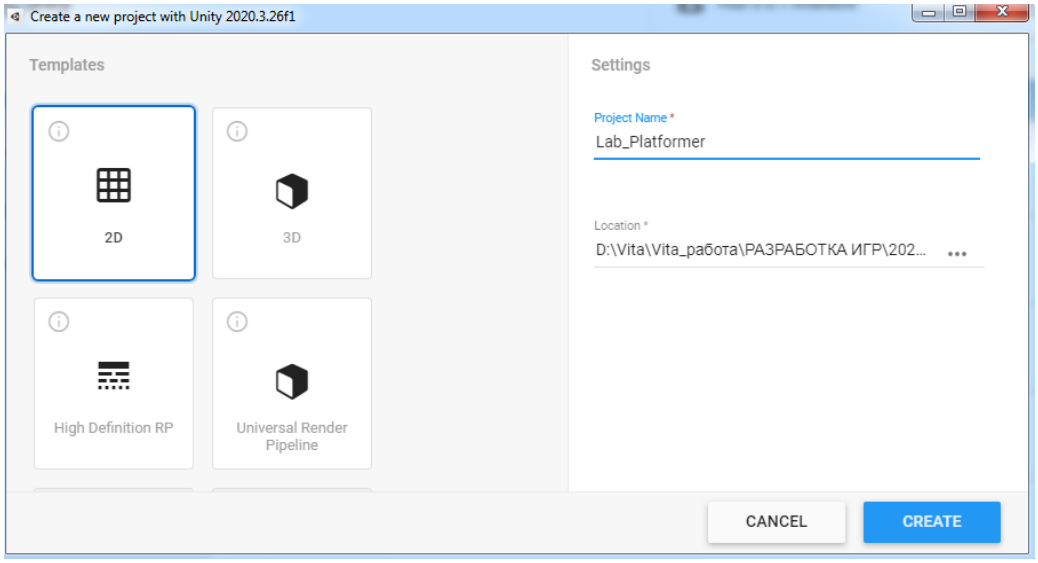
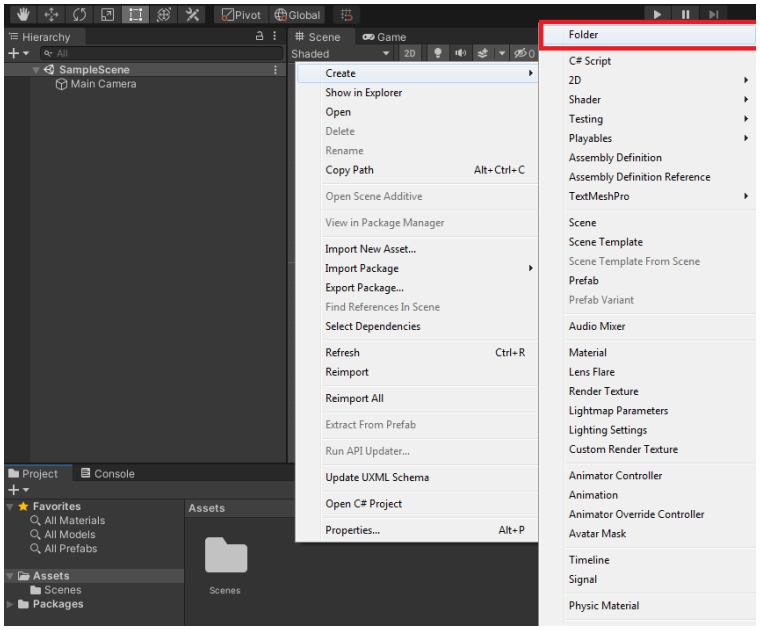
Лабораторная 2. «Создание 2D платформера»

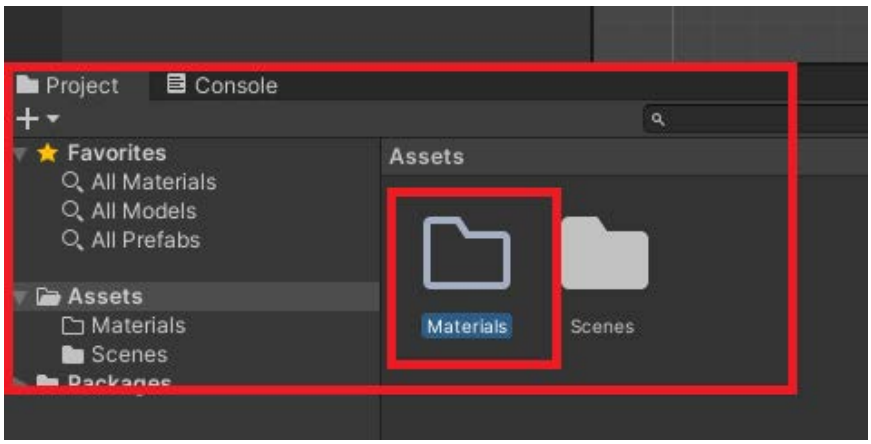
1. Создаем 2D проект

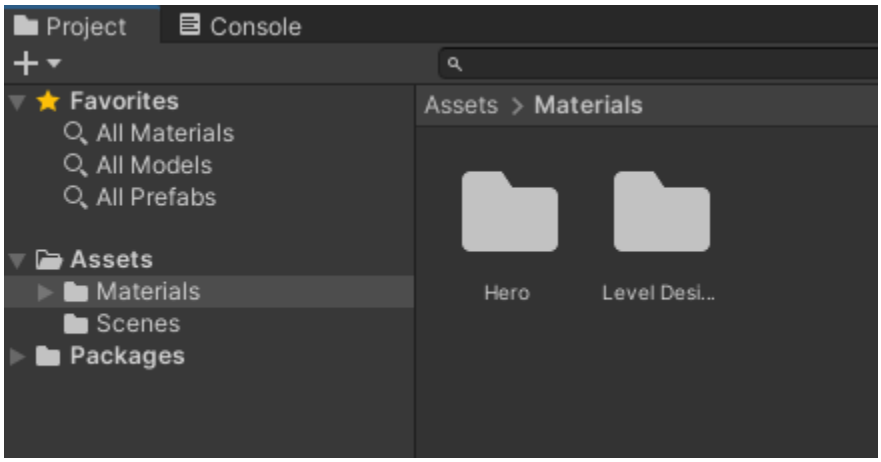


2. Используем пак для графики

После создания нового проекта создаем папку Materials и скопируем две папки (Hero и Level Design)

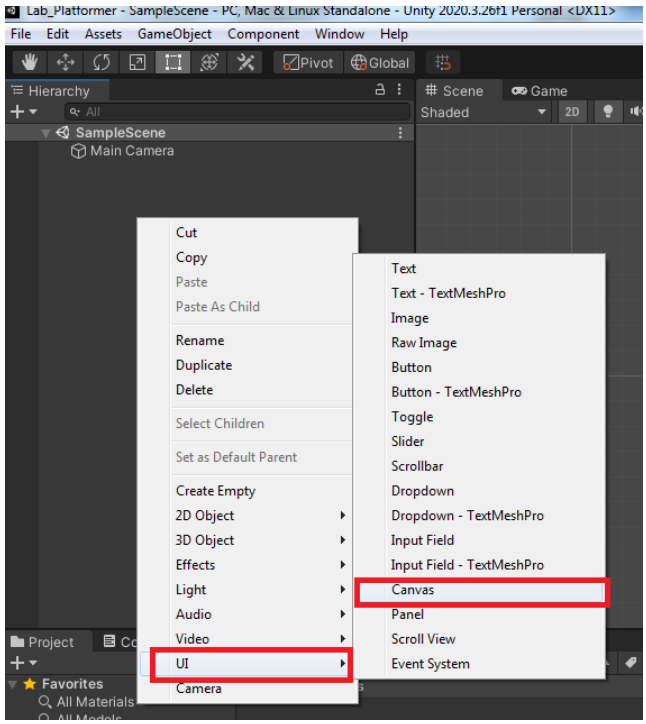




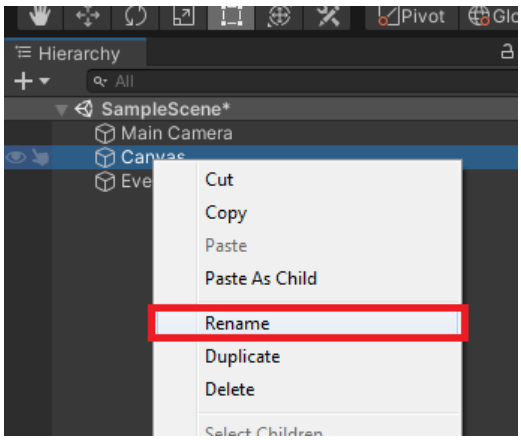


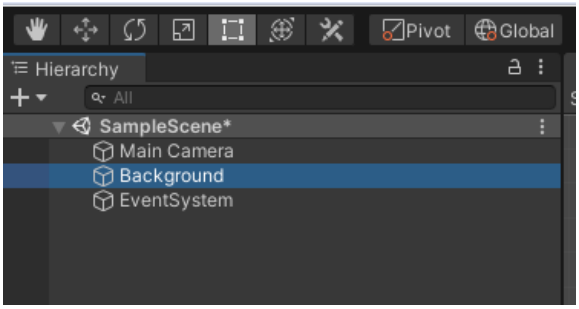
3. Создаем задний фон.

