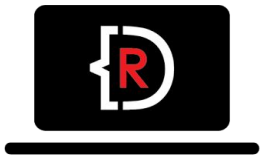
Game design

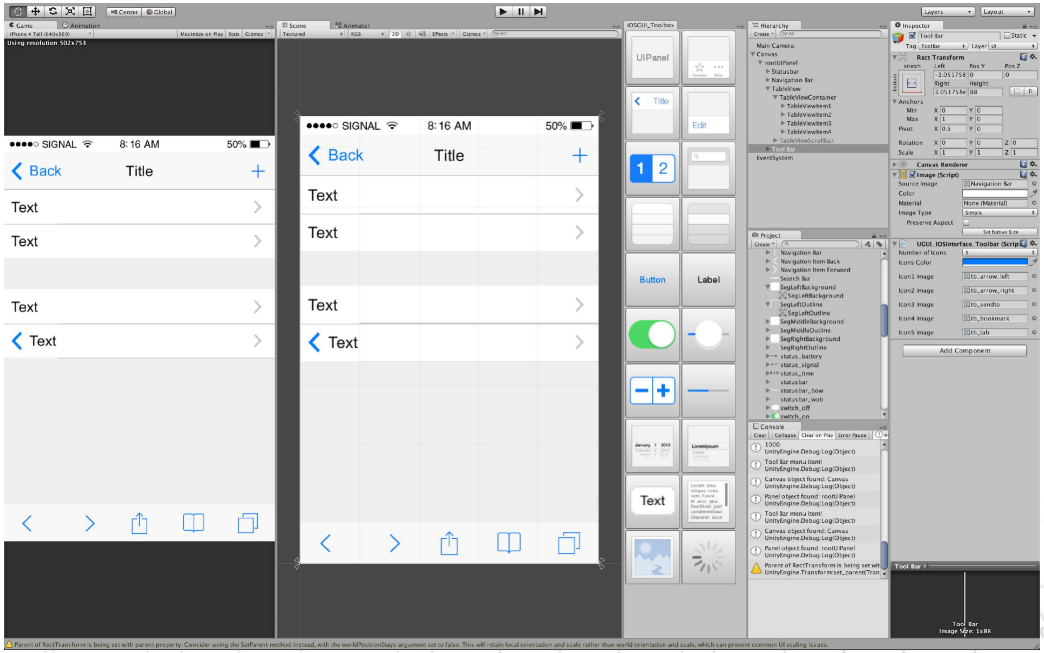


<{DeveloperR/>

Платформа Unity

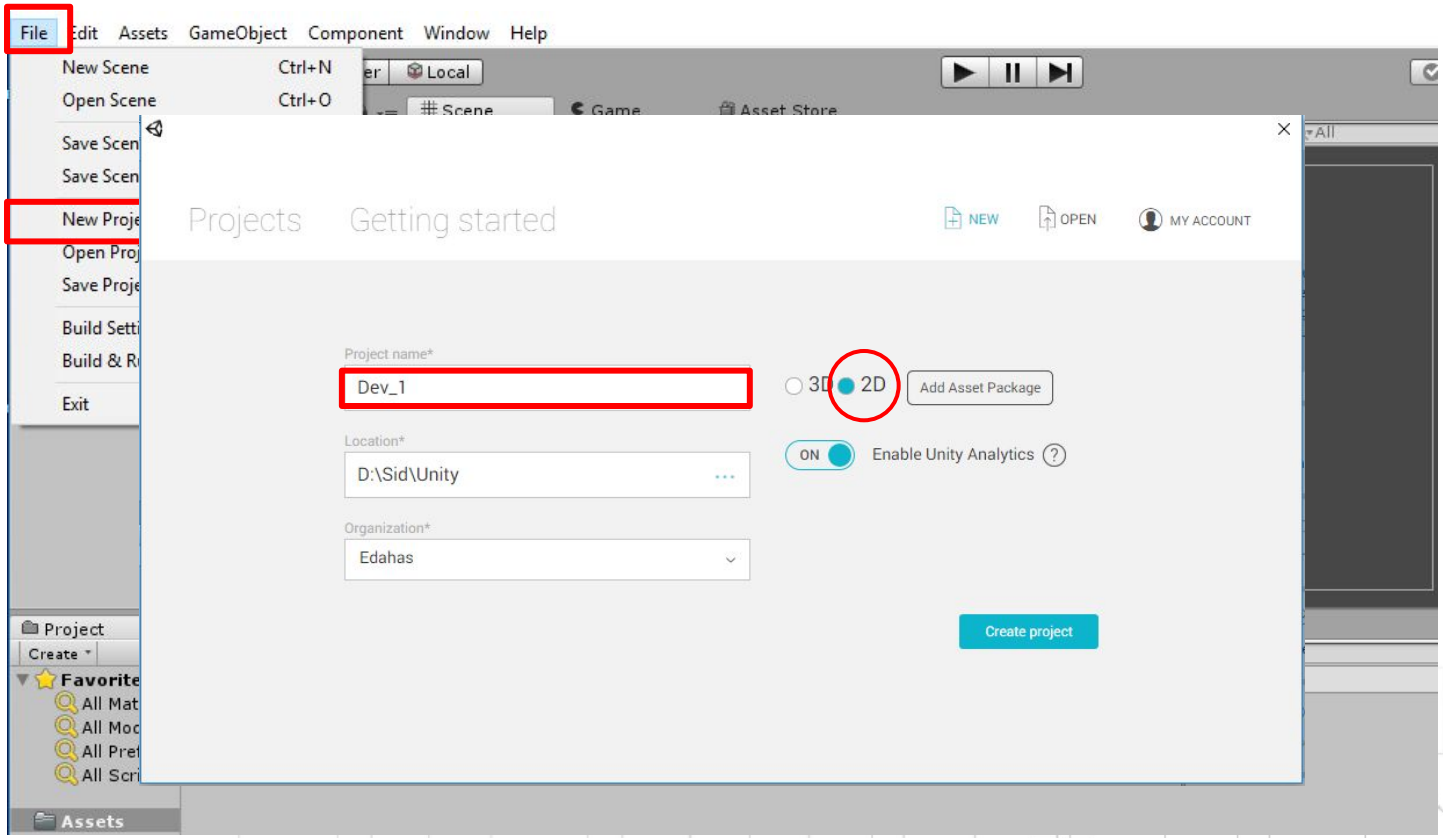
Платформа Unity

Платформа Unity, останнім часом все ширше застосовується не лише для розробки ігор, а й для створення інтерактивного інтерфейсу різних додатків.



Платформа Unity

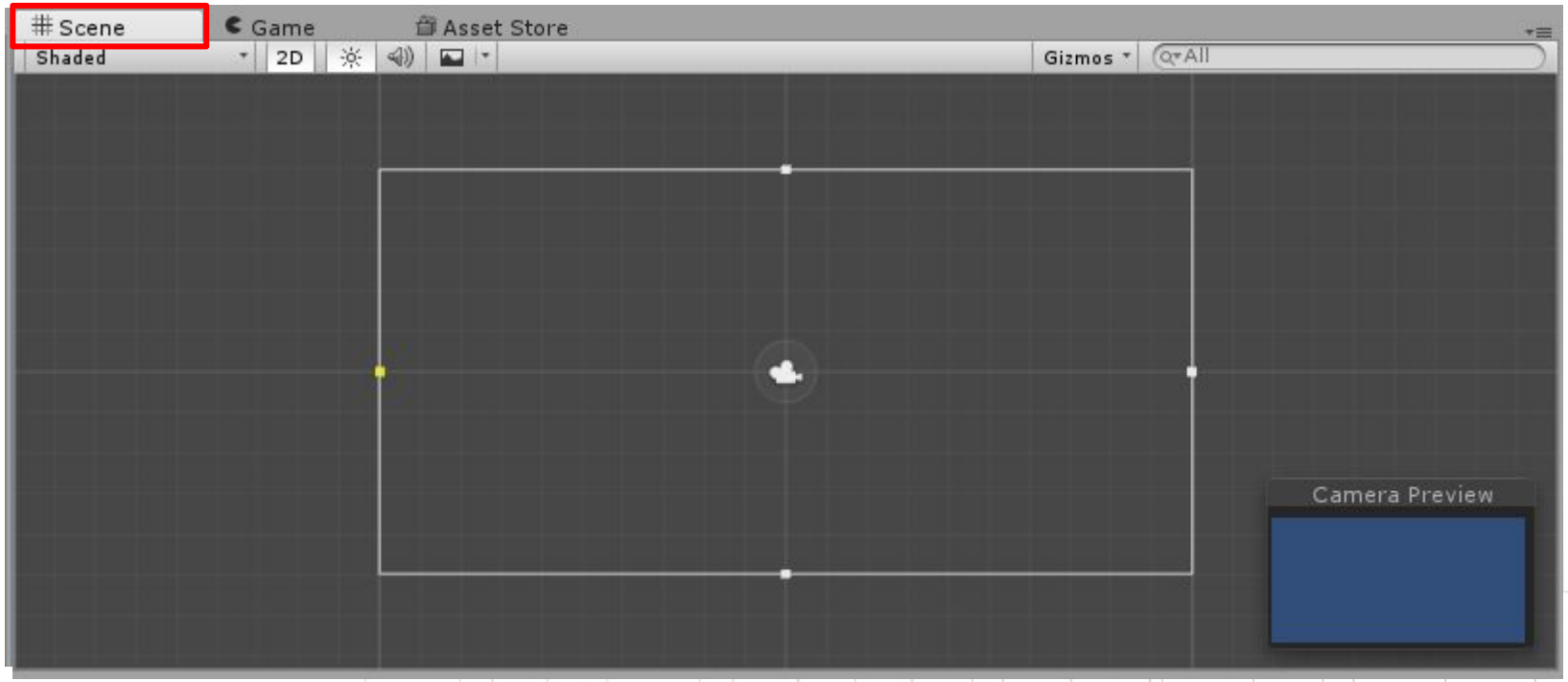
Для створення першого проекту в Unity натисніть “File”->”New Project”.



