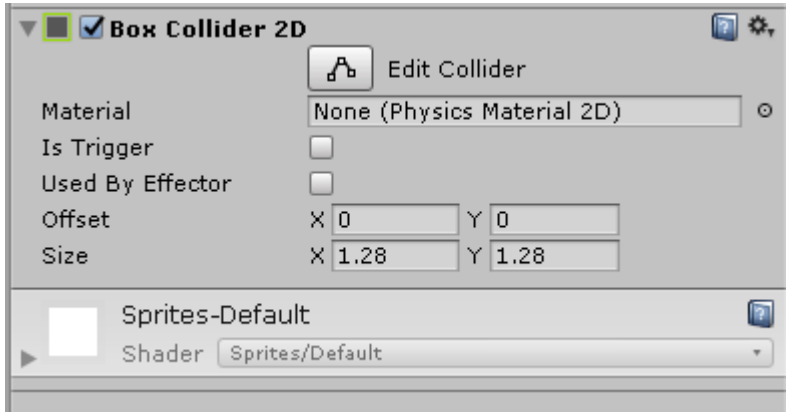
Também mantem informações sobre o tipo de interação que tem com outros objetos (colisão ou sobreposição). Esse componente tem métodos e eventos que utilizamos para criar lógica para interações físicas, como OnCollisionEnter e OnCollisionExit. Para que um script possa utilizar essas funções-evento, o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

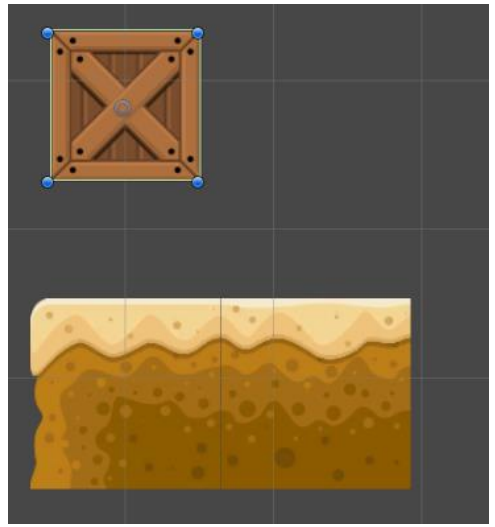
Corpos rígidos

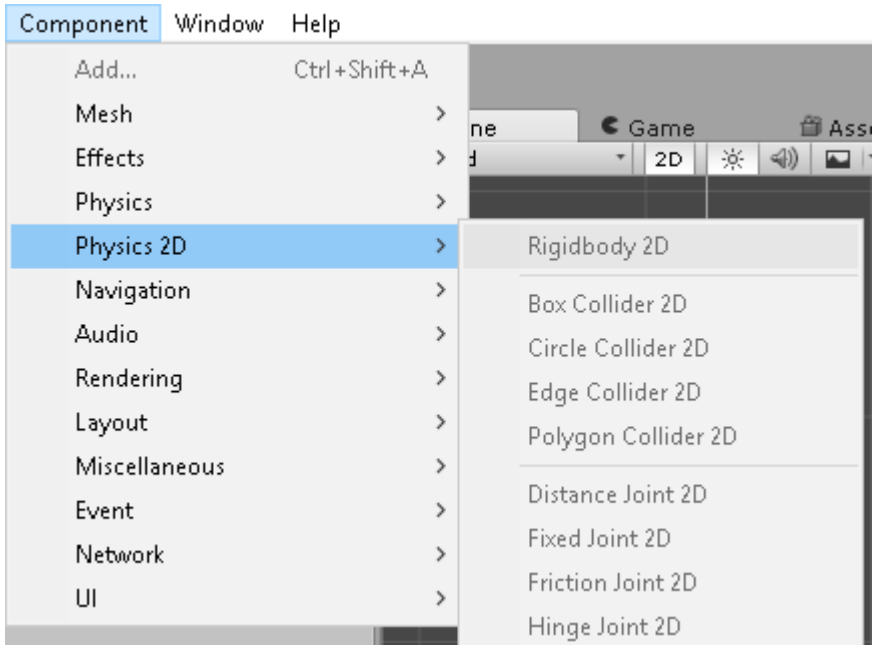
- O principal tipo de física usado em jogos é a física de corpos rígidos, que considera que objetos não deformam ou alteram seu volume e material quando afetados por interações como colisões.

- 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.