3.1. Создаем элемент Canvas



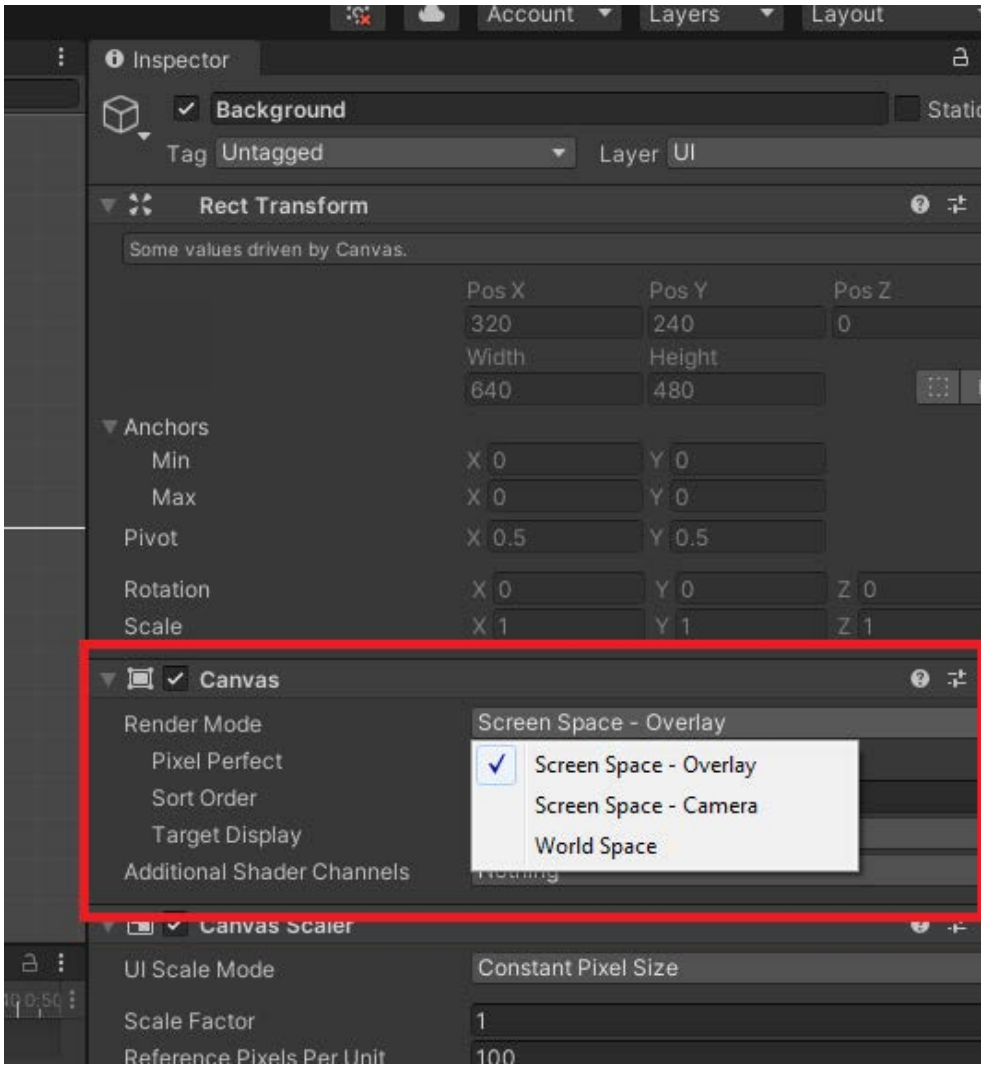
3.2. Переименуем его в Background



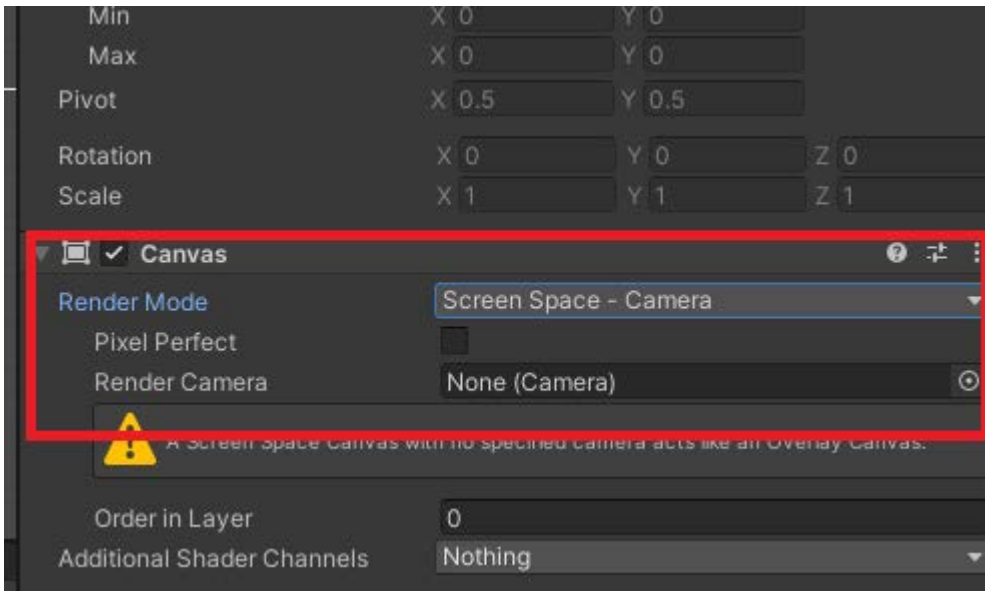


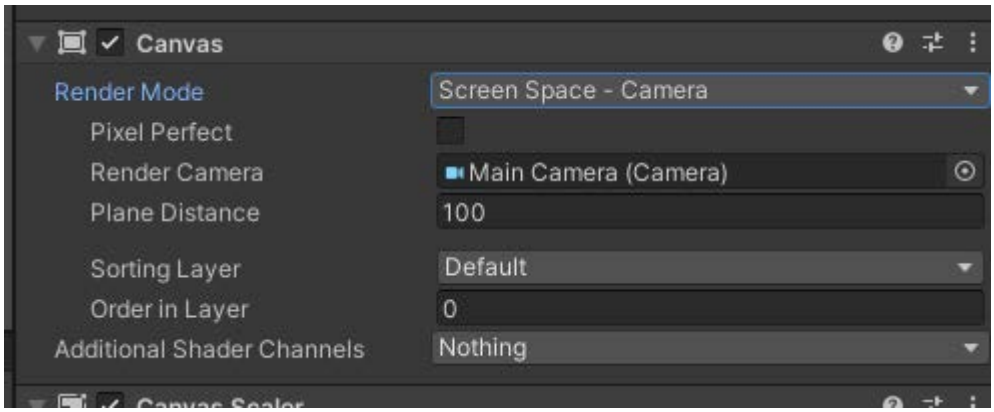
3.3 Меняем в Inspector

Screen Space – Overlay на Screen Space – Camera

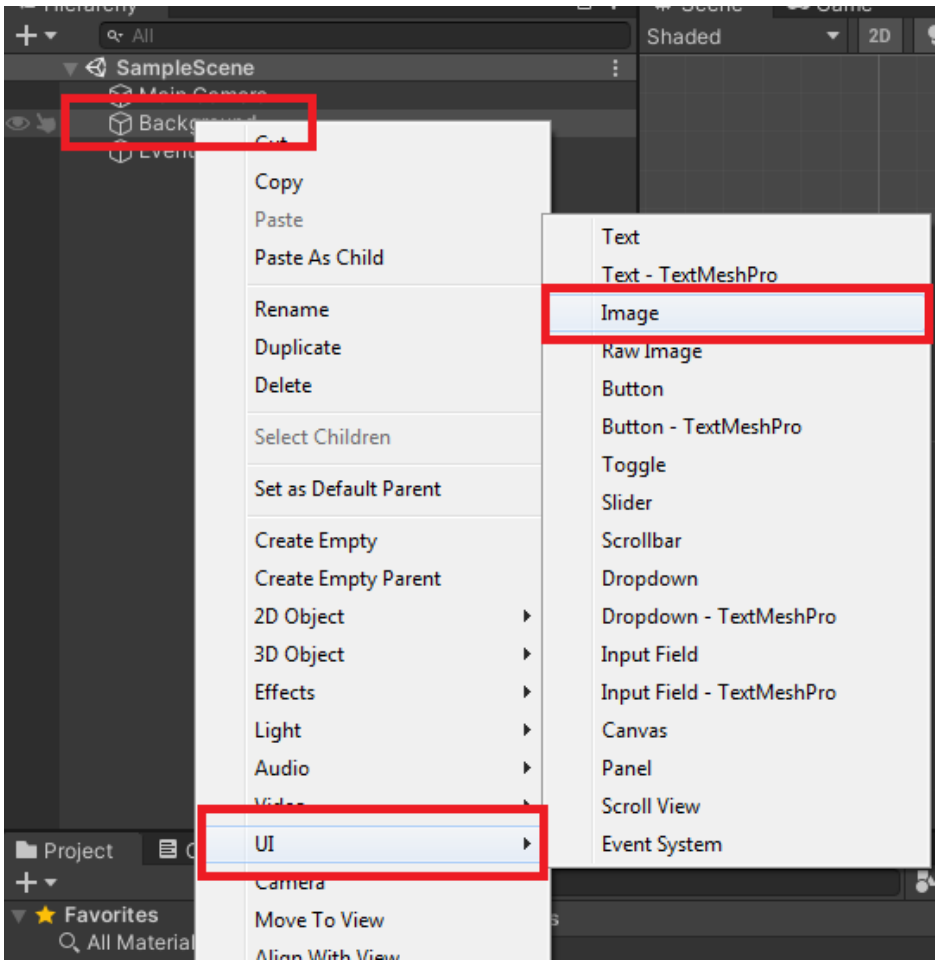


3.4 Перетаскиваем Main Camera

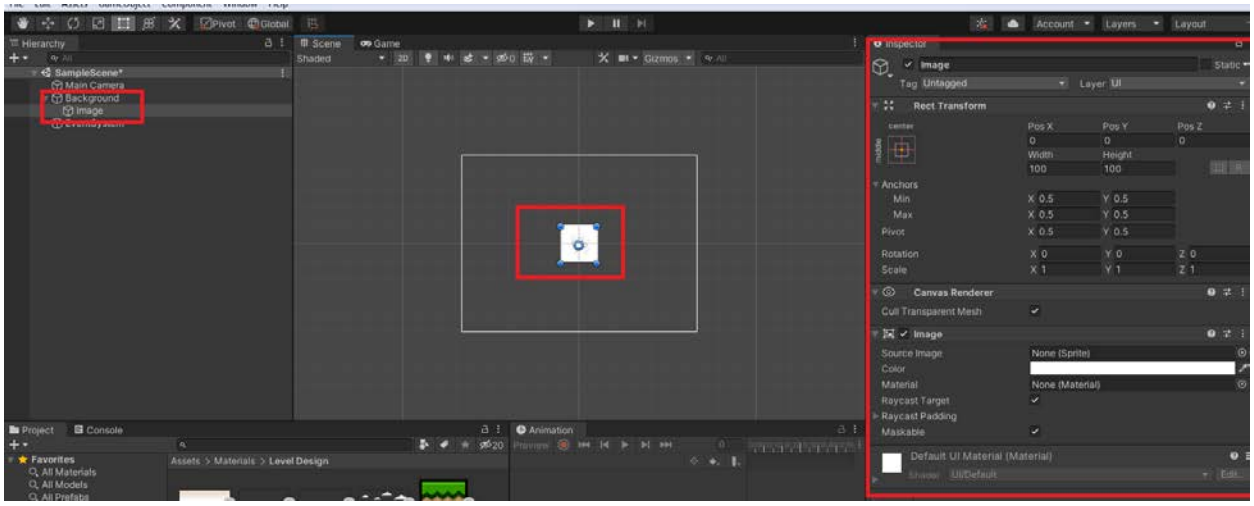




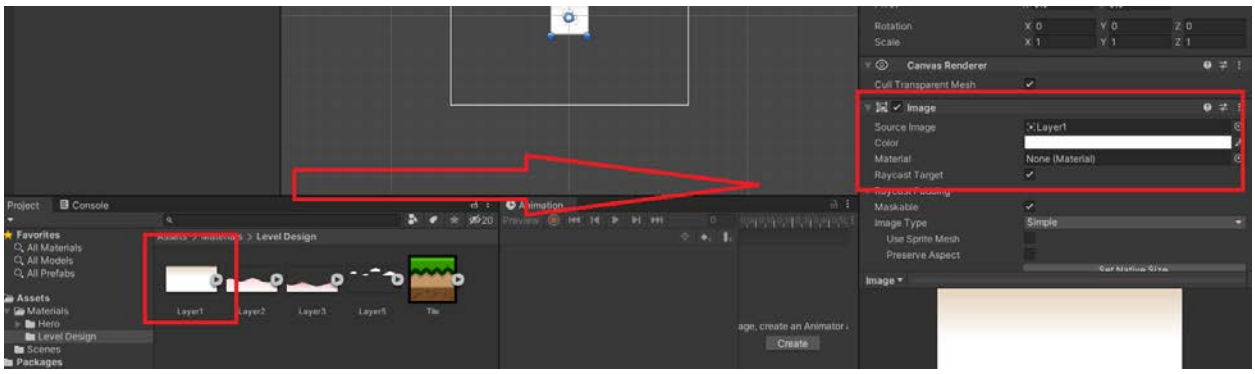
3.5. В Background – выбираем UI – Image



результат

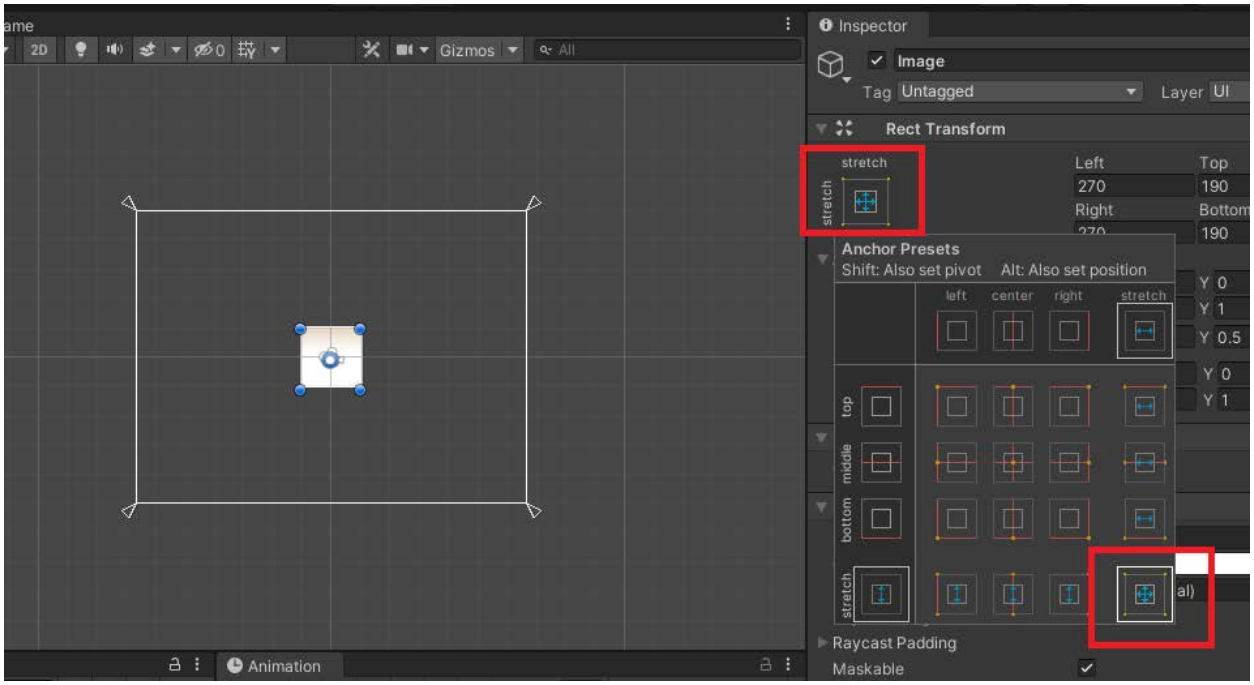


Перетаскиваем картинку Layer1



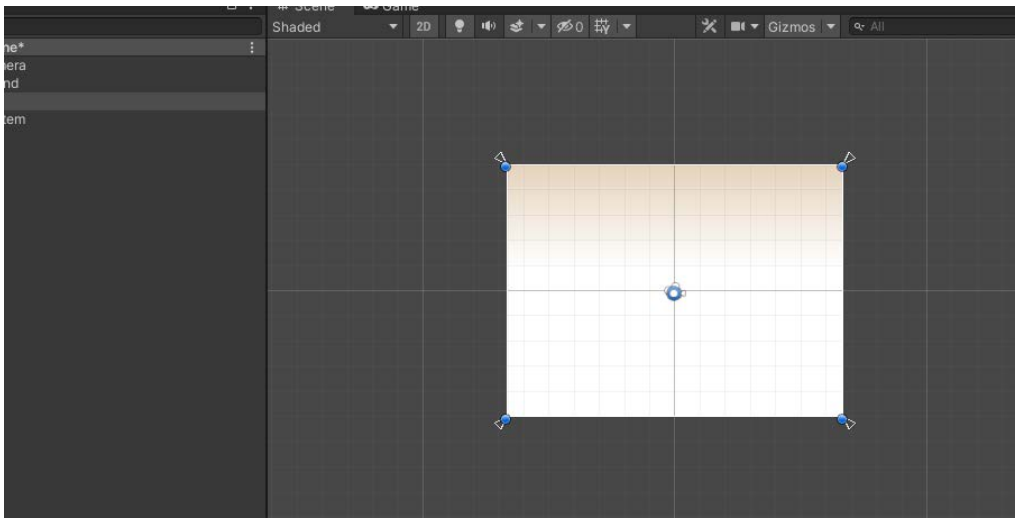
Увеличиваем размер картинки фона

Для этого в Inspector

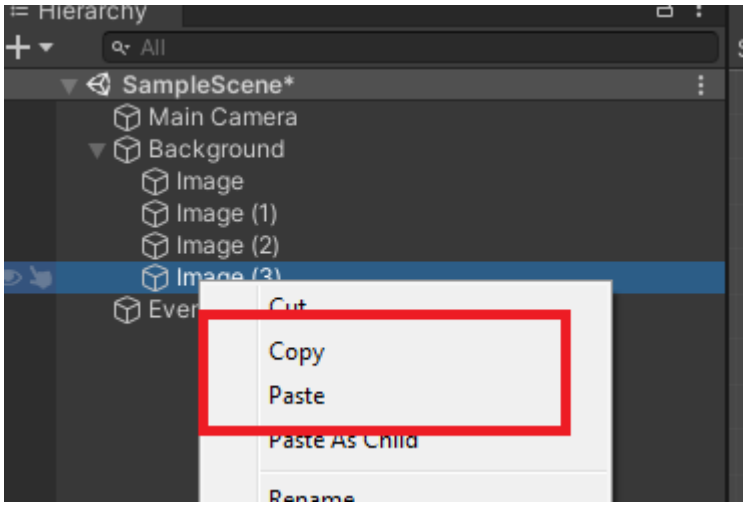


Alt и щелчок правой кнопкой мыши

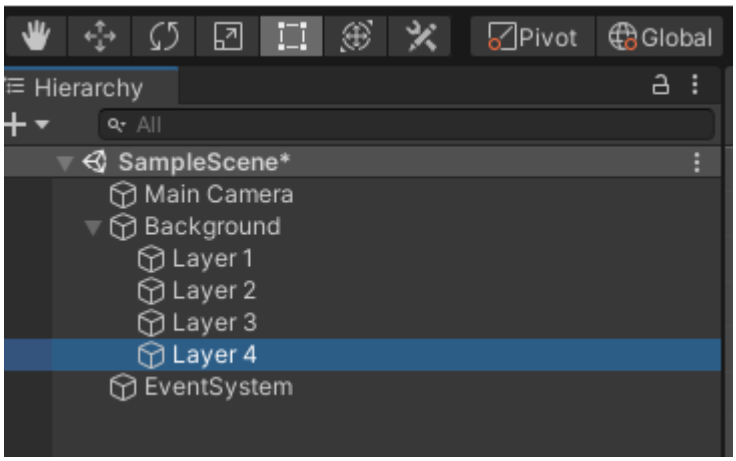
Результат



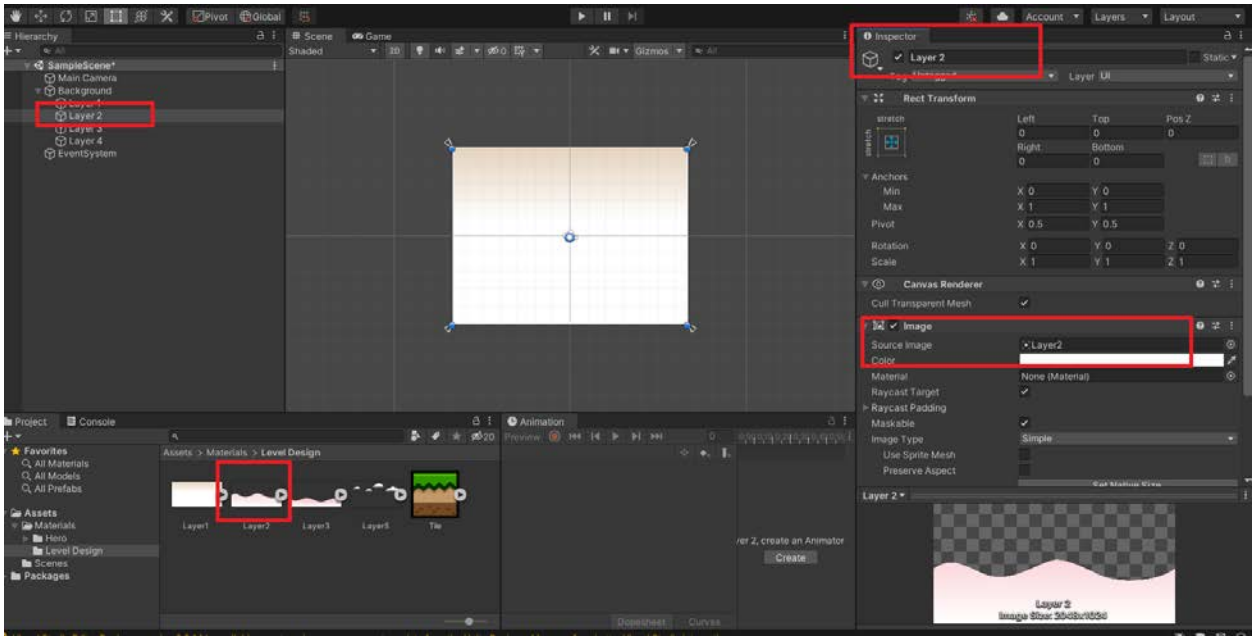
3.6. Создадим еще таких 3 слоя, воспользуемся копированием и вставкой