Платформа Unity

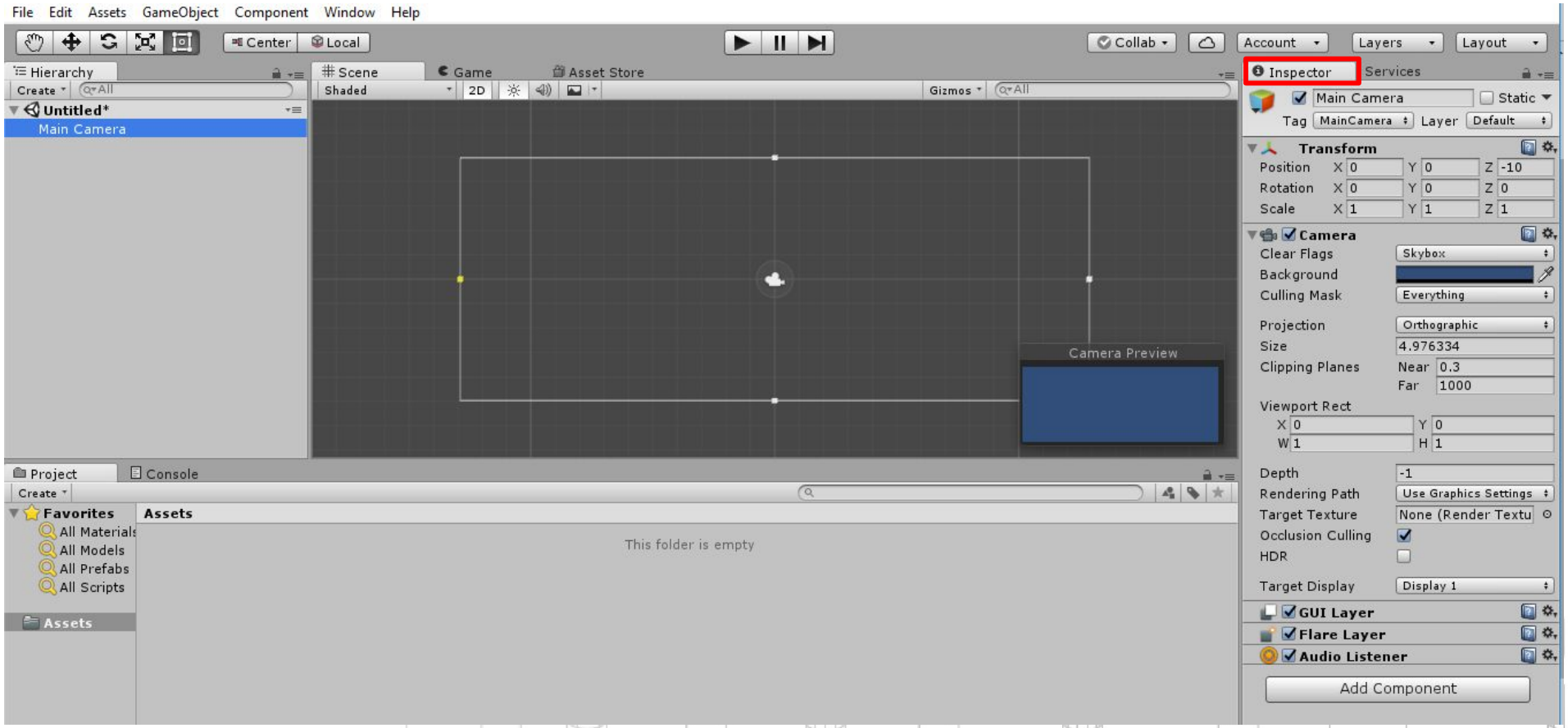
У вікні «Scene» ми будемо бачити всі ігрові об`єкти під час розробки.

У вікні «Game» ми бачимо об`єкти так, як вони відображаються в камері .



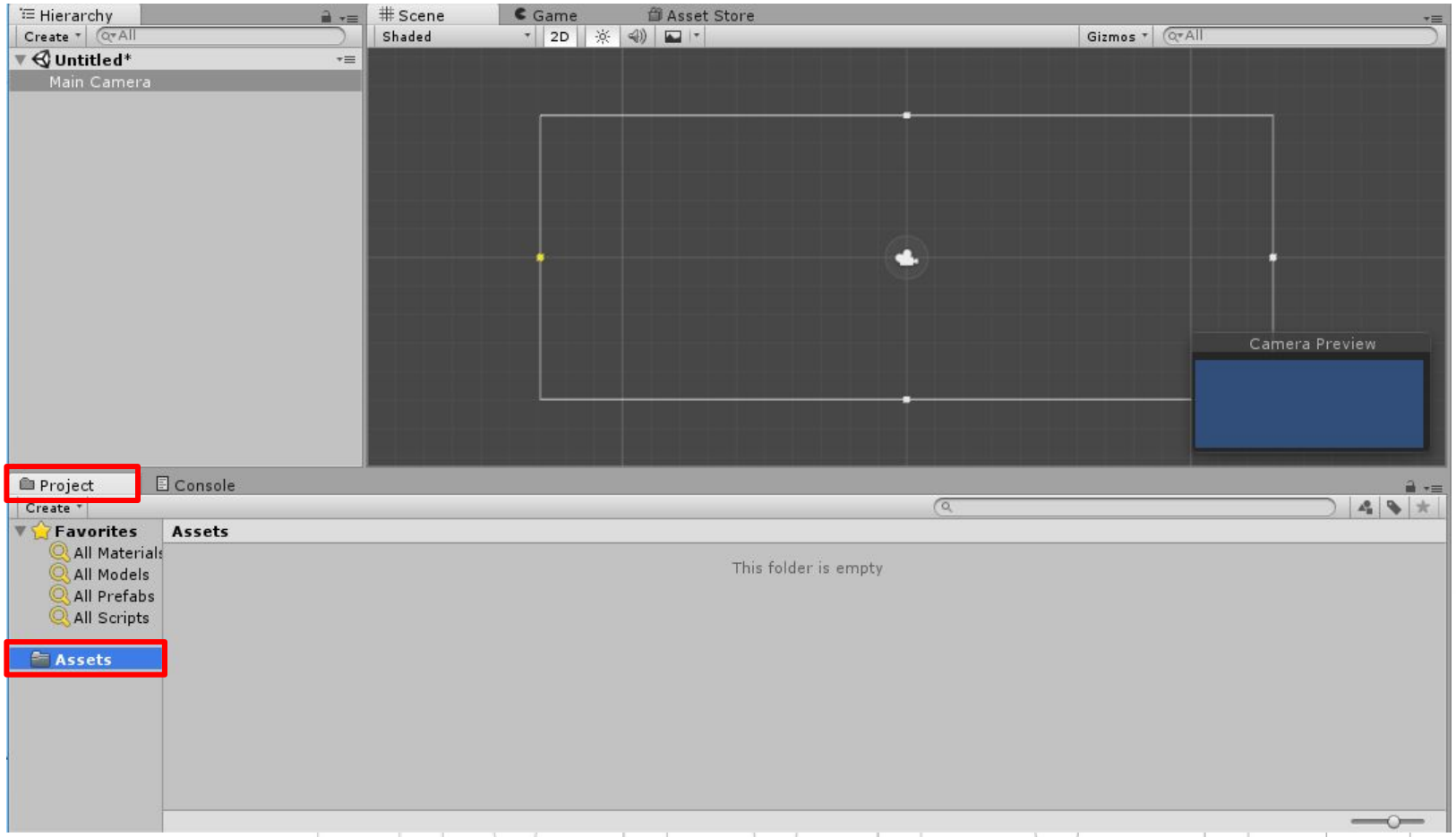
Платформа Unity

У вікні Inspector відображаються властивості всіх об`єктів.