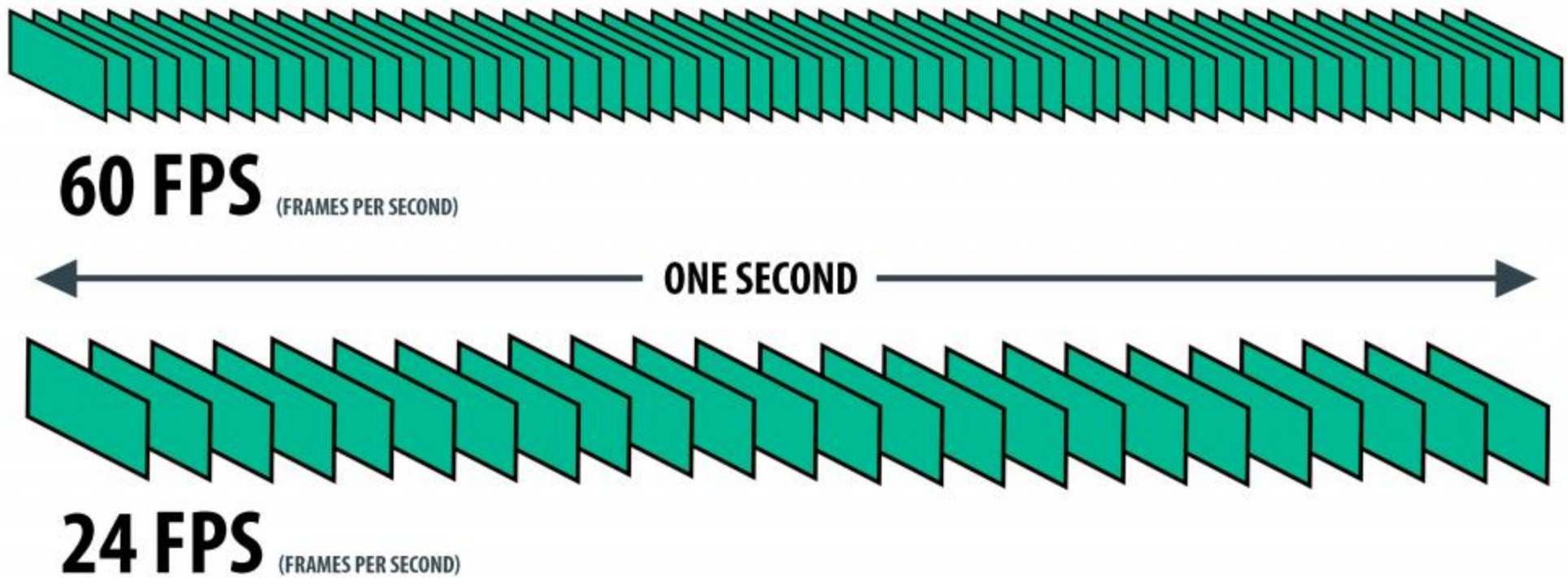


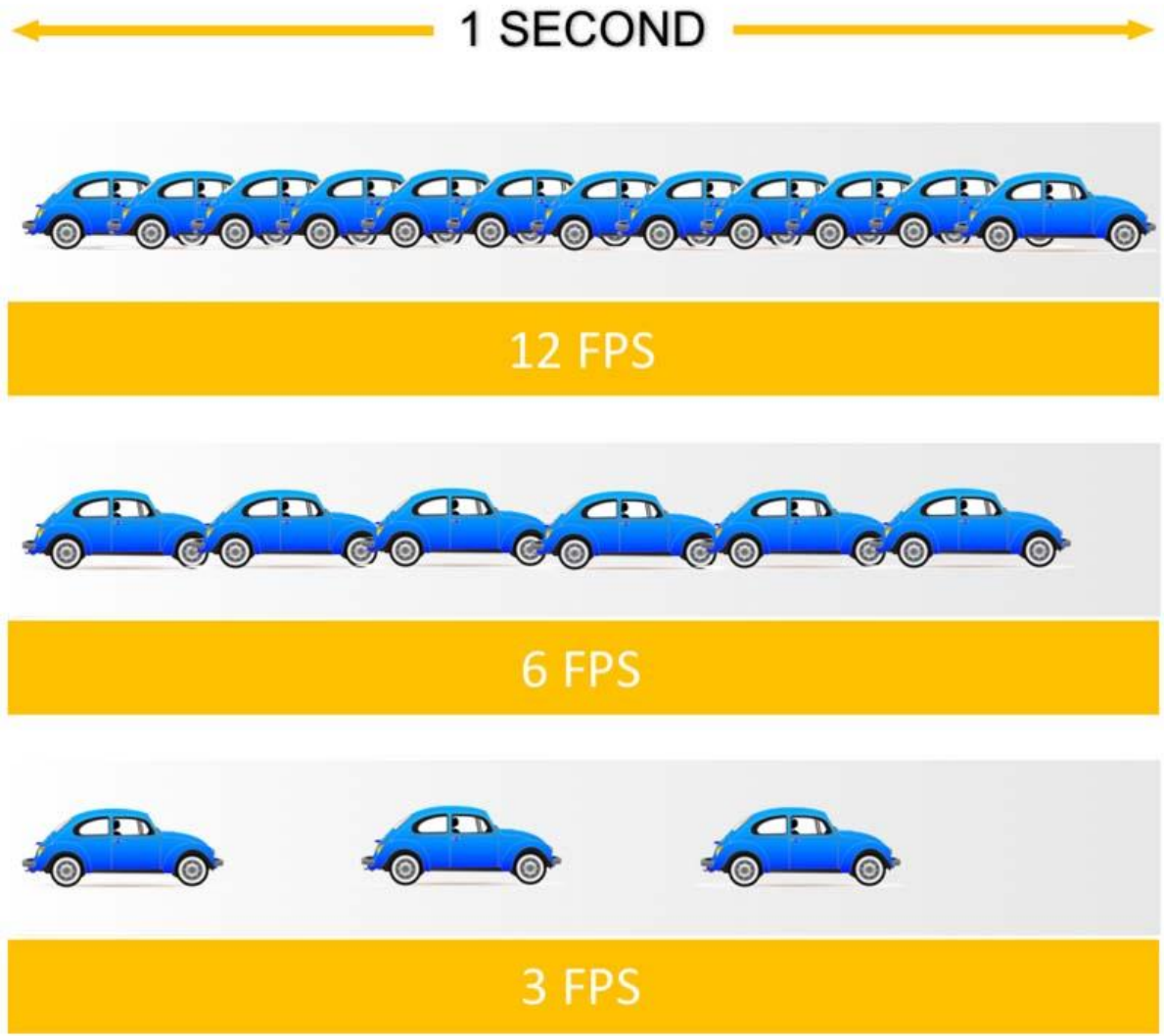
Faça suposições

- 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)