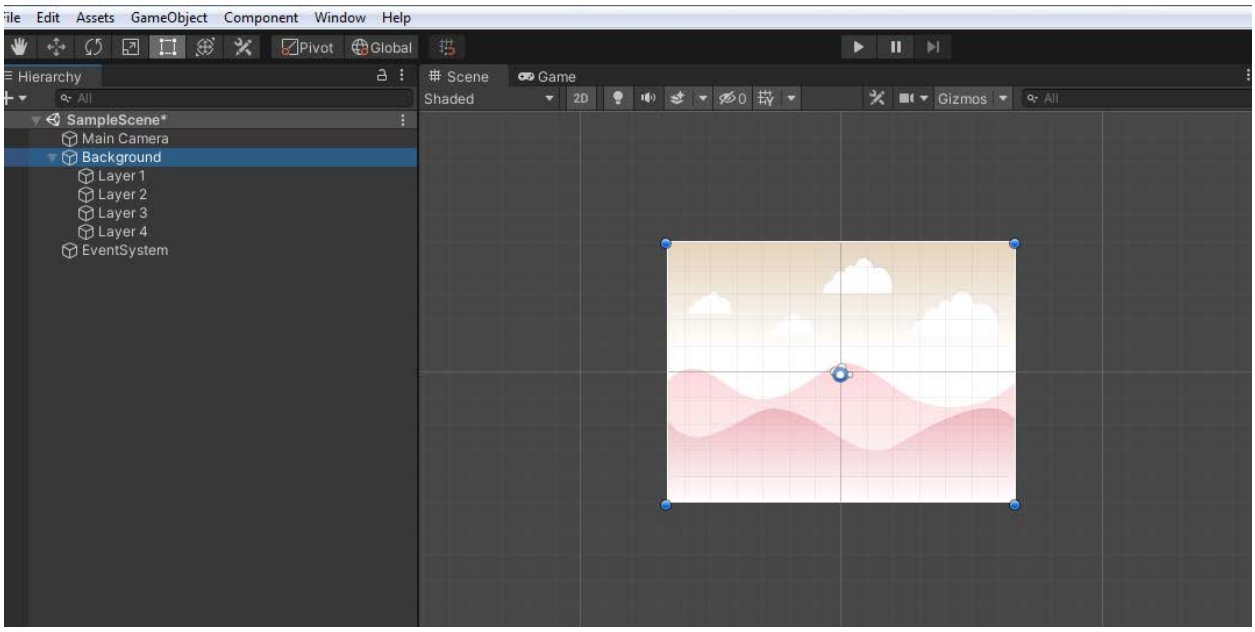
Переименуем, назовем Layer 1



и на каждый из этих объектов перетащим слой из папки Level Design

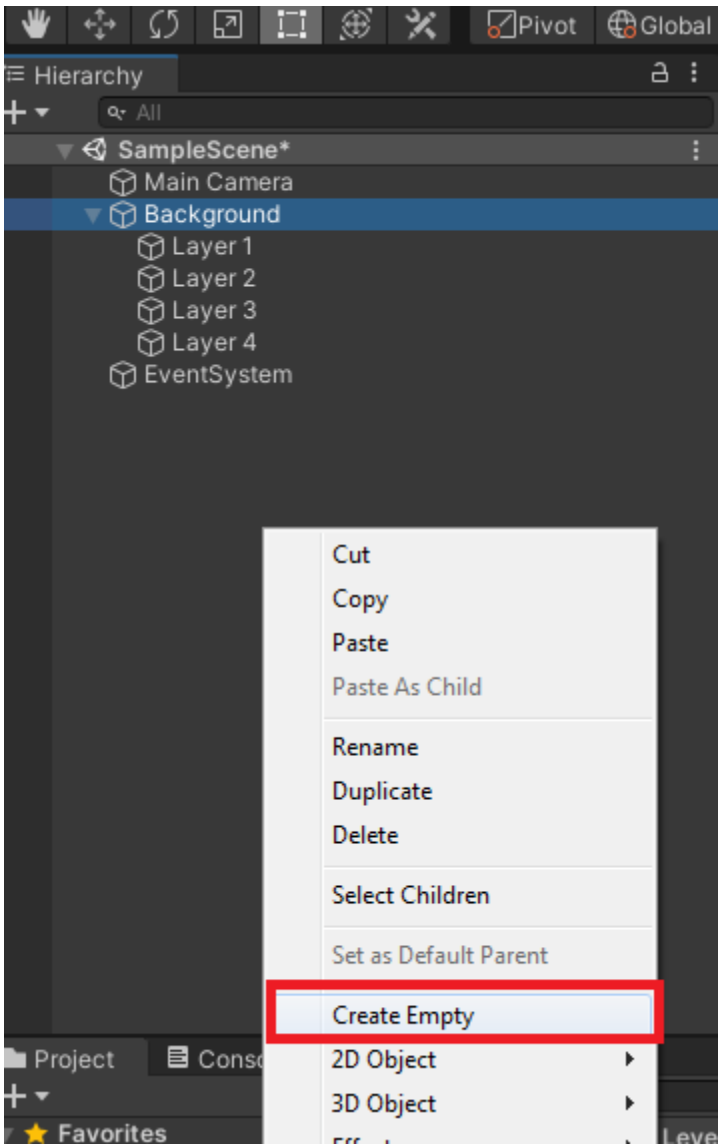


Результат – получившийся фон для уровня

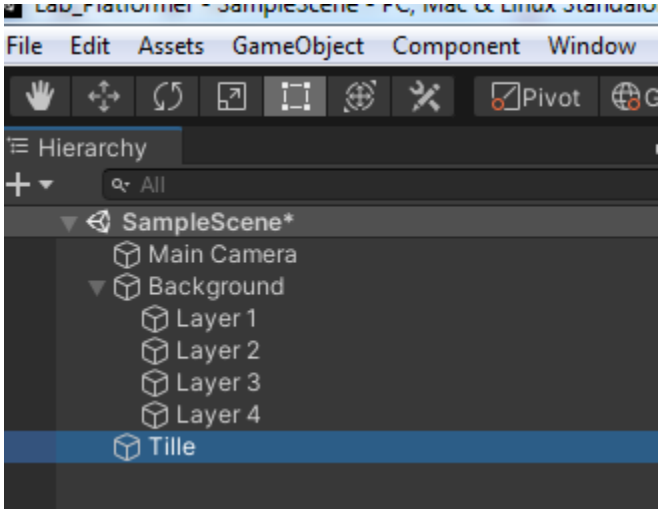


4. Создаем платформы

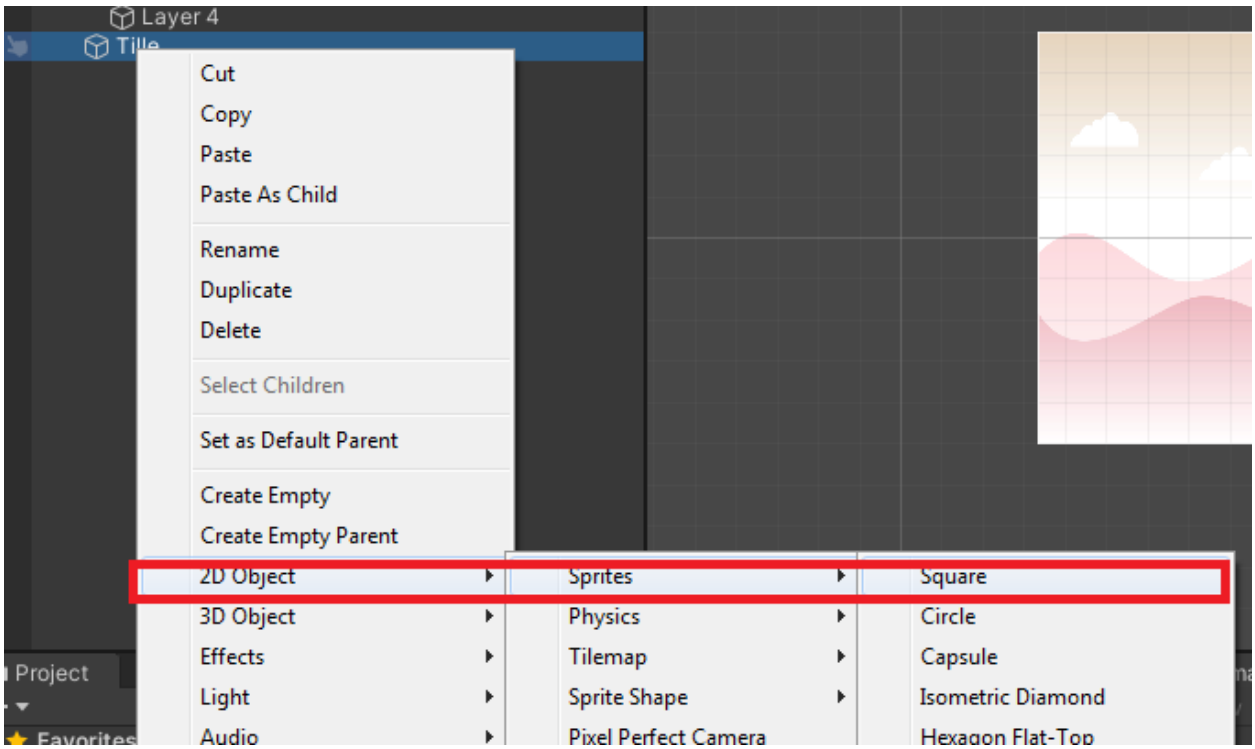
4.1. Создаем пустой Game Object



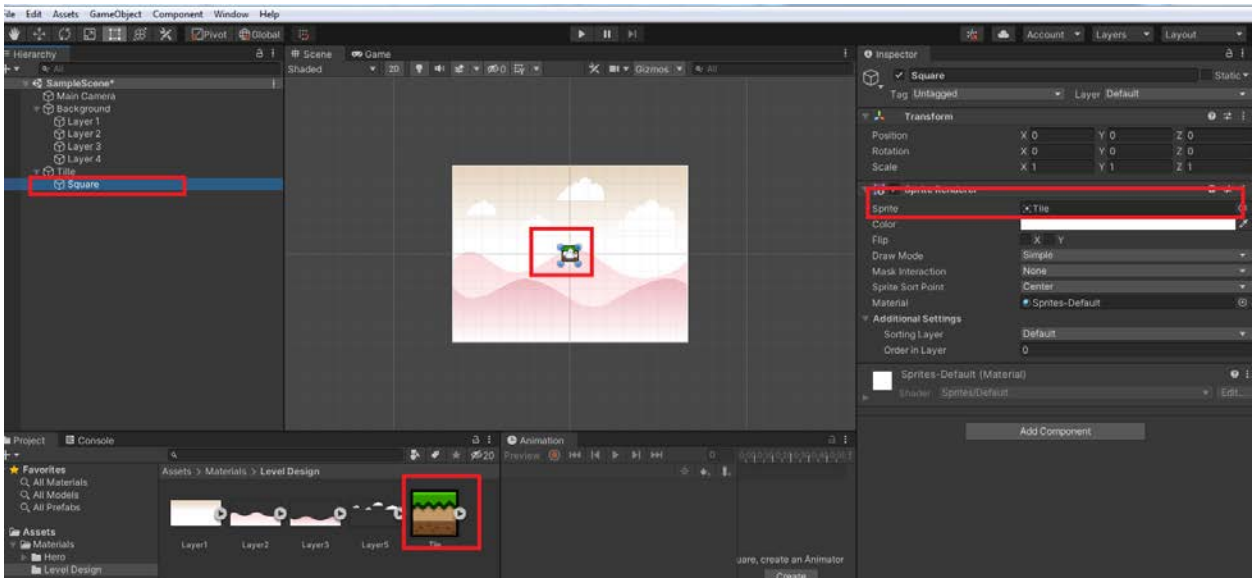
Переименуем в Tile



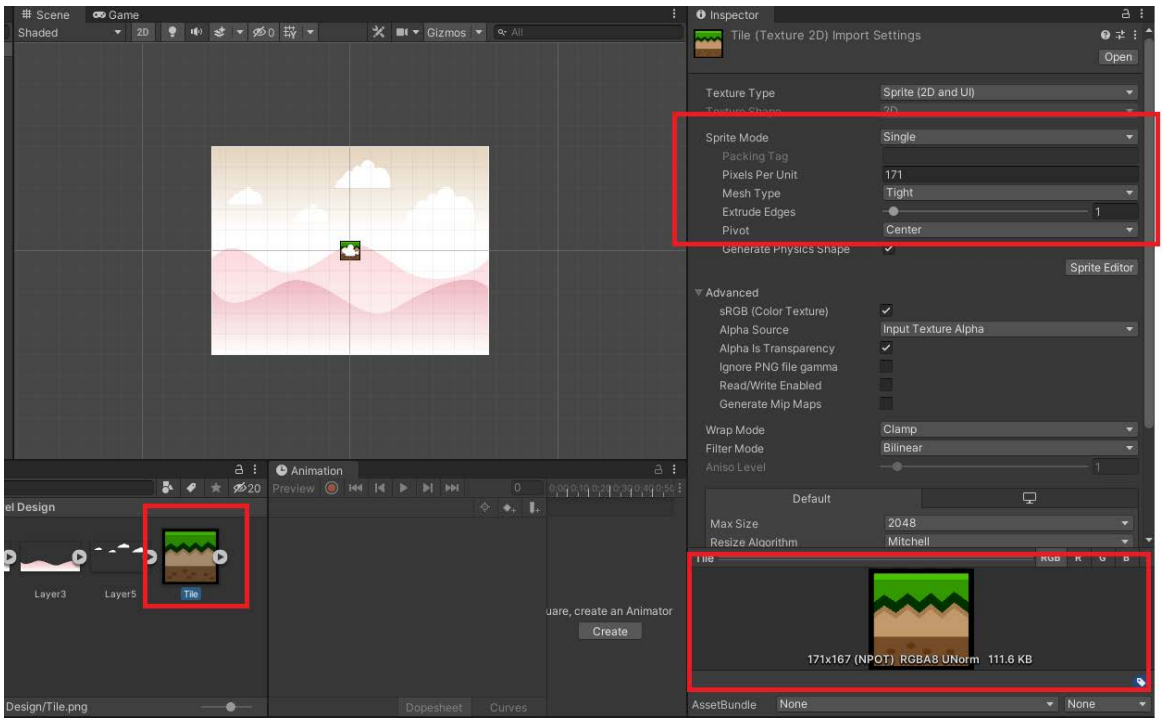
Добавляем спрайт