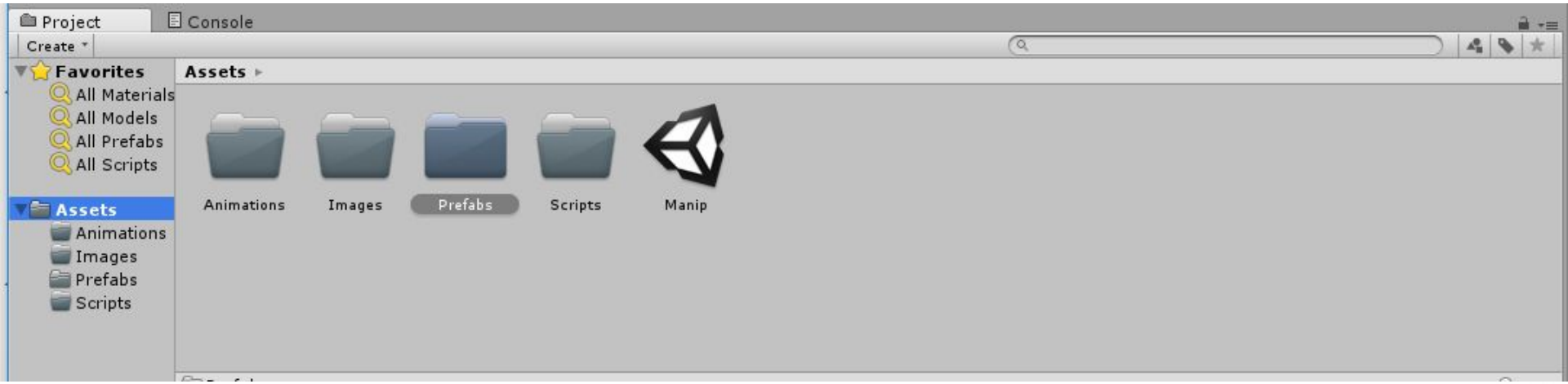
Платформа Unity

В папці Projects знаходиться папка Assets, в ній зберігаються всі файли, що ми використовуємо.



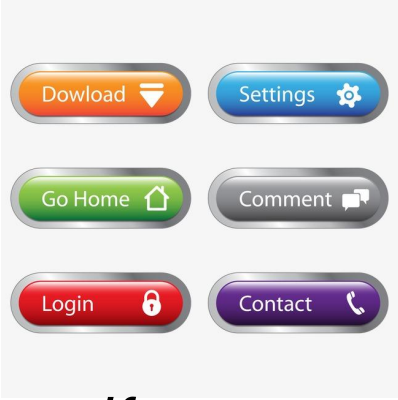
Платформа Unity

В папці Assets, для зручності створимо папки, в яких будуть зберігатись зображення, скрипти, префаби, тощо..



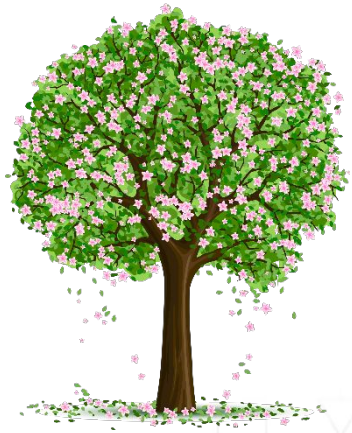
Платформа Unity

Об`єктами в Unity може бути що завгодно:



Кнопка

Будівля



Дерево

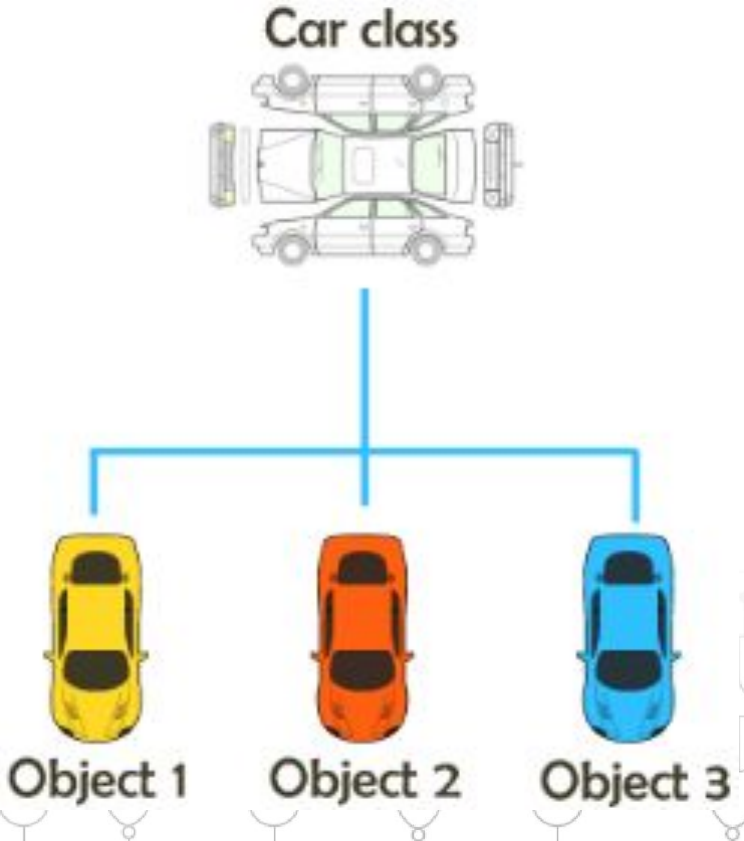
Людина

Платформа Unity

3 кити ООП

Платформа Unity

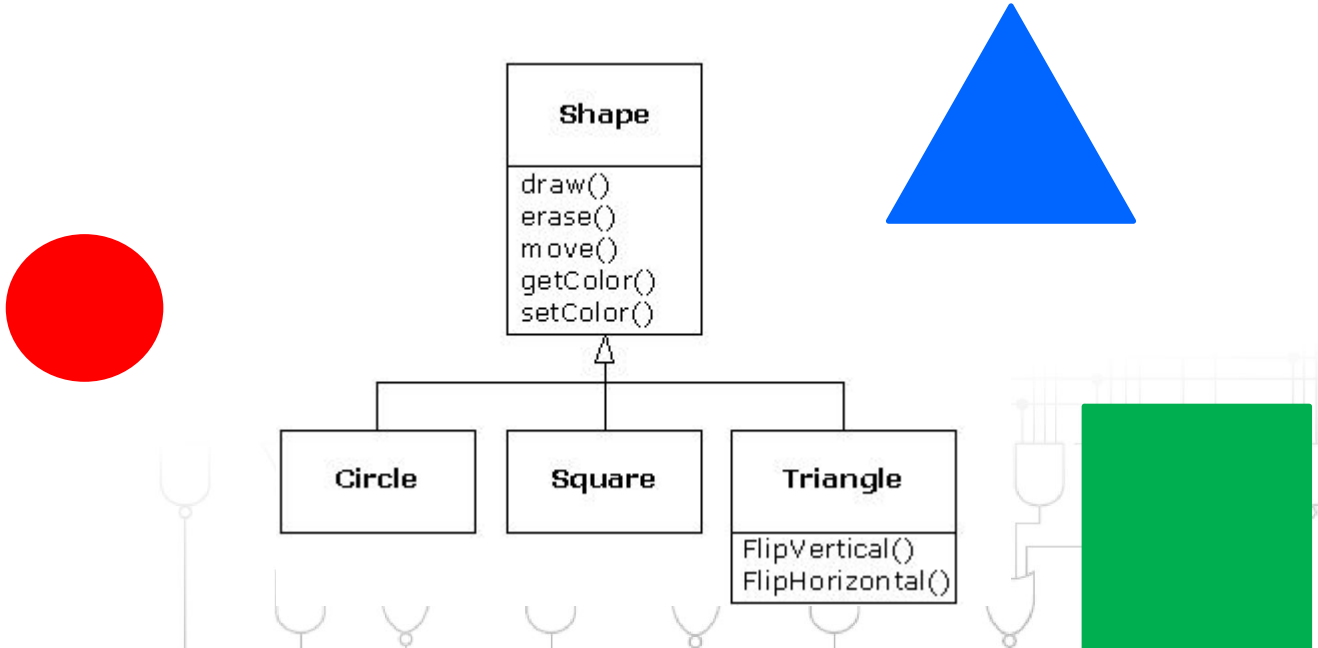
Класи - *це, по суті, шаблони, за якими можна створювати об'єкти. Кожен об'єкт містить дані і методи, котрі маніпулюють цими даними.*



Платформа Unity

Поліморфізм

Поліморфізм *– властивість, яка дозволяє одне і те саме ім'я використовувати для вирішення декількох технічно різних задач, тобто основною метою поліморфізму є використання одного імені для задання загальних класу дій.*



Платформа Unity

Наслідування

Наслідування *– процес, завдяки якому один об'єкт може придбати властивості іншого, тобто наслідувати властивість іншого обєкту і додавати риси характерні тільки для нього самого.*