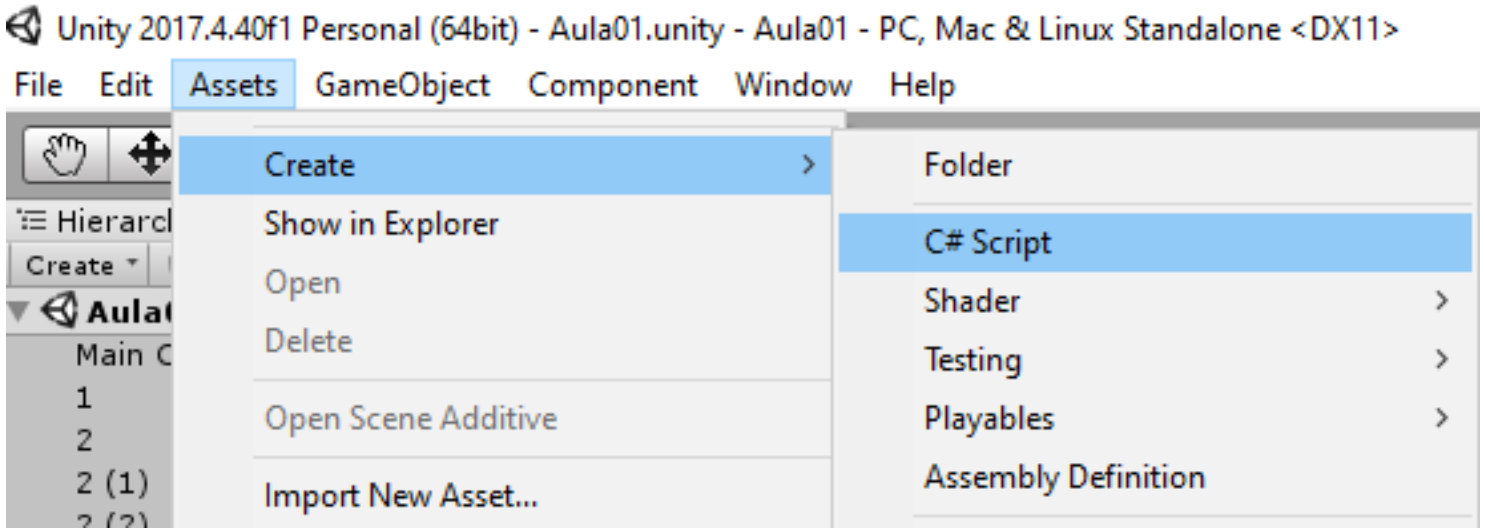

1 SECOND
12 FPS
6 FPS
3 FPS

Script na computação

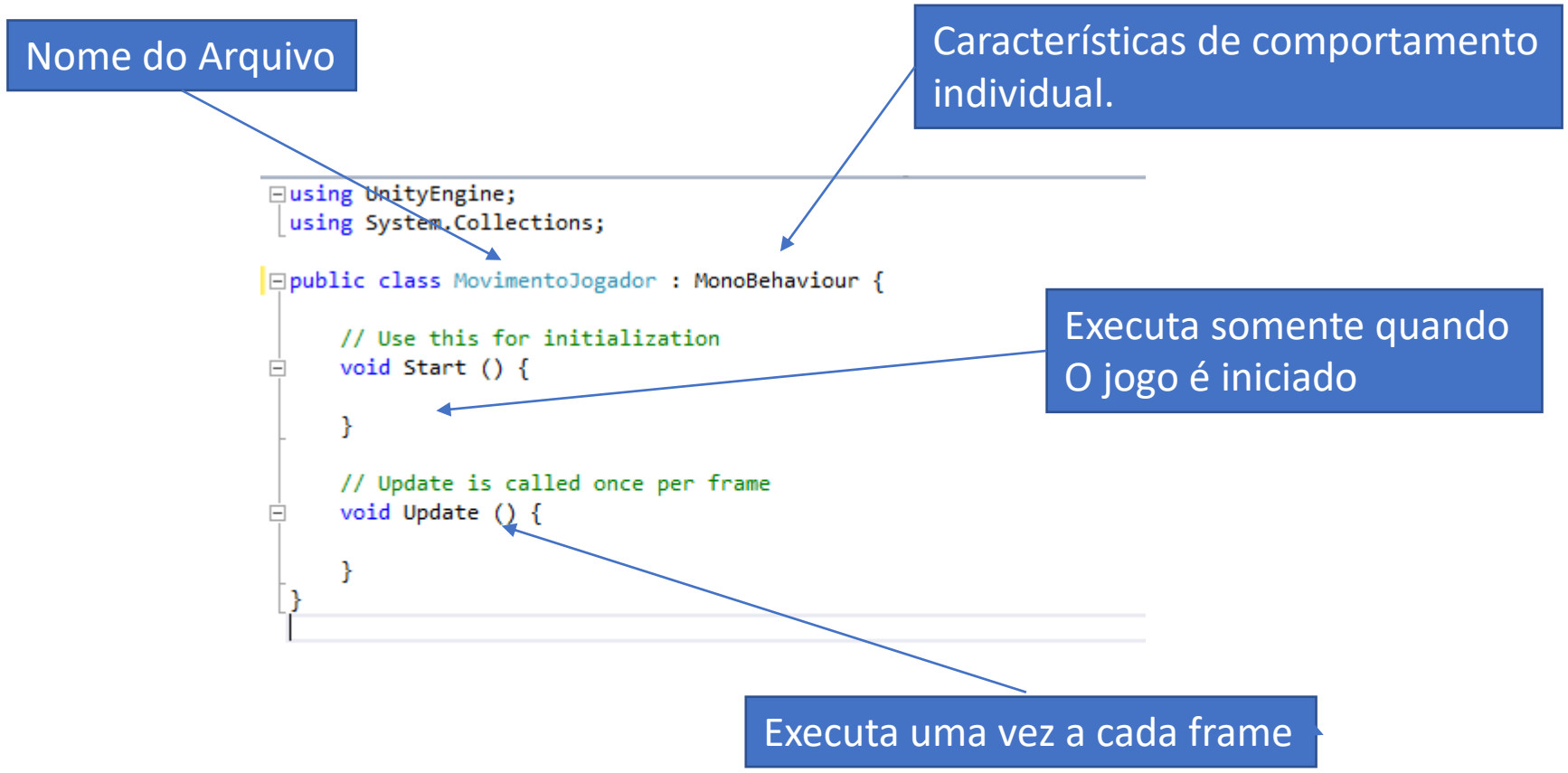
- Script é um texto com uma série de instruções escritas para serem seguidas por um computador.
- O termo é uma redução da palavra inglesa manuscript, que significa “manuscrito”, “escrito à mão”.

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



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: velocidade atual em x do personagem**
- **Y: velocidade atual 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

Salvar o projeto

- Lembre-se da pasta onde você escolheu para salvar os arquivos do projeto 2d.
- No exemplo ao lado os arquivos serão salvos na pasta:
  - **d:\professor\2023\cti\jogos\meujogo**
- Antes de fechar o jogo salve a cena.
- Menu:
  - File>> Save Scene>> de um nome para o cenário
- Salve todo o conteúdo da pasta:
  - **d:\professor\2023\cti\jogos\meujogo**
- **Veja**: https://youtu.be/mT122XNuAPU