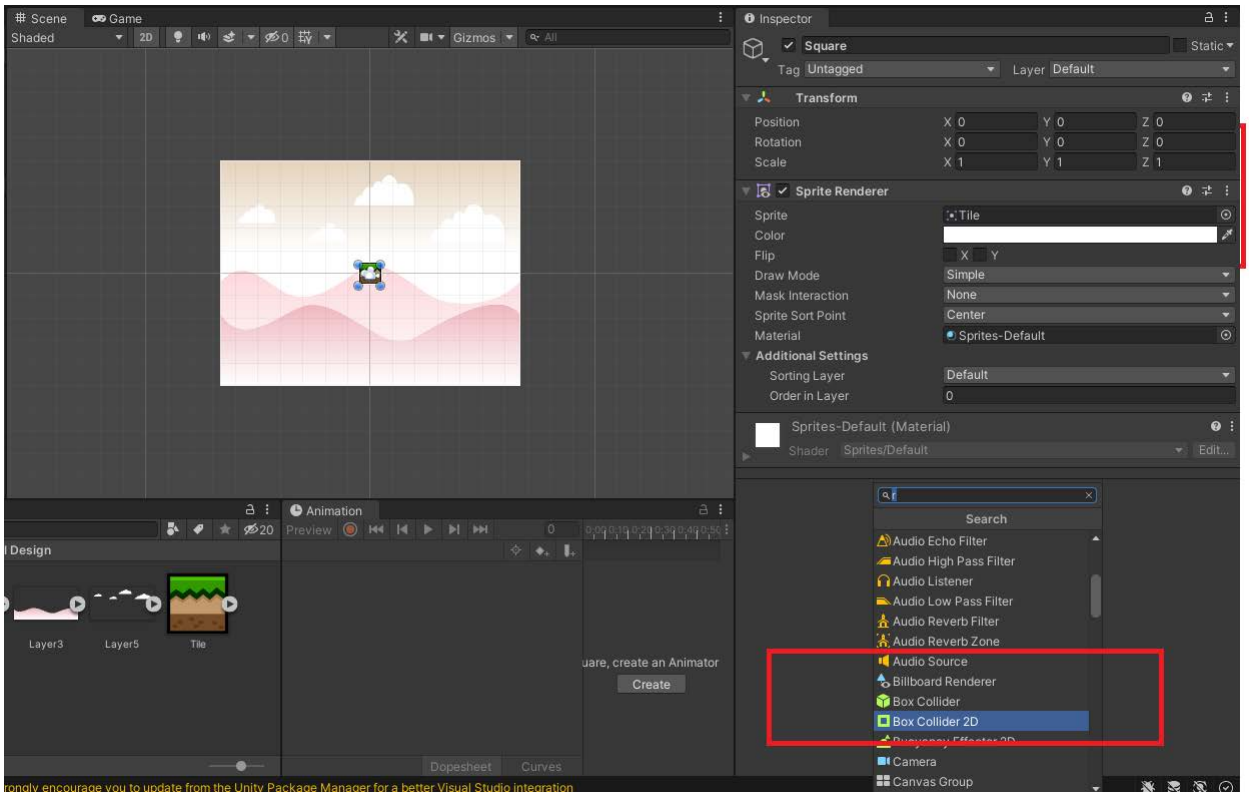
И перетаскиваем из папки изображение на спрайт

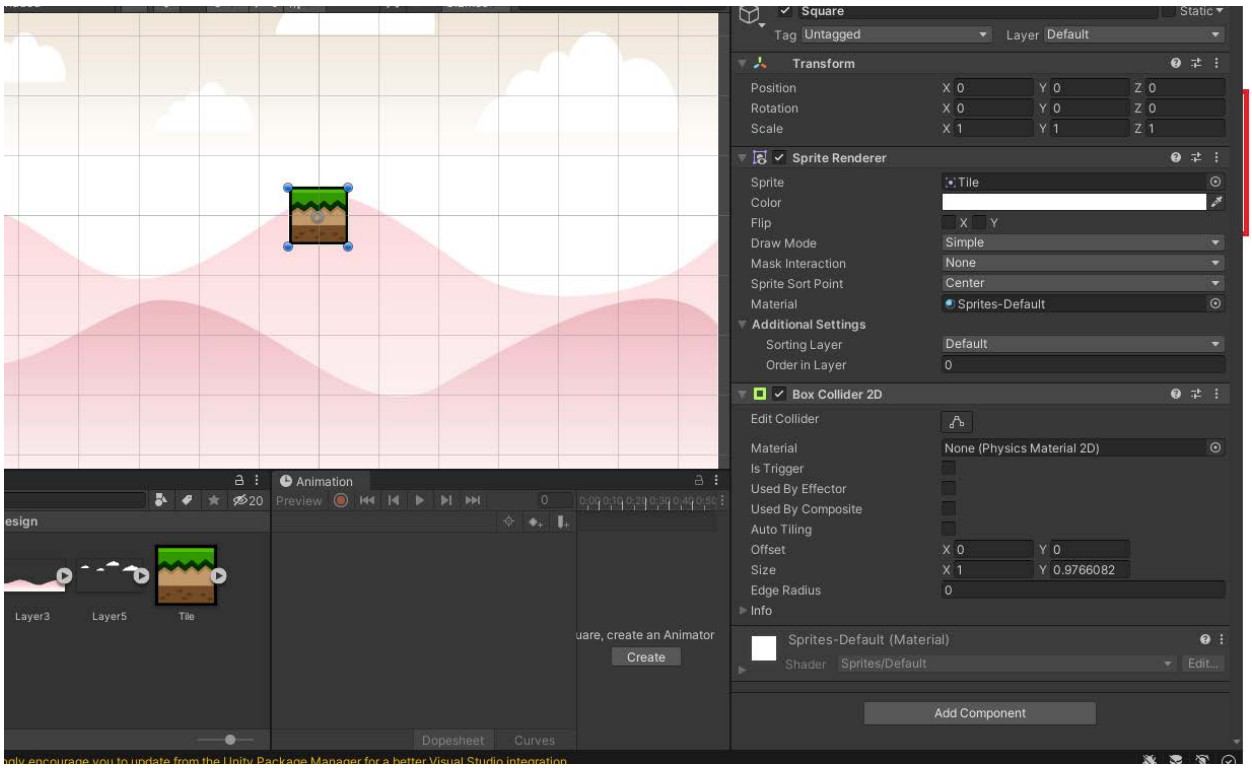


Размерность спрайта платформы можно установить в свойствах

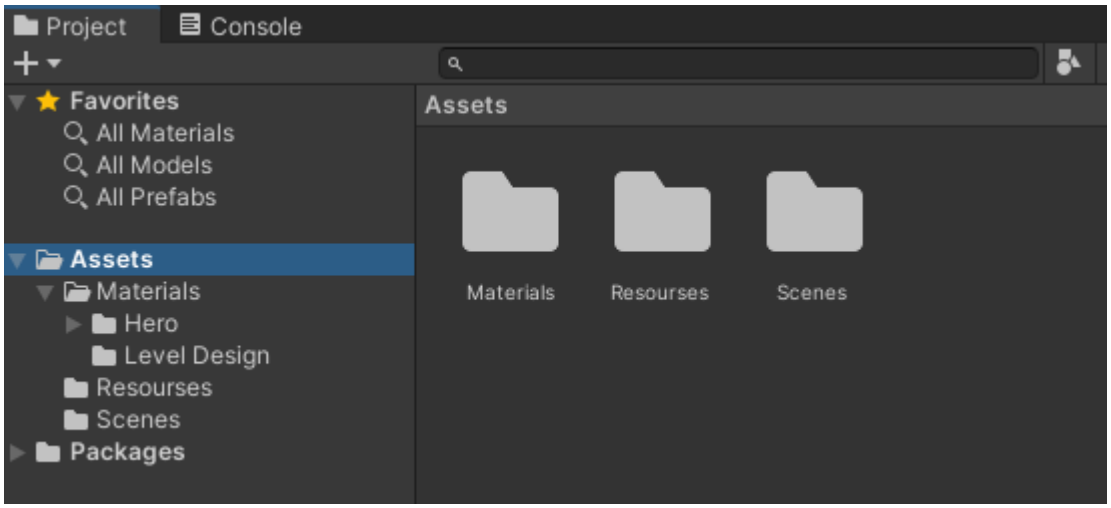


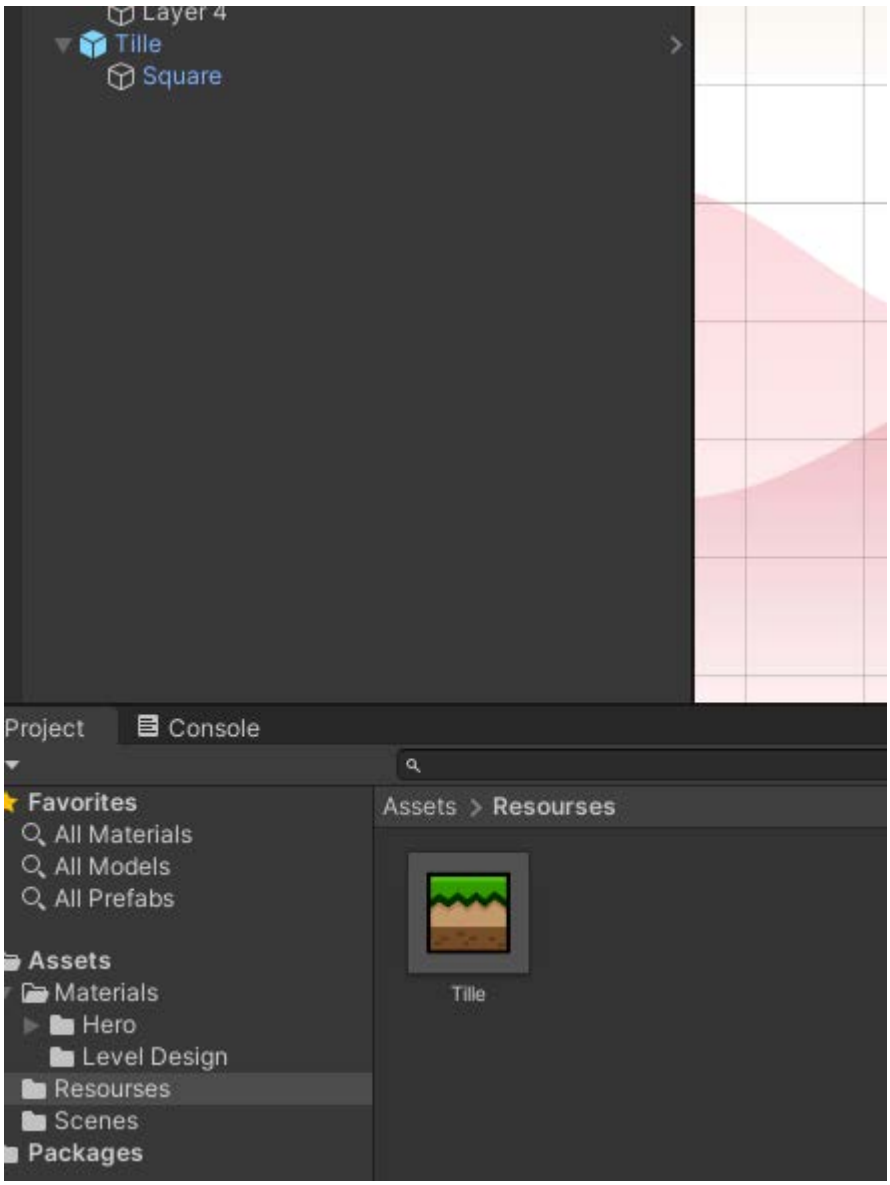
чтобы на платформе можно было стоять необходимо добавить компонент Box Collider 2D





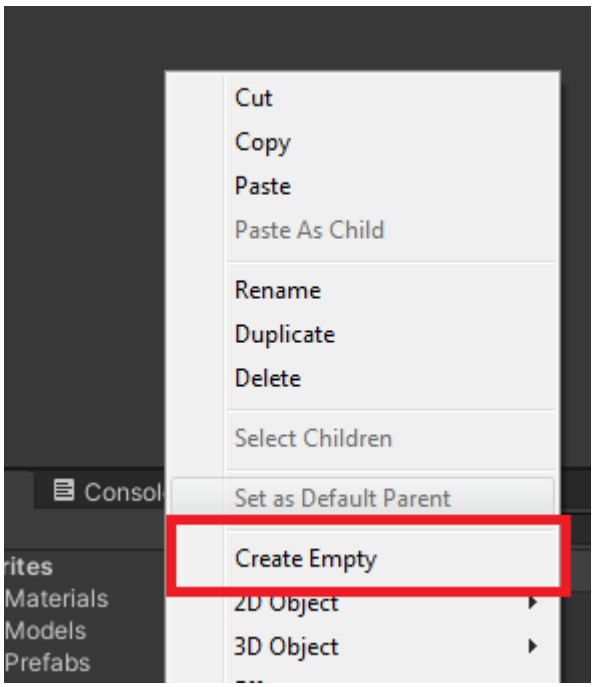
4.2. Создаем папку Resourses и туда перетаскиваем наш кубик