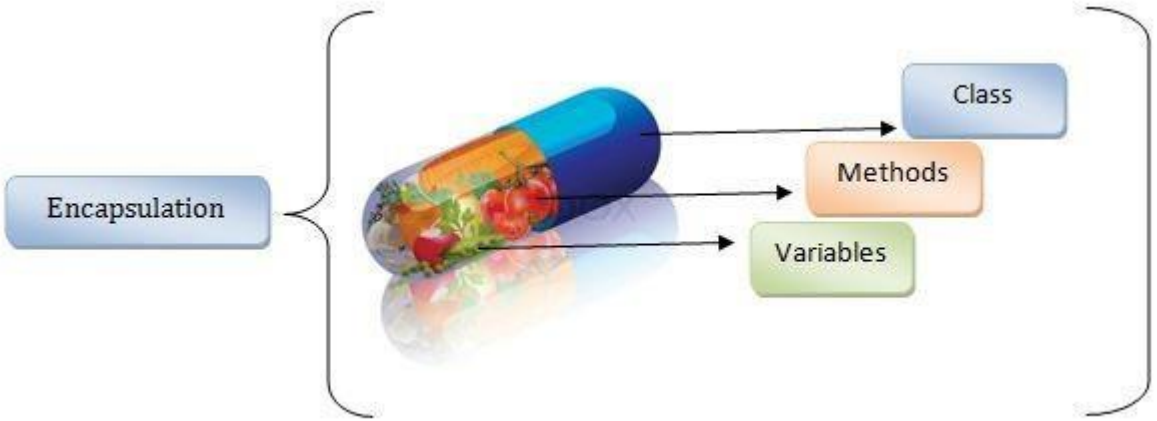
Virtual class Enemy

Платформа Unity

Інкапсуляція

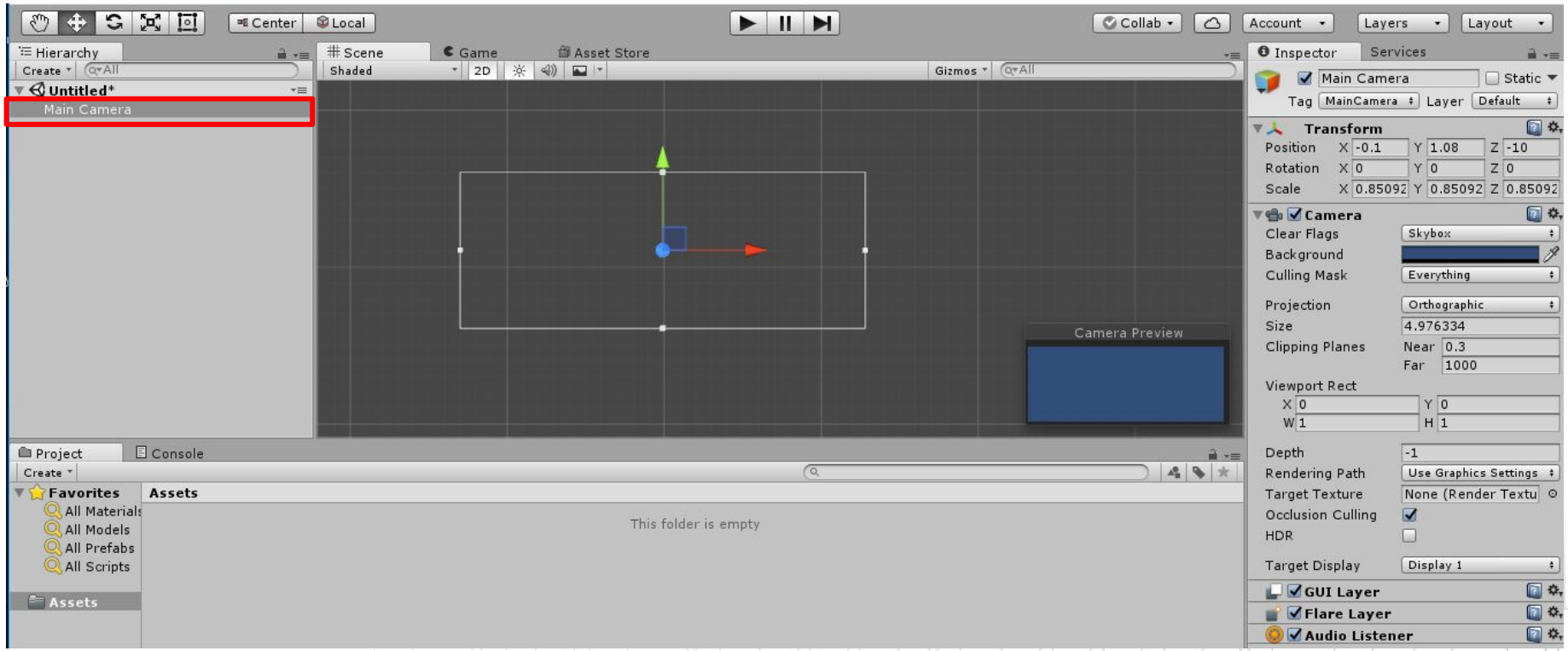
Це фундаментальна об'єктно-орієнтована концепція, що дозволяє «упаковувати» і «приховування» дані. Переваги:

- *Контроль доступу;*
- *Контроль цілісності / валідності даних;*
- *Можливість зміни реалізації.*



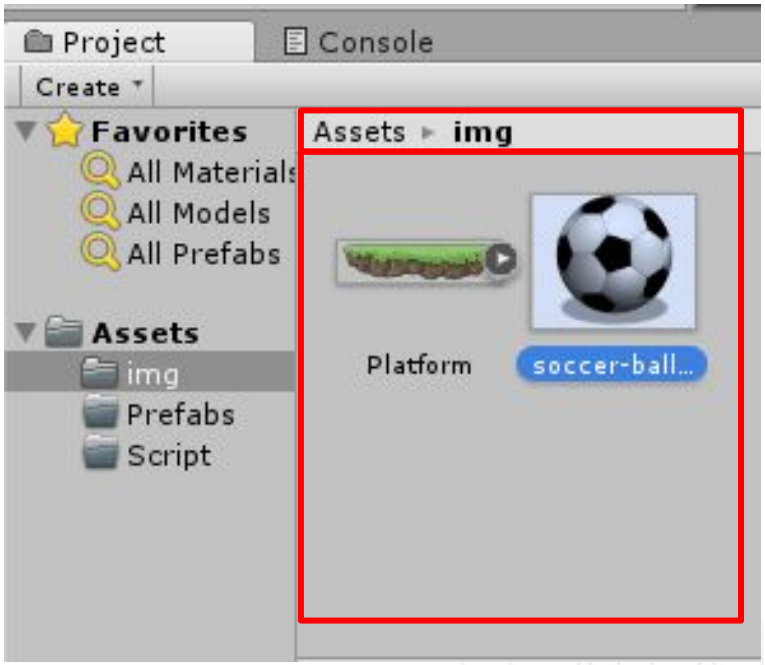
Платформа Unity

*Об`єкт “**Main camera**” – це головна камера, через яку ми будемо бачити, все, що відбувається у грі.*



Платформа Unity

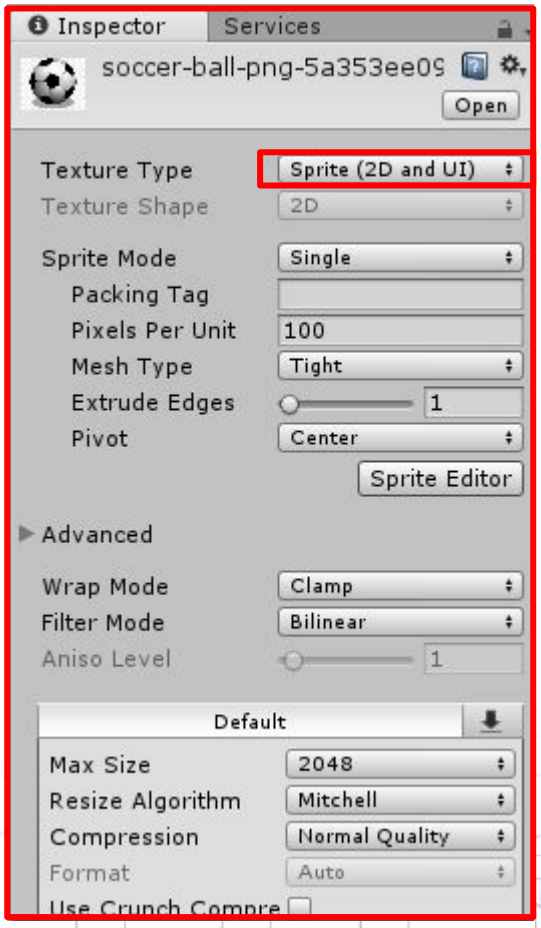
Завантажимо до папки Images, зображення, які ми використовуватимемо в якості наших об'єктів. Для цього перетягніть зображення у відповідну папку в Assets:



Перетягувати зображення слід у вказану область

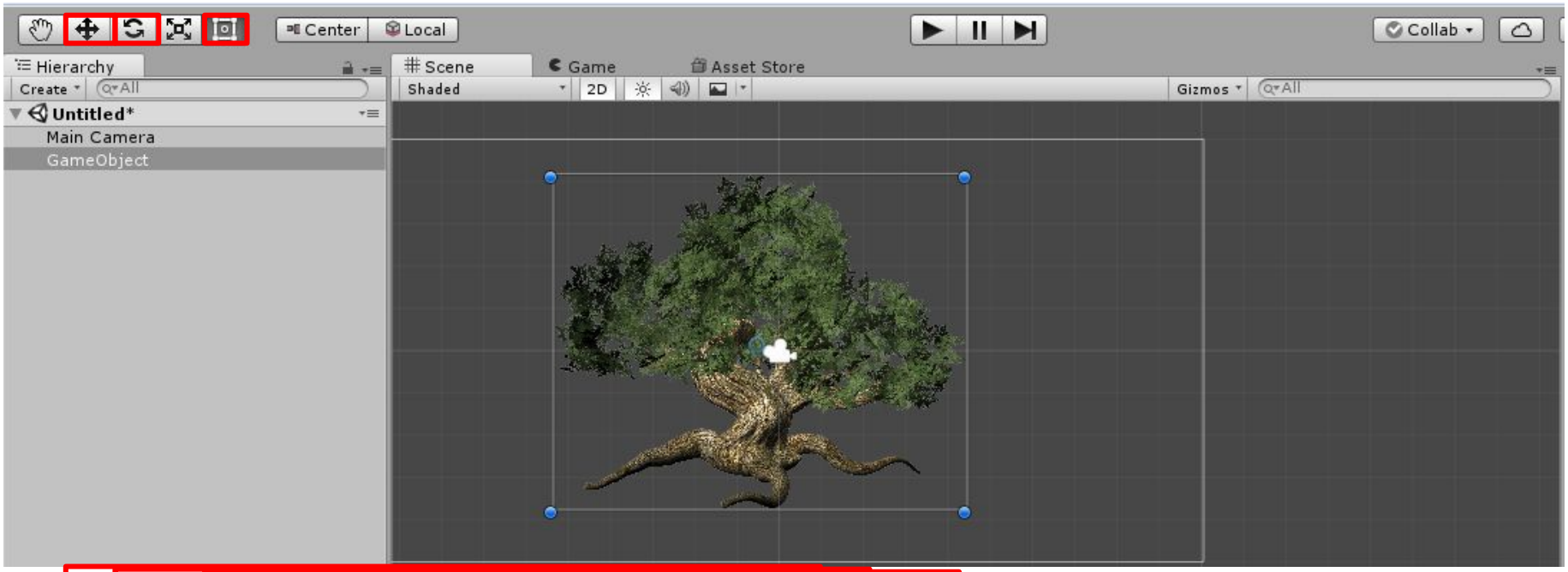
Платформа Unity

В закладці Inspector, переконайтесь, що тип текстур Sprite (2D and UI):



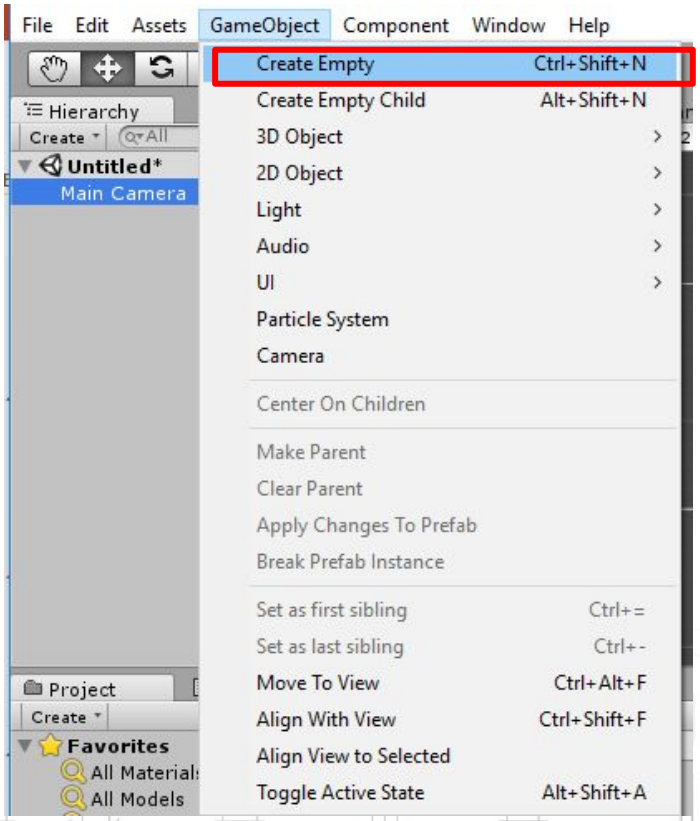
Платформа Unity

Налаштування для роботи із об`єктом:



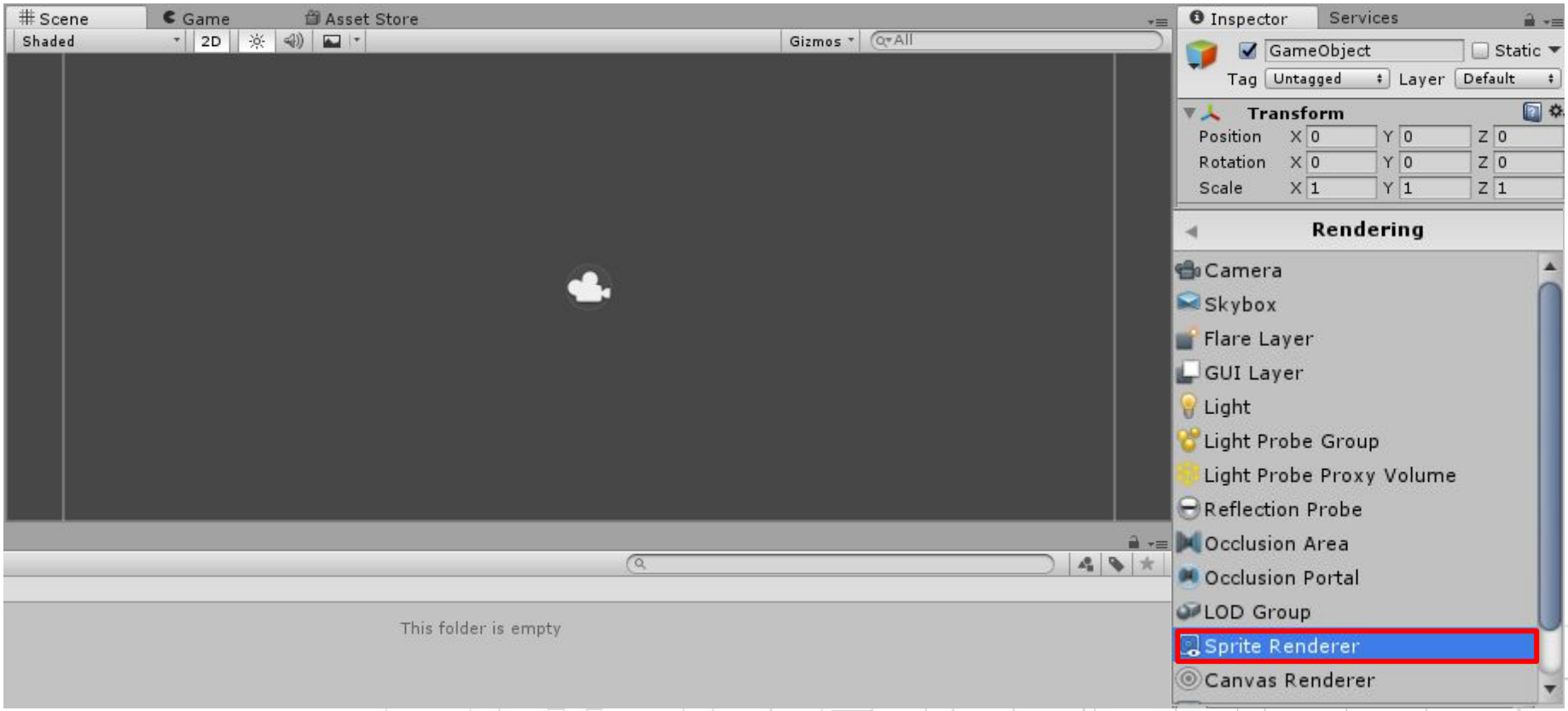
Платформа Unity

Щоб створити новий об`єкт натисніть “Game object” -> “Create Empty”.