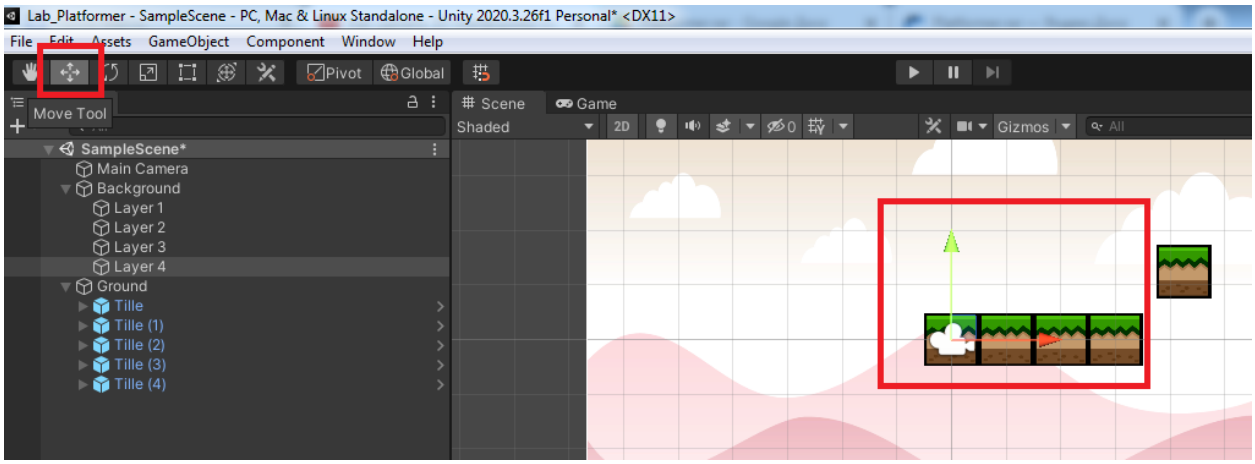
Получили префаб

4.3. Создаем уровень

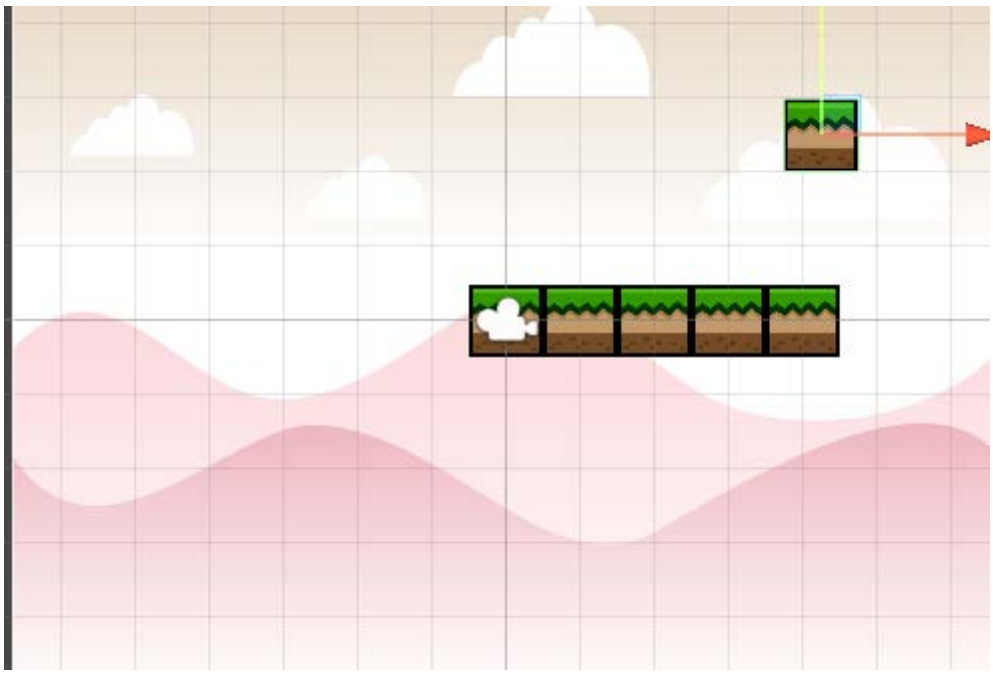


Переименовываем и перетаскиваем Tile

Воспользуемся Move Tool чтобы перемещать по сетке

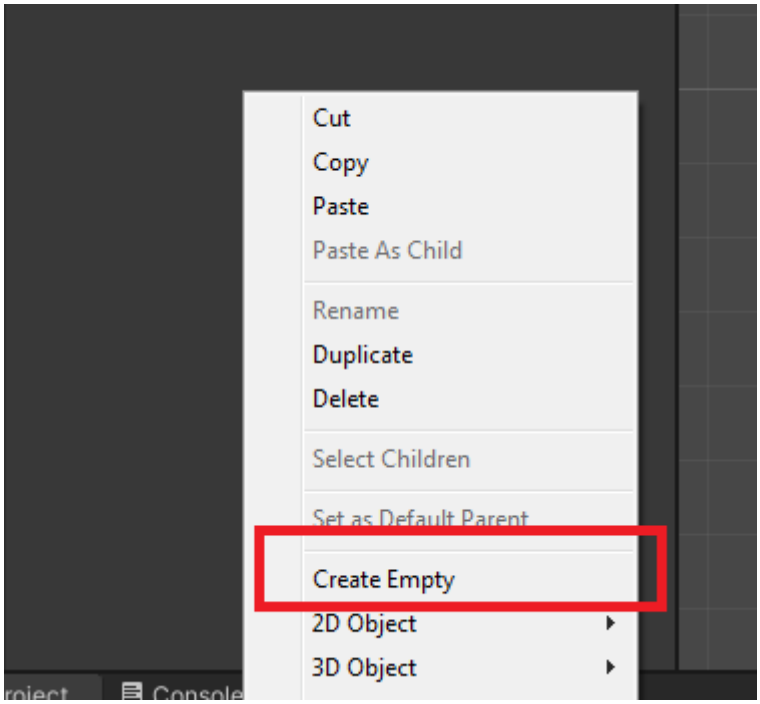


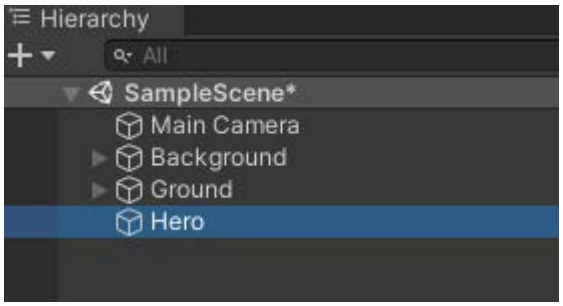
Создадим платформы



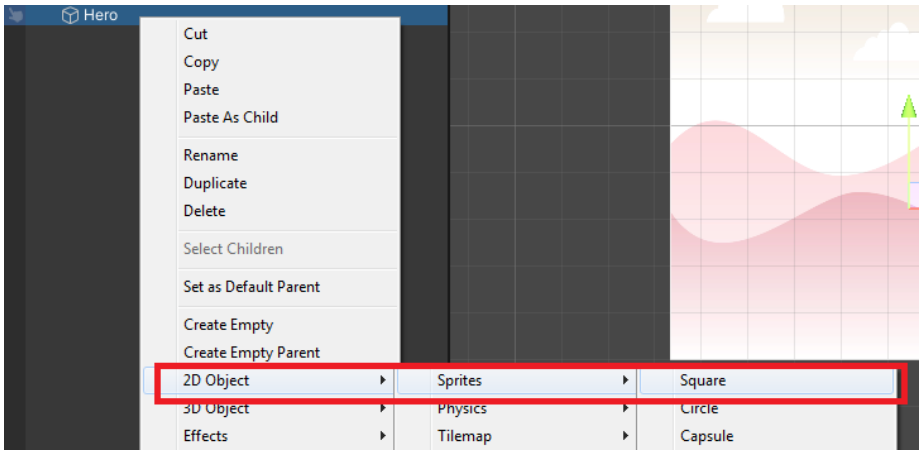
5. Создаем персонаж.

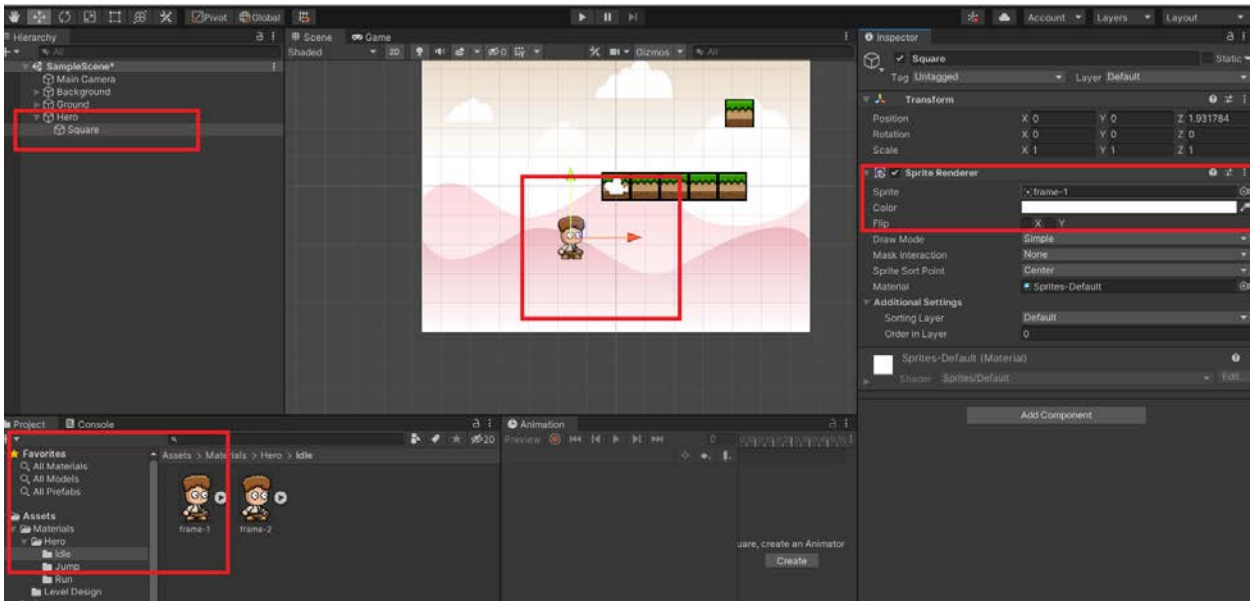
Пустой объект. Переименовываем



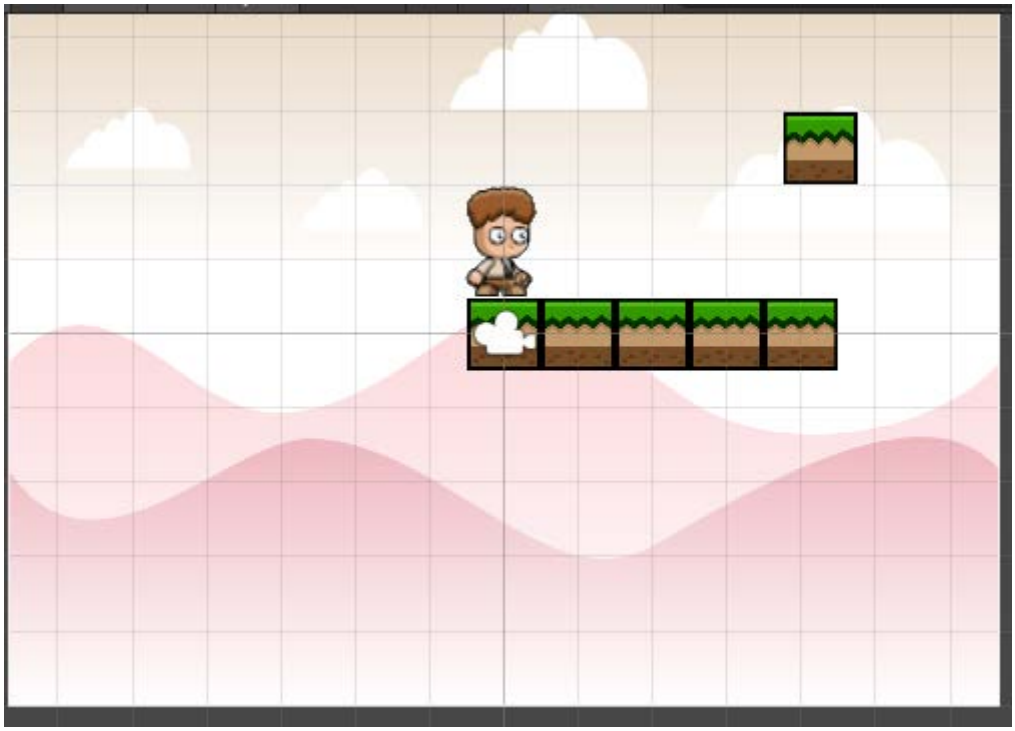


Создаем спрайт и из папки Hero – Idle (состояние покоя) перетаскиваем спрайт

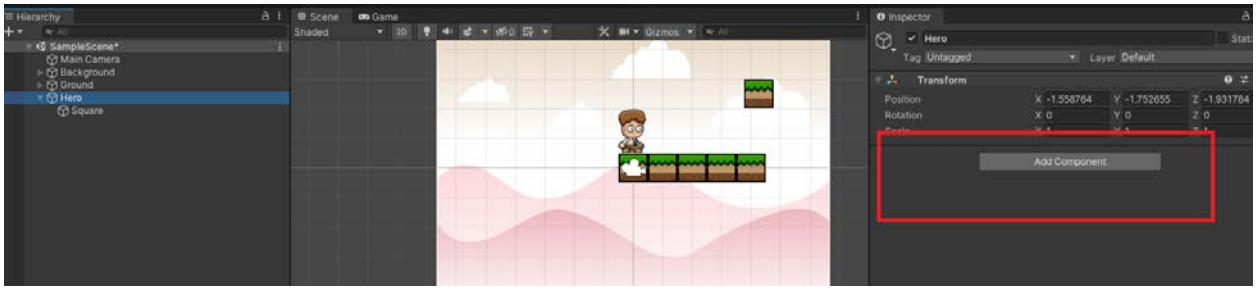


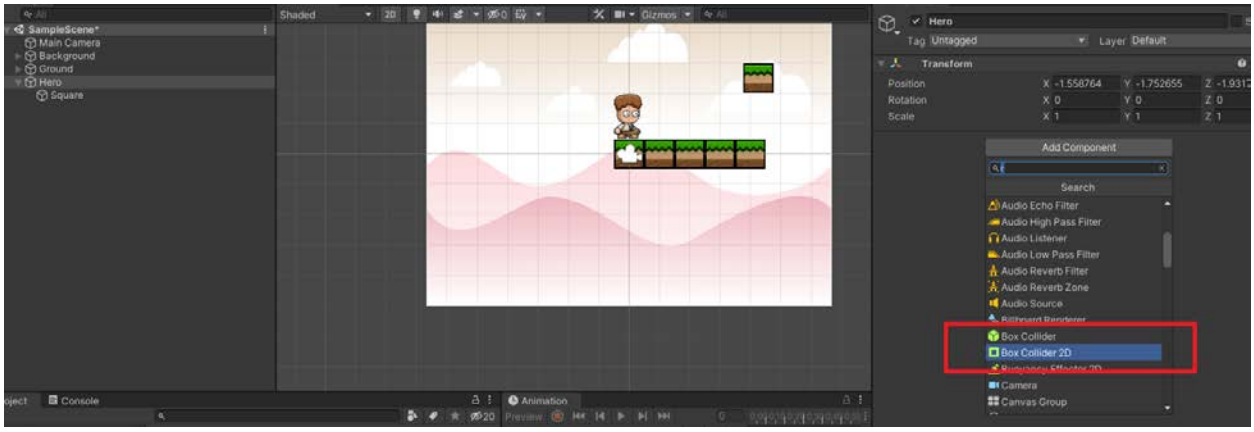


Изменяем положение героя как нам нравится

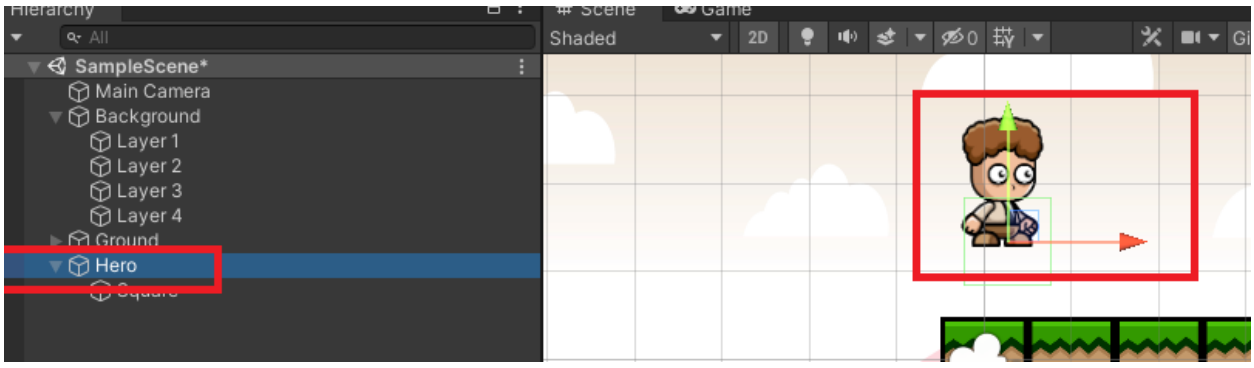


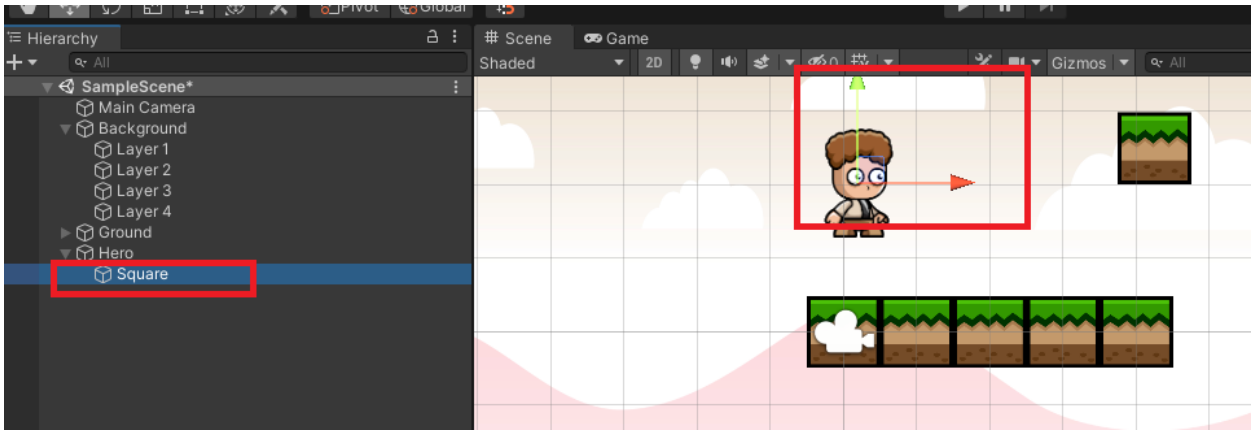
Добавляем на него коллайдер



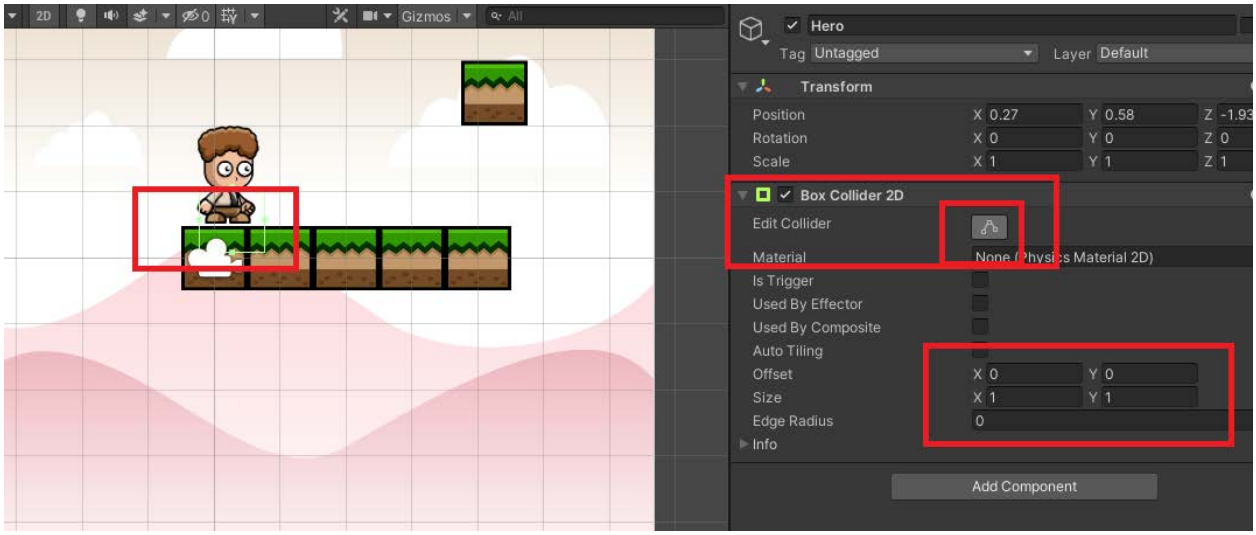


Для удобства работы с прыжками надо немного сдвинуть систему коллайдера Hero и Спрайта

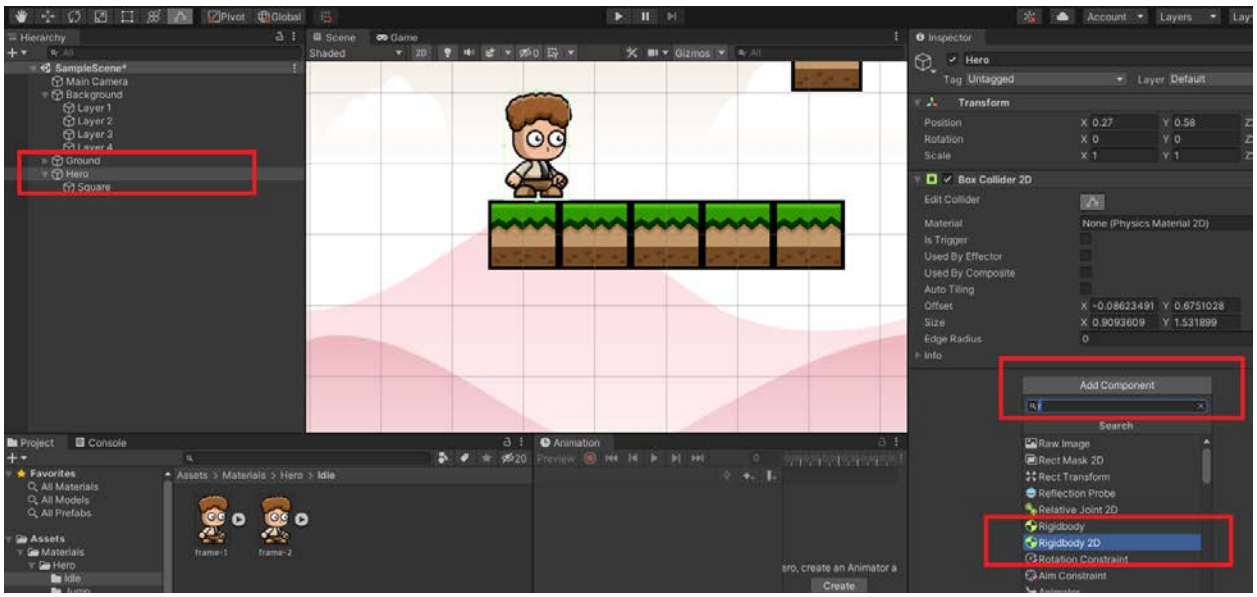




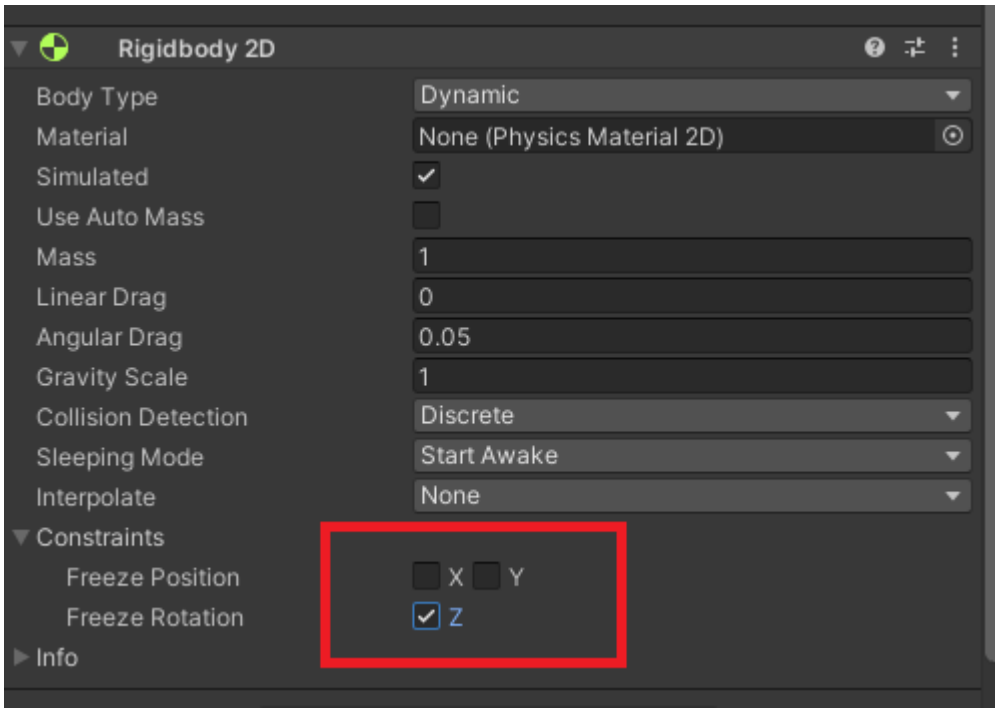
Отредактируем коллайдер у персонажа



Добавляем

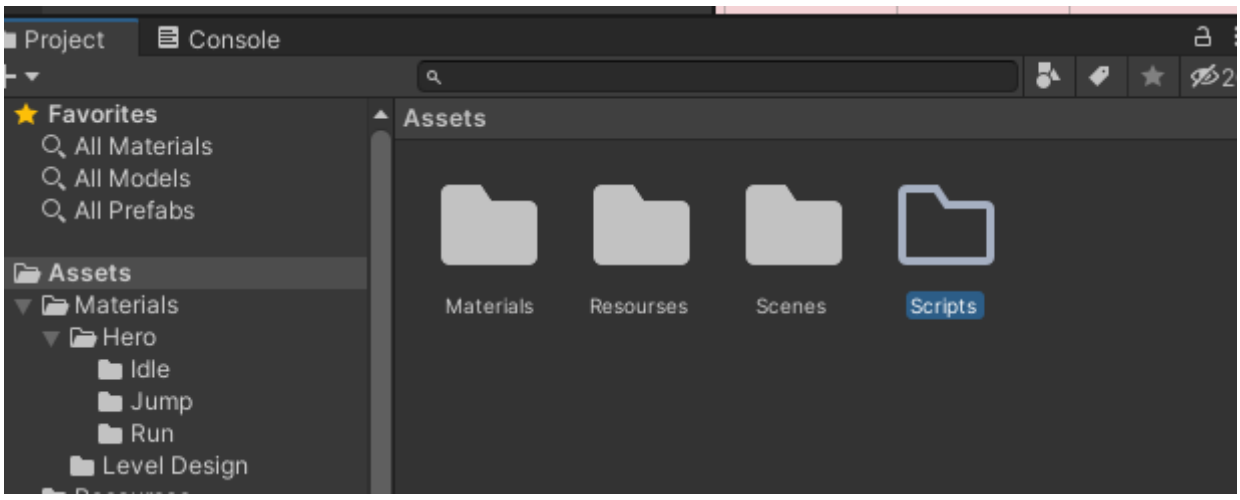


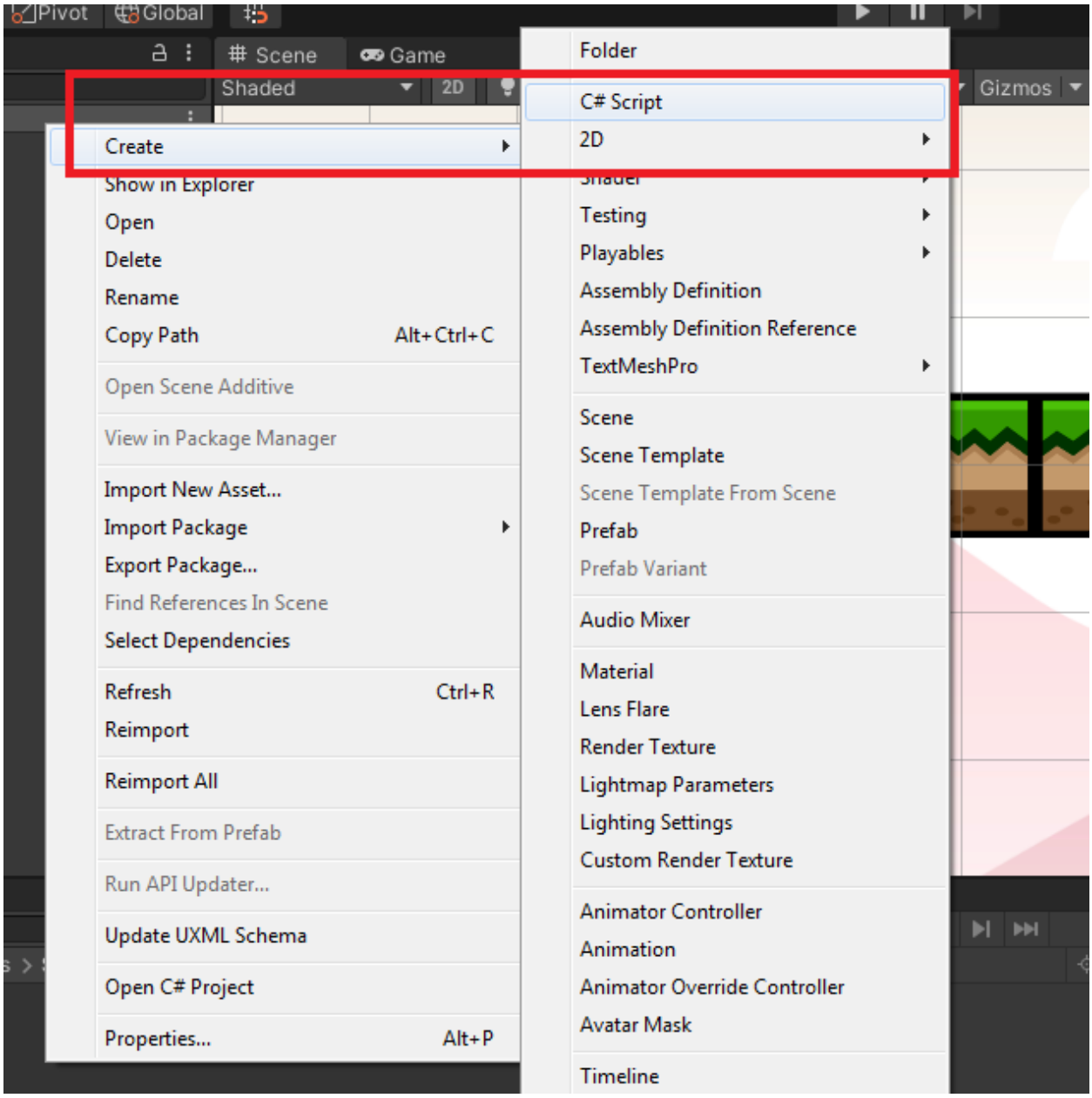
Заморозим движение по оси Z

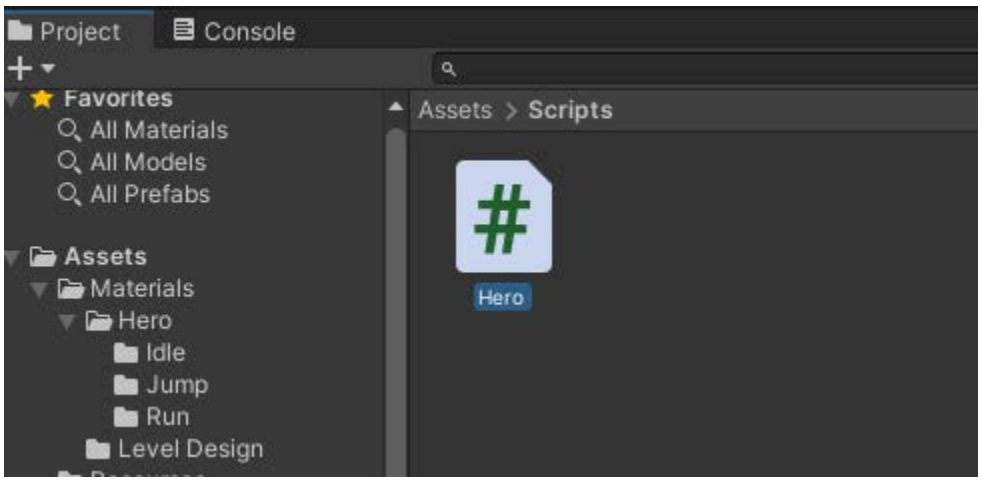


Можно настроить гравитацию , массу

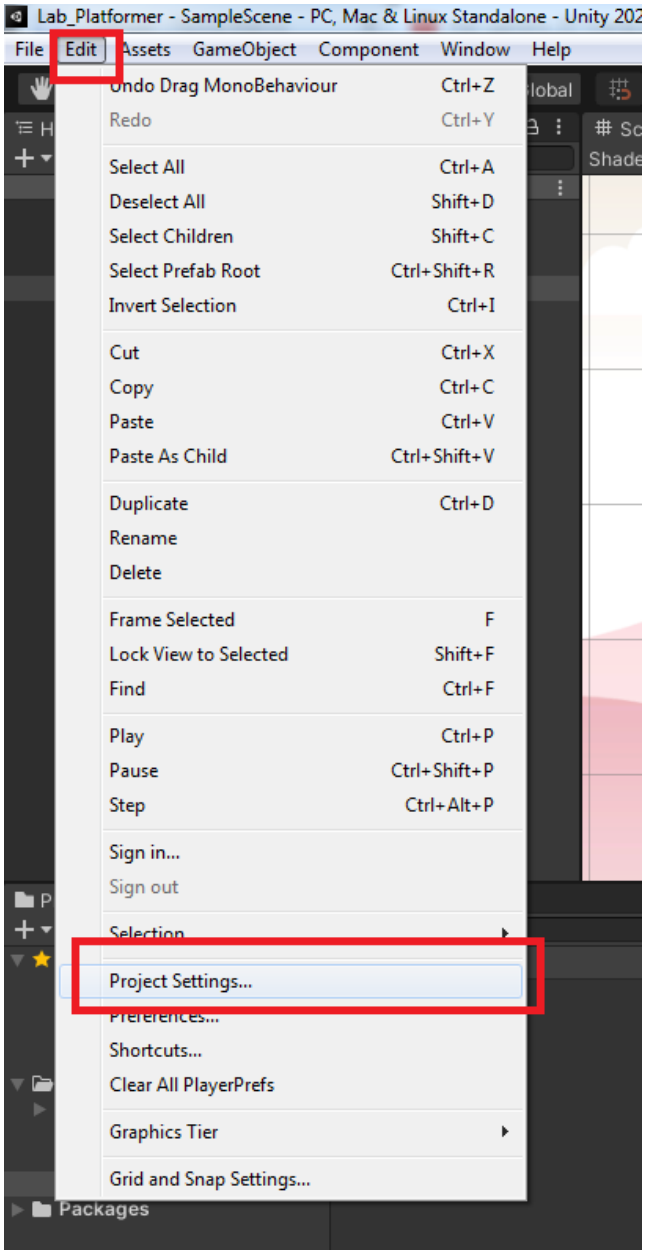
6. Создаем папку Script и в ней создаем скрипт

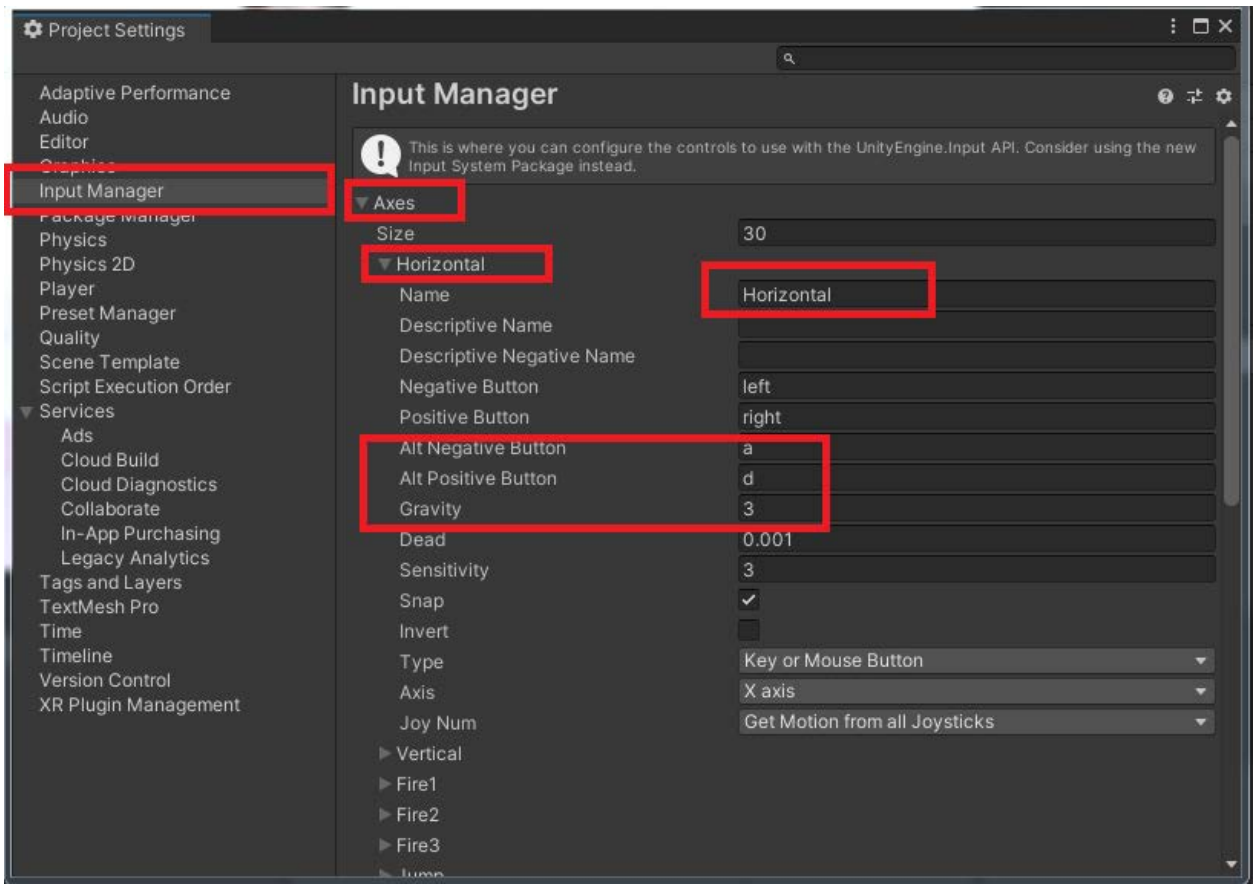

Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Materials
Hero
Idle
Jump
Run
Level Design
Assets
Materials
Resourses
Scenes
Scripts


Pivot
Global
Scene
Game
Shaded
2D
Gizmos
Create
Show in Explorer
Open
Delete
Rename
Copy Path Alt+Ctrl+C
Open Scene Additive
View in Package Manager
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Extract From Prefab
Run API Updater...
Update UXML Schema
Open C# Project
Properties... Alt+P
Folder
C# Script
2D
Testing
Playables
Assembly Definition
Assembly Definition Reference
TextMeshPro
Scene
Scene Template
Scene Template From Scene
Prefab
Prefab Variant
Audio Mixer
Material
Lens Flare
Render Texture
Lightmap Parameters
Lighting Settings
Custom Render Texture
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Timeline



В скрипте будем использовать настройки для движения, которые можно посмотреть




Project Settings
Adaptive Performance
Audio
Editor
Input Manager
Physics
Physics 2D
Player
Preset Manager
Quality
Scene Template
Script Execution Order
Services
Ads
Cloud Build
Cloud Diagnostics
Collaborate
In-App Purchasing
Legacy Analytics
Tags and Layers
TextMesh Pro
Time
Timeline
Version Control
XR Plugin Management
Input Manager
This is where you can configure the controls to use with the UnityEngine.Input API. Consider using the new Input System Package instead.
Axes
Size
30
Horizontal
Name
Horizontal
Descriptive Name
Descriptive Negative Name
Negative Button
left
Positive Button
right
Alt Negative Button
a
Alt Positive Button
d
Gravity
3
Dead
0.001
Sensitivity
3
Snap
Invert
Type
Key or Mouse Button
Axis
X axis
Joy Num
Get Motion from all Joysticks
Vertical
Fire1
Fire2
Fire3

Скрипт

Hero.cs

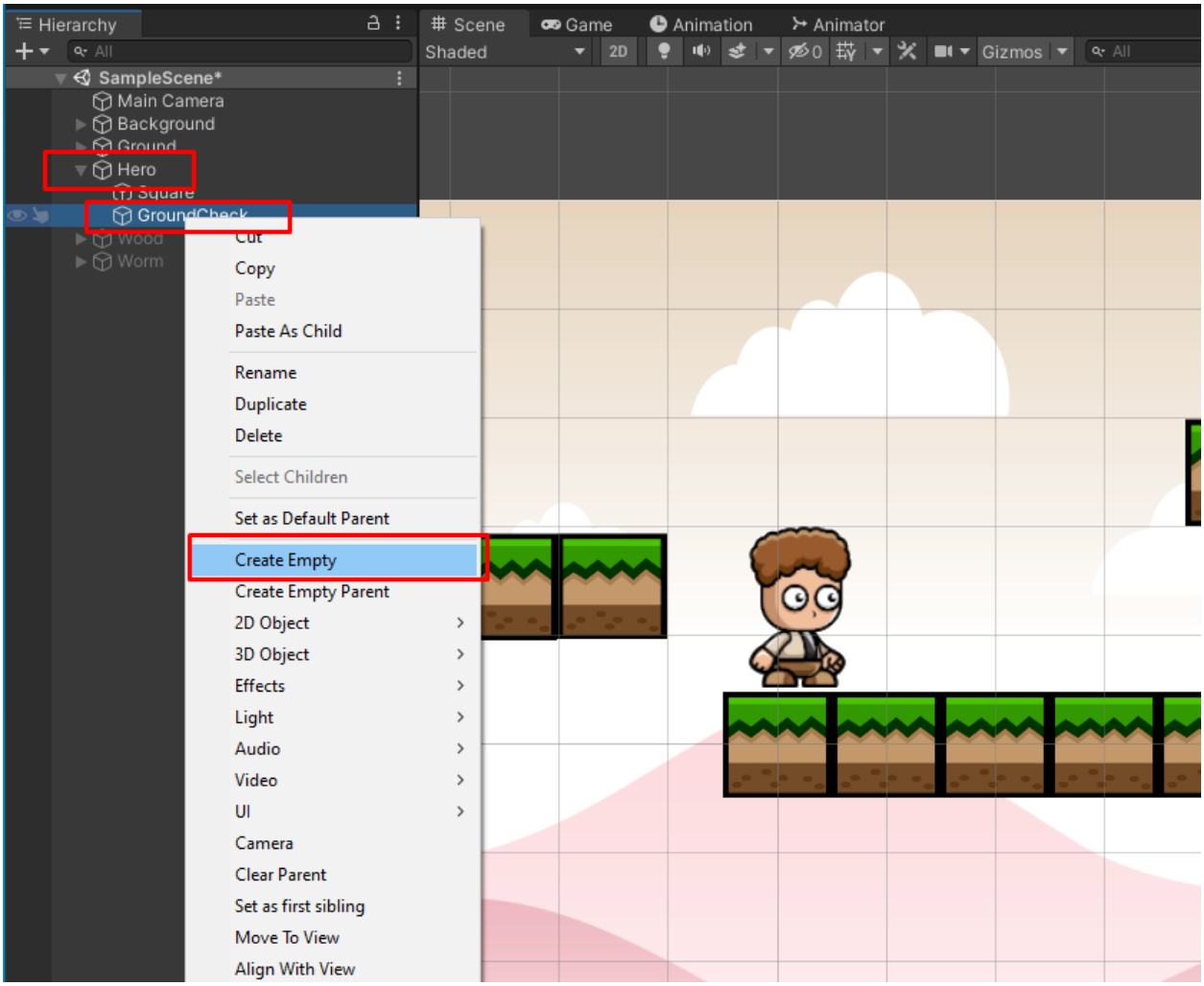
Assembly-CSharp | Hero

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
    Скрипт Unity | Ссылок: 0
5   public class Hero : MonoBehaviour
6   {
7       [SerializeField] private float speed = 3f; // скорость
8       [SerializeField] private int lives = 5; // количество жизни
9       [SerializeField] private float jumpForce = 15f; // сила прыжка
10
11      private bool isGrounded = false;
12
13
14      private Rigidbody2D rb;
15      private SpriteRenderer sprite;
16
        Сообщение Unity | Ссылок: 0
17      private void Awake()
18      {
19          sprite = GetComponentInChildren<SpriteRenderer>(); //обращение к спрайту
20          rb = GetComponent<Rigidbody2D>();
21      }
22
        ссылка: 1
23      private void Run()
24      {
25          Vector3 dir = transform.right * Input.GetAxis("Horizontal");
26          transform.position = Vector3.MoveTowards(transform.position, transform.position + dir, speed * Time.deltaTime);
27          sprite.flipX = dir.x < 0.0f; // поворот глаз
28      }
        Сообщение Unity | Ссылок: 0
29      private void Update()
30      {
31          if (Input.GetButton("Horizontal"))
32              Run();
33          if (isGrounded && Input.GetButtonDown("Jump"))
34              Jump();
35      }
        Сообщение Unity | Ссылок: 0
36      private void FixedUpdate()
37      {
38          CheckGround();
39      }
40
        ссылка: 1
41      private void Jump()
42      {
43          rb.AddForce(transform.up * jumpForce, ForceMode2D.Impulse);
44      }
45
        ссылка: 1
46      private void CheckGround()
47      {
48          Collider2D[] collider = Physics2D.OverlapCircleAll(transform.position, 0.3f);
49          isGrounded = collider.Length > 1;
50      }
51  }
52
```

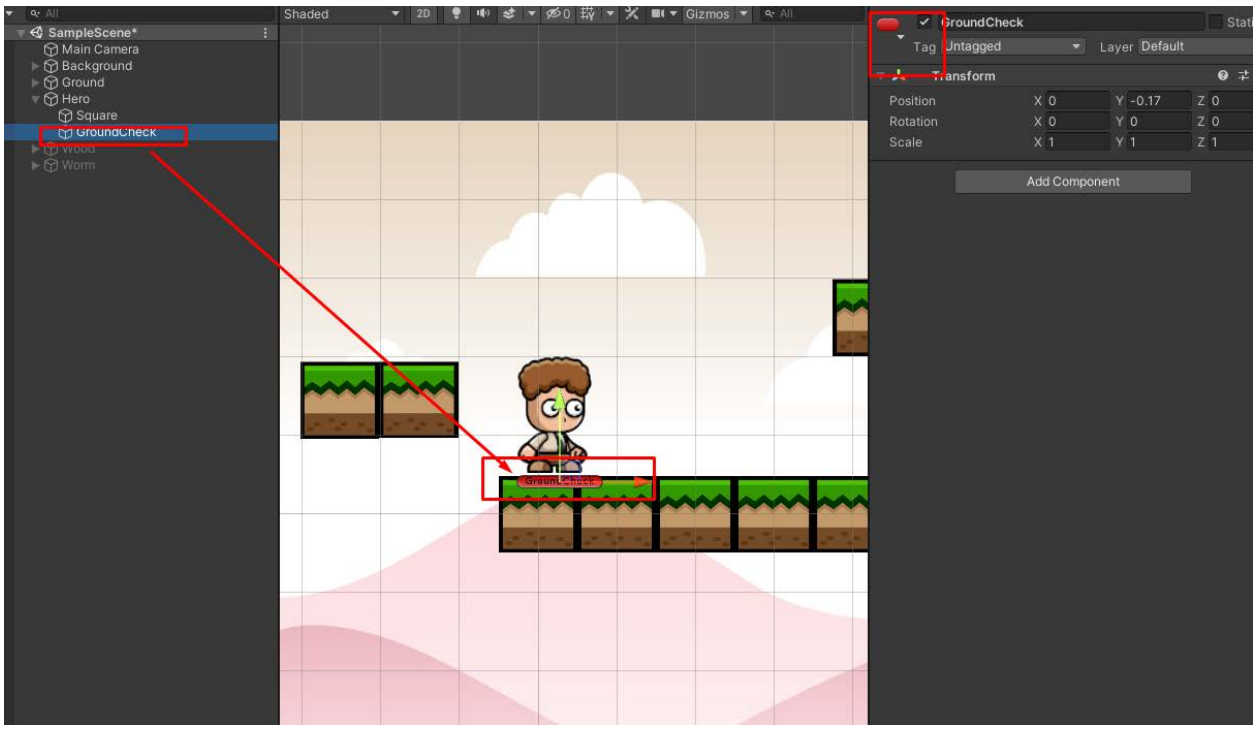
Прыжок и движение. Второй способ

Есть очень много способов написать прыжок персонажу (триггер, raycast, таймер и т.д.). Вторым способом реализуем через *Physics2D.OverlapPoint* , для этого нужно определить, какой слой находится под контрольной точкой.

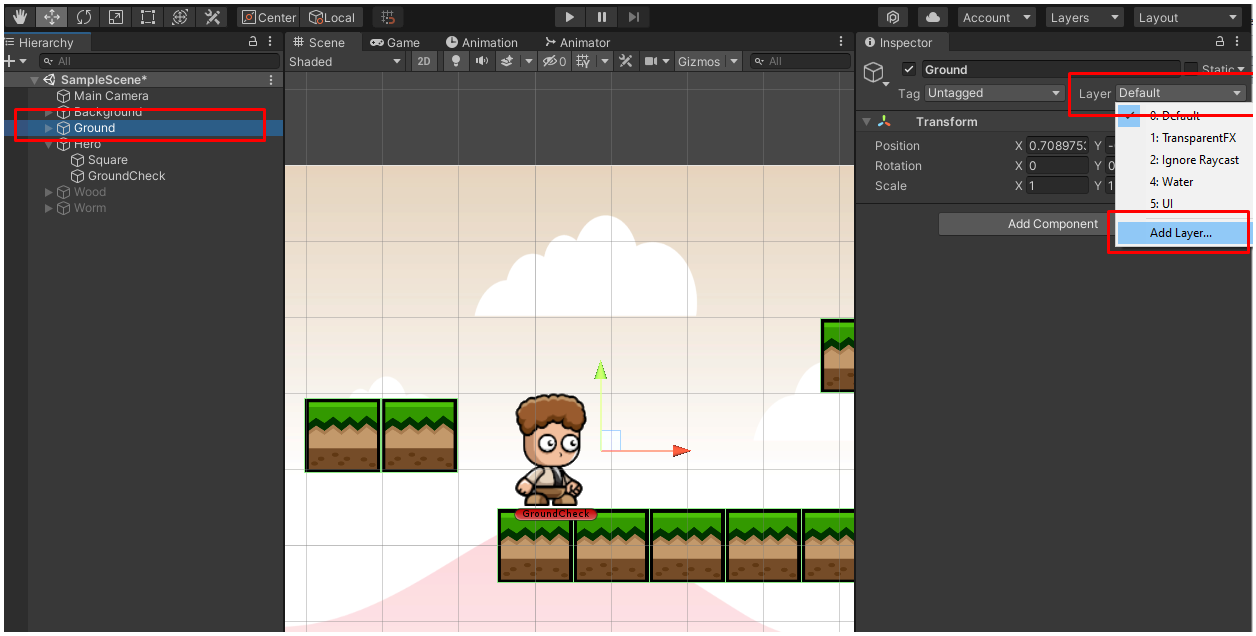
Выберите Hero в иерархии, щелкните по нему правой кнопкой мыши и выберите Create Empty . Выберите созданный объект и переименуйте его в GroundCheck .

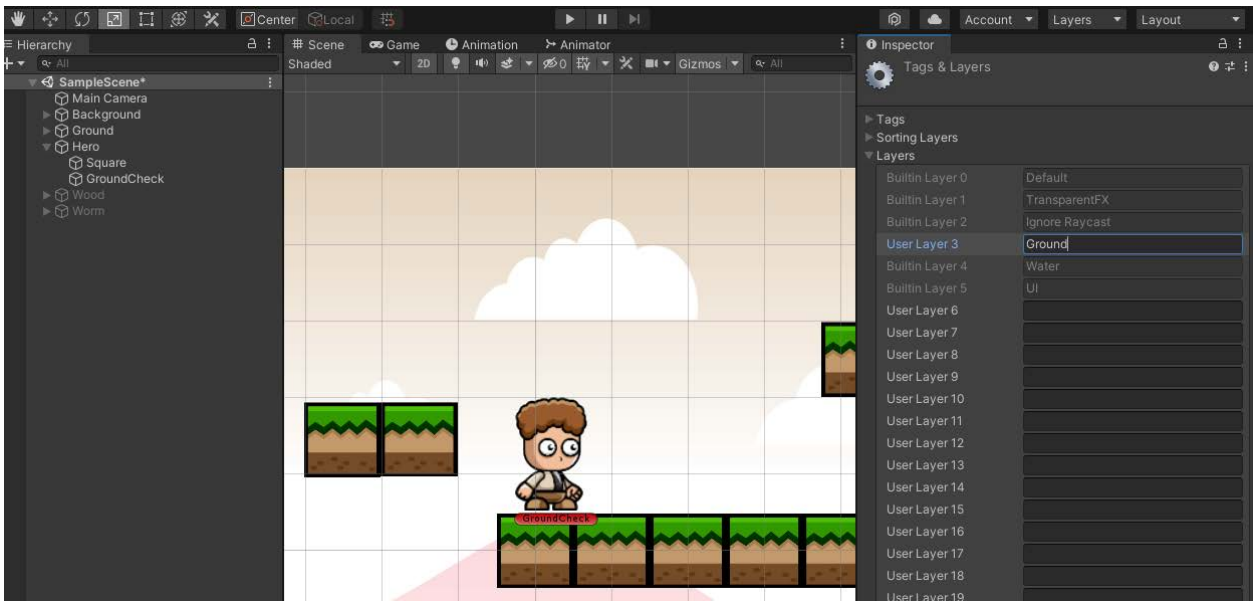


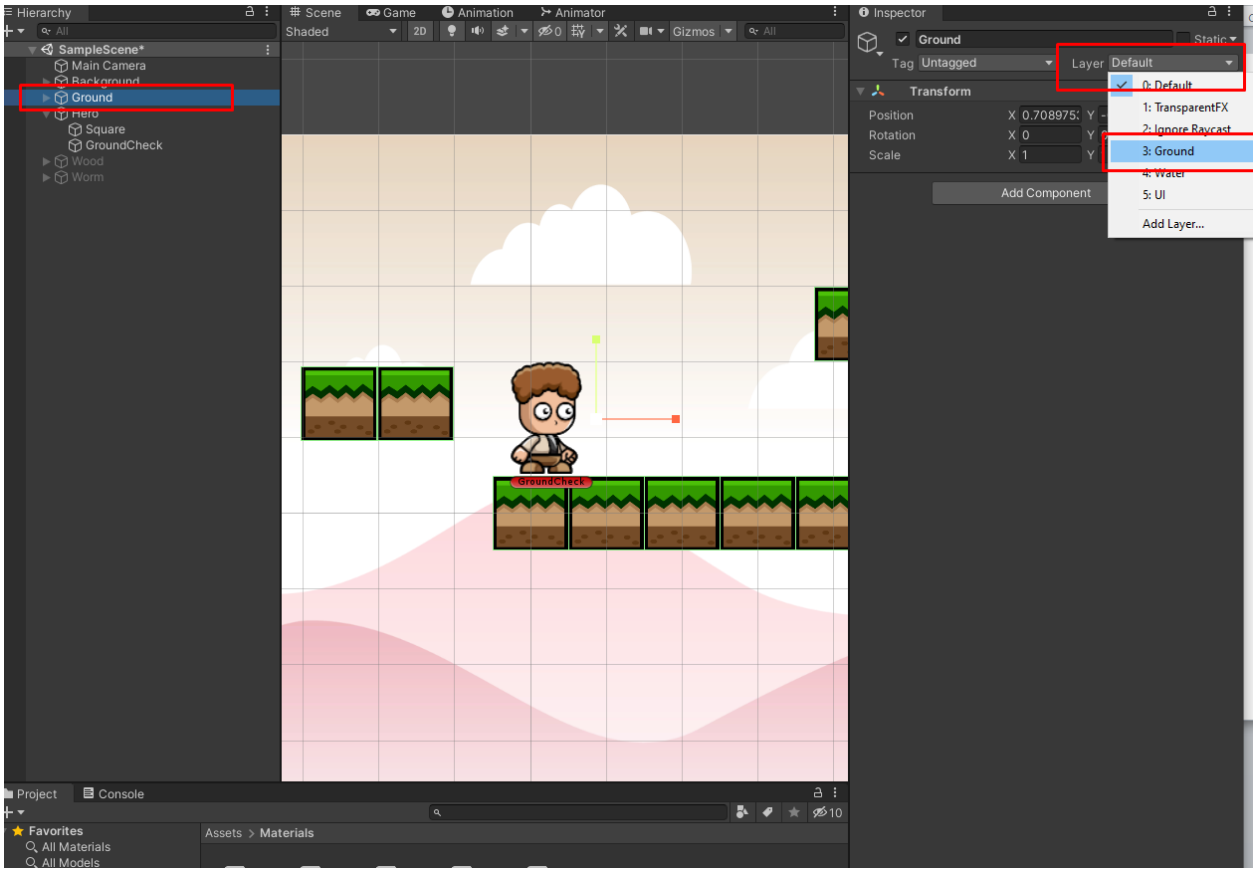
Выбираем GroundCheck и включаем иконку для лучшей видимости (смотрите скриншот). С помощью moving tool и передвигаем GroundCheck под героя, но не в сам коллайдер героя. Убедитесь, что его Z-Position равна 0.



Выбераем Ground в иерархии и меняем Layer на Ground (добавьте новый Layer с этим названием, если в списке нет такого).







Изменяем скрипт.

1. Добавляем переменные

```
//в инспекторе мы можем выбрать, какие слои будут землёй
public LayerMask whatIsGround;
//позиция для проверки касания земли
public Transform groundCheck;
```

И изменяем функцию проверки земли

```
private void CheckGround()
{
        isGrounded = Physics2D.OverlapPoint(groundCheck.position, whatIsGround);

    // Collider2D[] collider = Physics2D.OverlapCircleAll(transform.position, 0.3f);
    //  isGrounded = collider.Length > 1;

}
```

Изменяем переменные скрипта