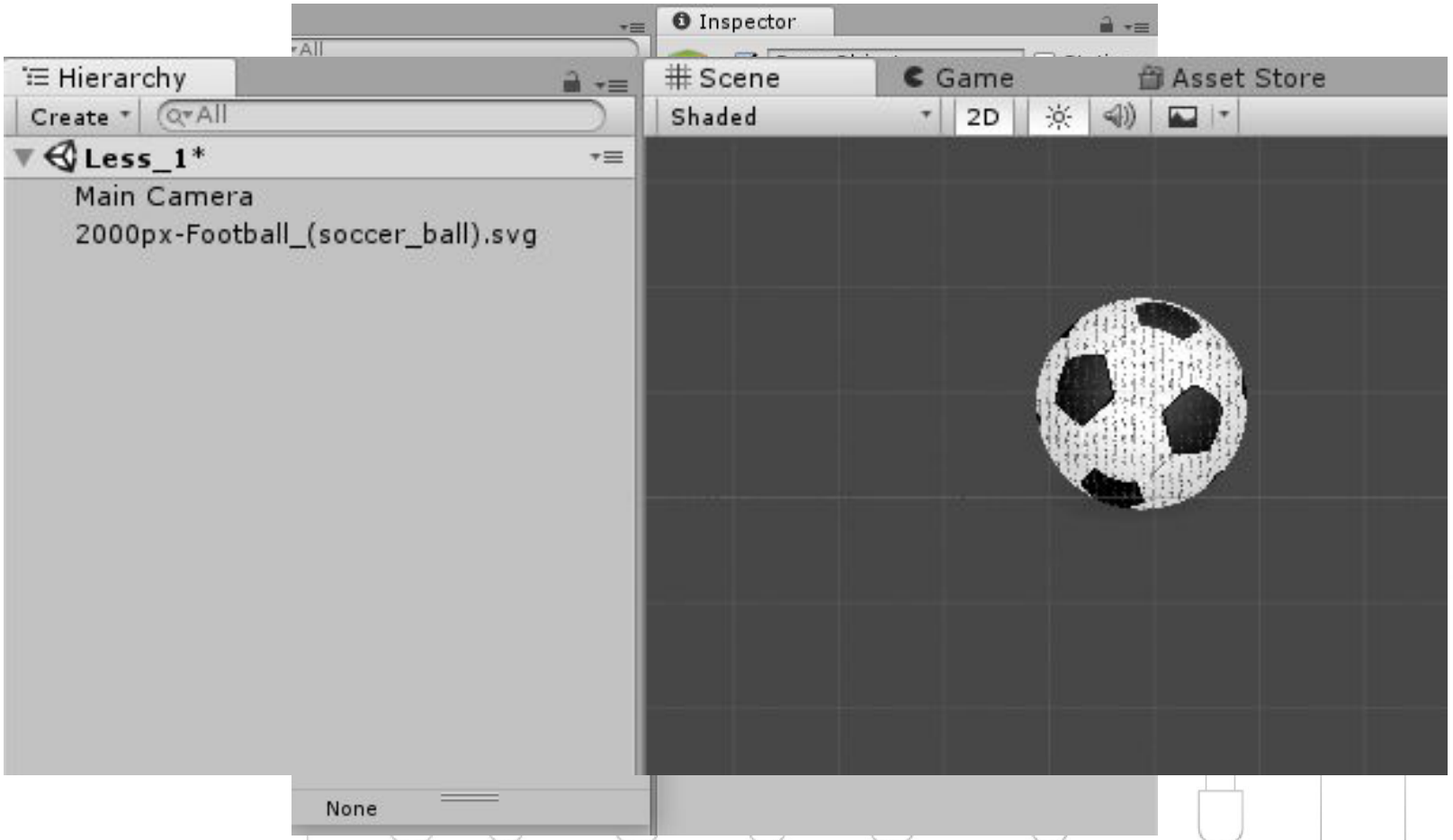
Платформа Unity

Прив`яжемо до об`єкта картинку. Для цього в Add Component оберемо пункт Rendering:



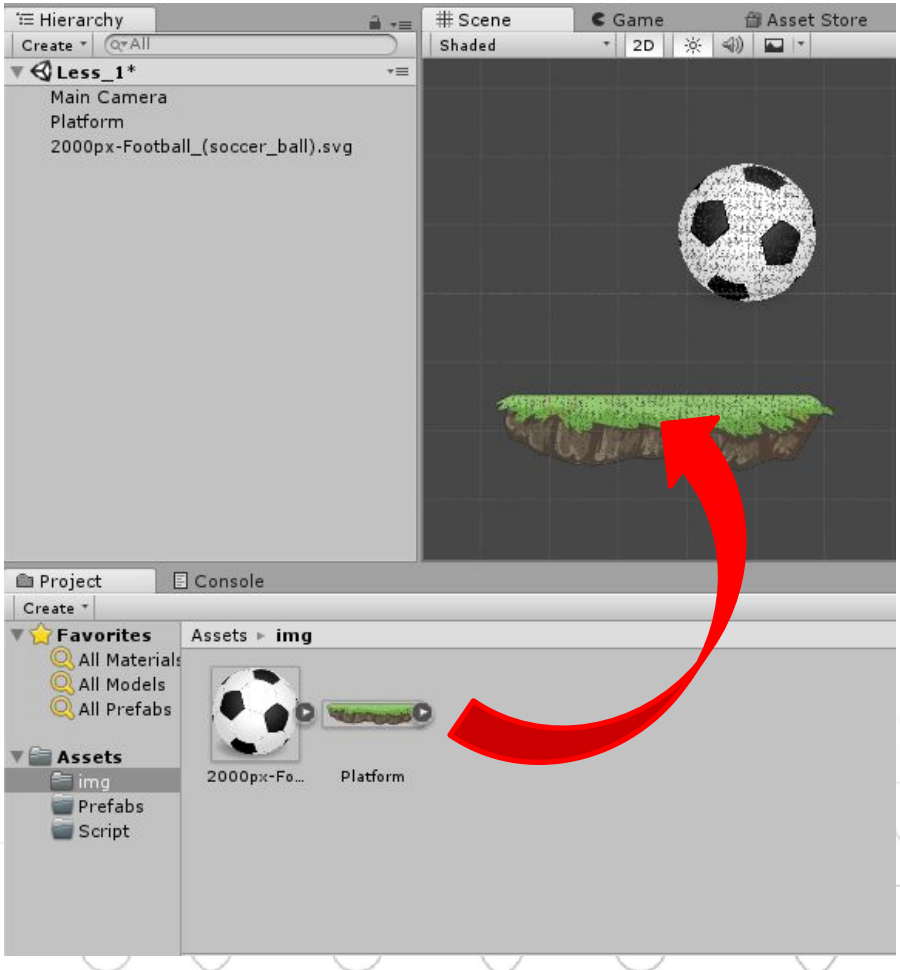
Платформа Unity

Натисніть на позначку біля Sprite, у вікні, що з'явиться оберіть ім'я вашого .png (.jpg або ж .svg) файлу:



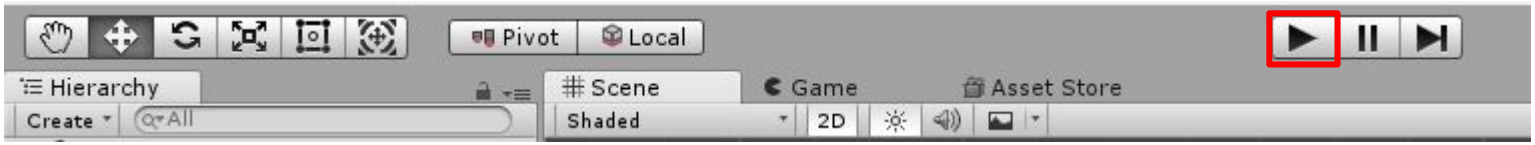
Платформа Unity

Простіший спосіб - перетягніть картинку з папки в робочу зону, автоматично буде створено об'єкт із заданим рендером:

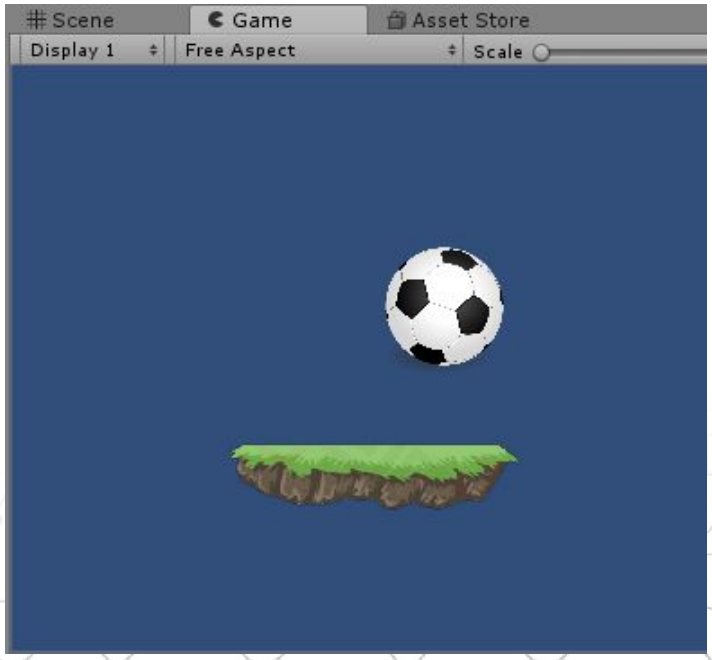


Платформа Unity

Запустіть гру:



Ви помітили, що об’єкти просто залишаються на своїх позиціях:

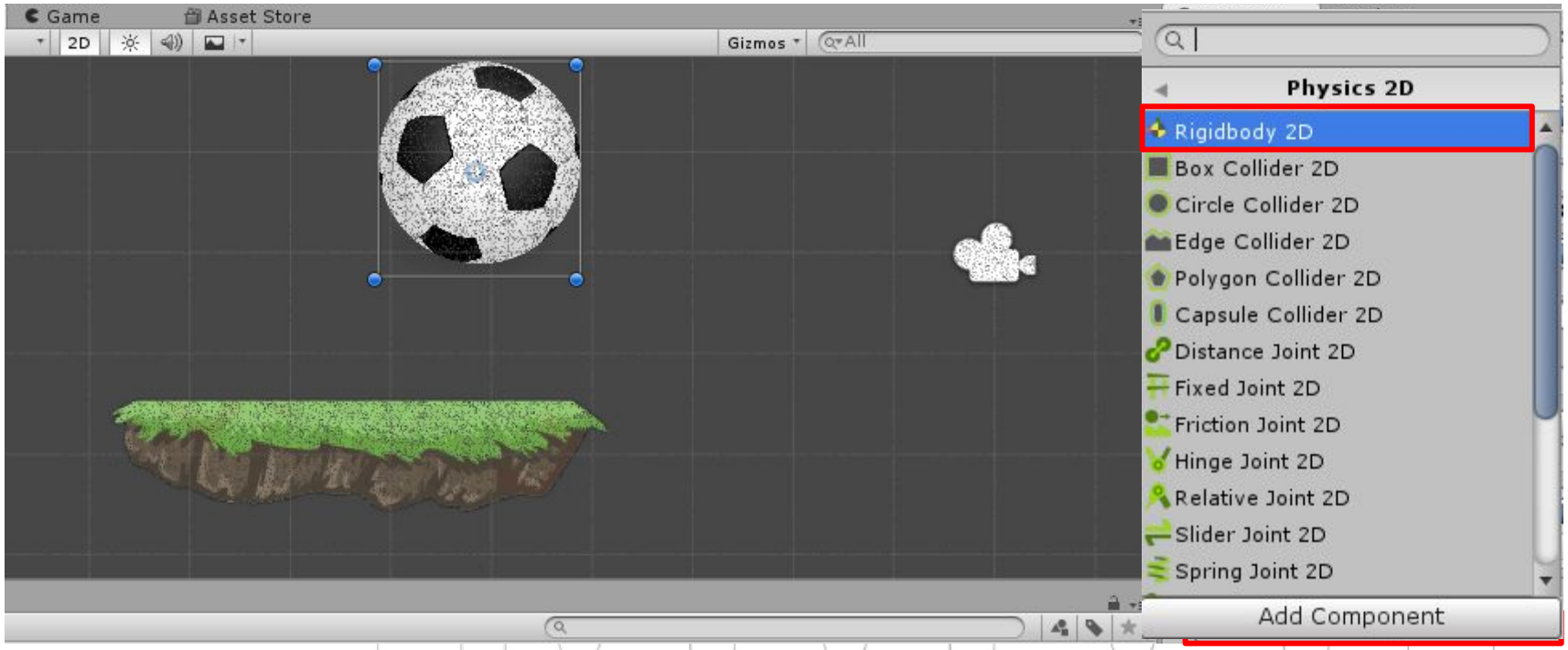


Платформа Unity

На даному етапі об’єкти – це тільки картинки. Будь-який фізичний об’єкт має масу, на нього діє сила тяжіння, тертя, він володіє коефіцієнтом пружності, тощо.

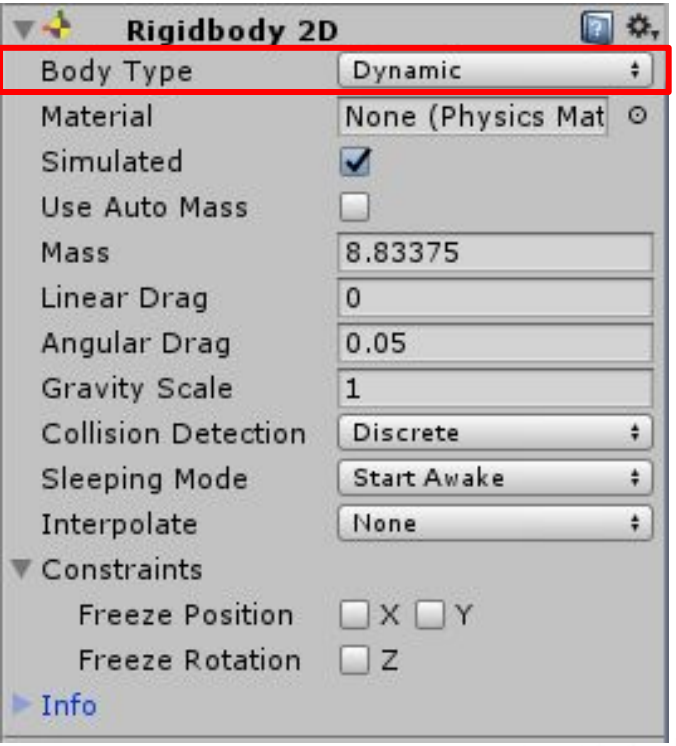
Платформа Unity

Додамо нашим об'єктам фізичних властивостей. Задамо об'єкту масу і силу тяжіння, що на нього діятиме:



Платформа Unity

Налаштування Rigidbody:

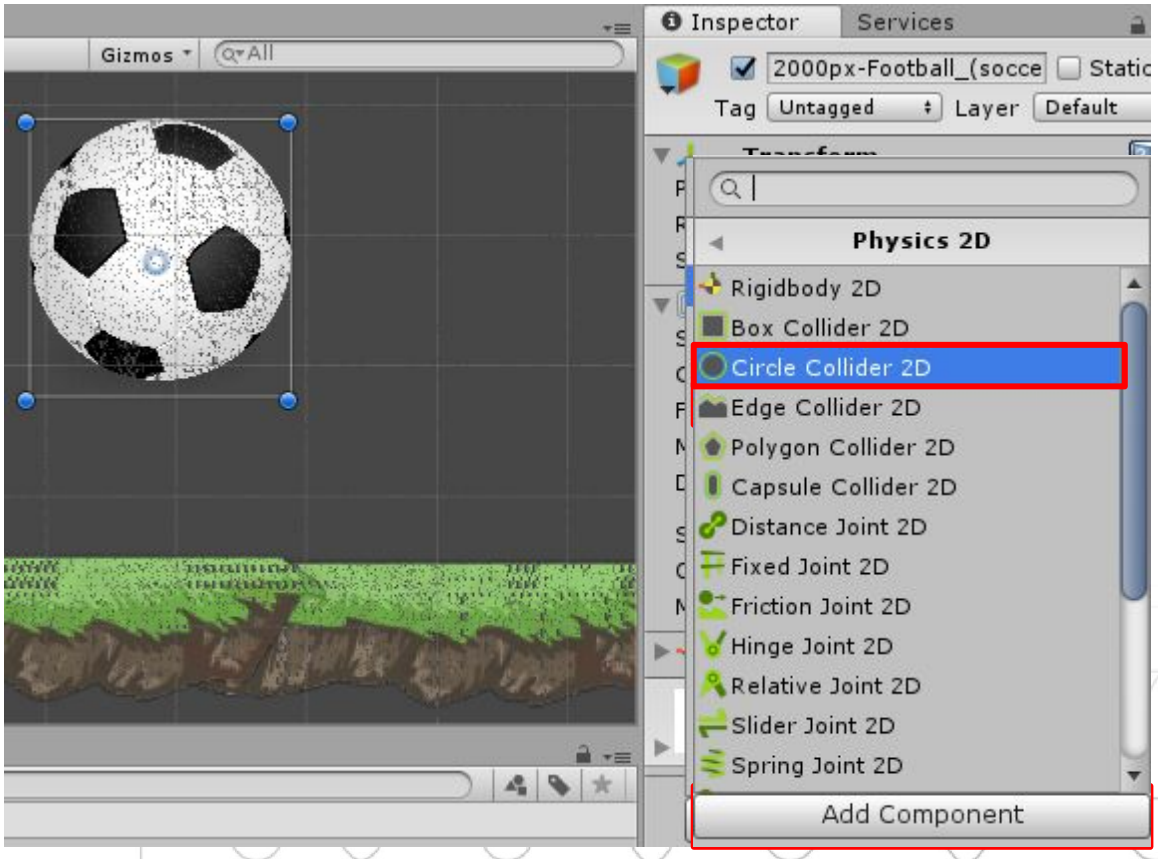


Тип Rigidbody
Матеріал

Маса об'єкта
Лінійне сповільнення
Кутове сповільнення
Гравітаційний масштаб

Платформа Unity

Поки що задані властивості пов'язані з деякою уявною точкою (центру мас об'єкта), слід зв'язати їх із самим об'єктом за допомогою Collider:



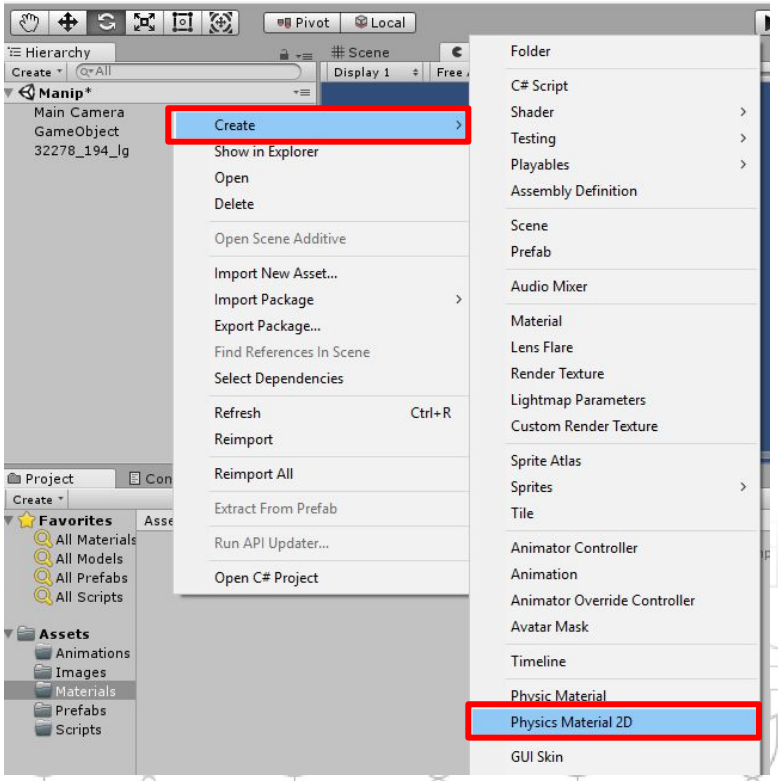
Платформа Unity

Додайте Poligon Collider 2D вашій основі і запустіть :

Ви помітили, що м’яч упав і зупинився. В реальному світі це не можливо, адже навіть металева кулька зробить незначний відскок.

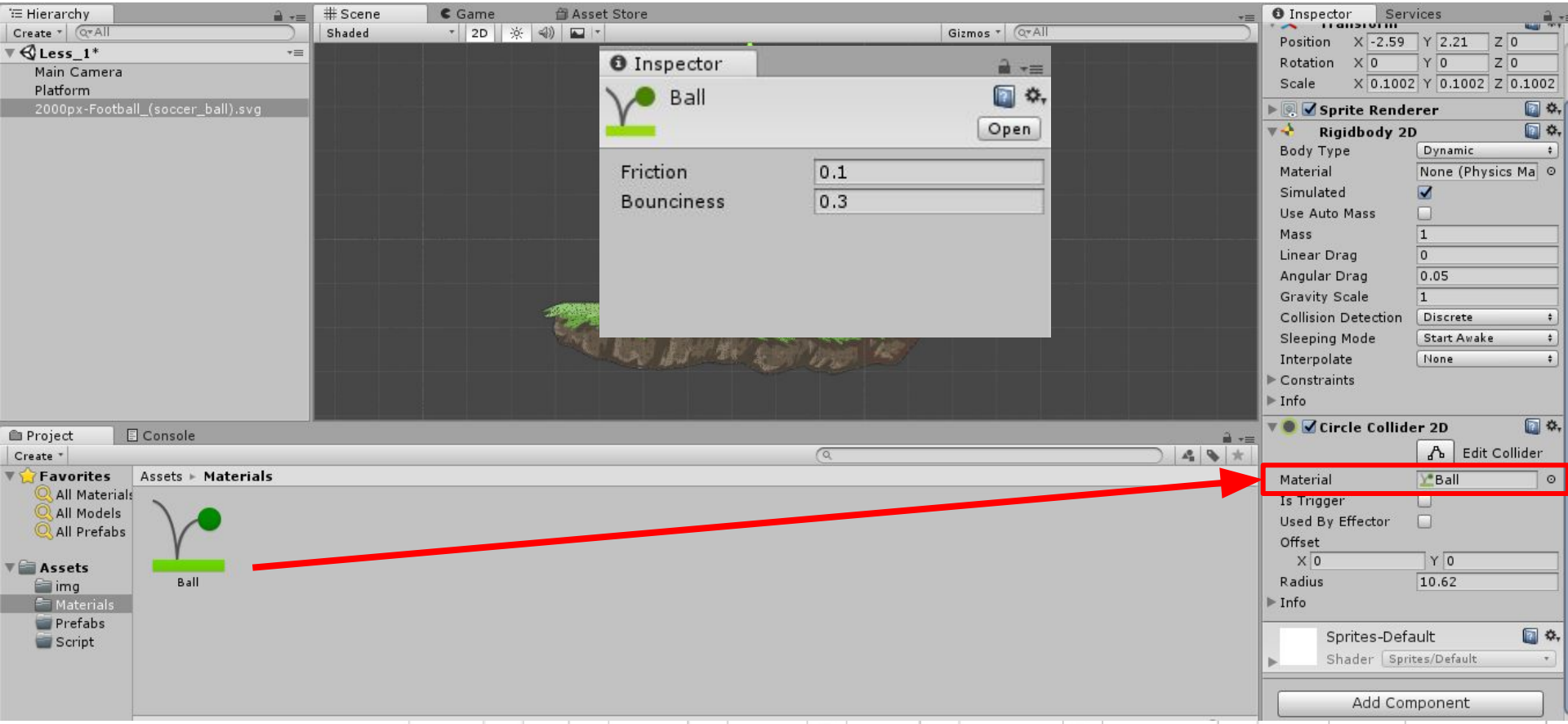
Платформа Unity

Нам слід додати ще 2 фізичні параметри: пружність та силу тертя. Для цього створимо в папці Assets папку Materials в ній пкм ->Create->Physical Material 2D:



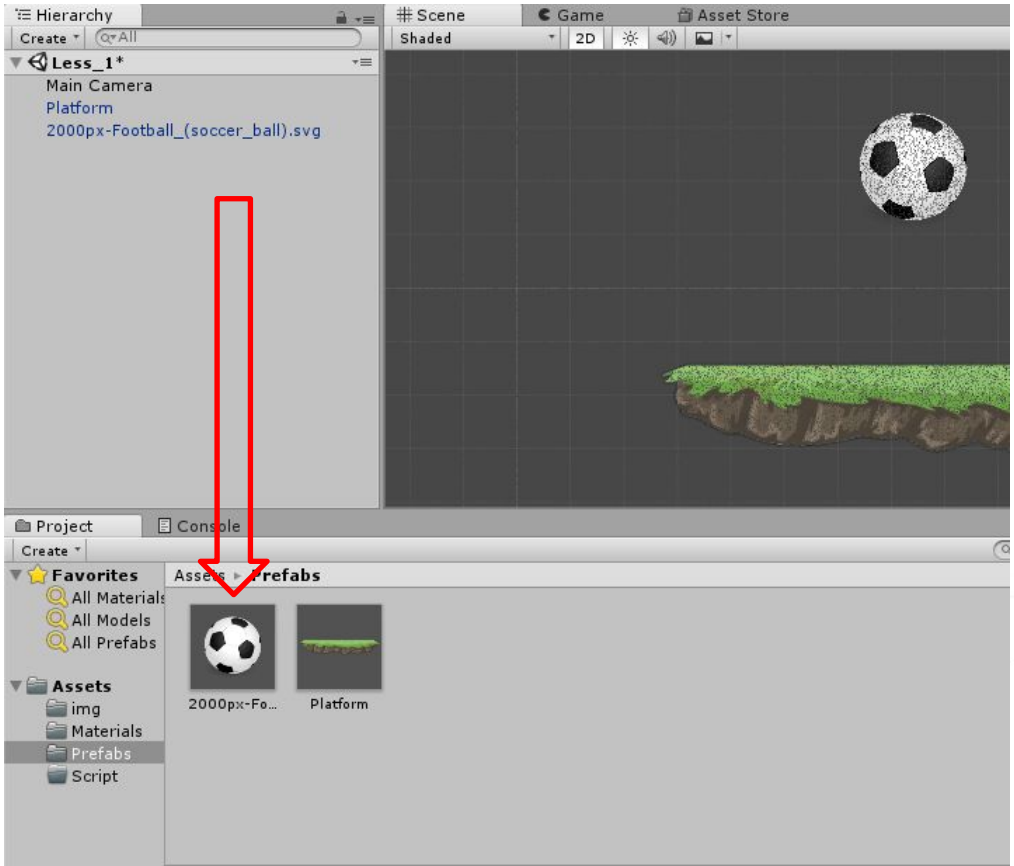
Платформа Unity

Підберіть значення пружності та сили тертя і приєднайте створений матеріал до колаедра м'яча:



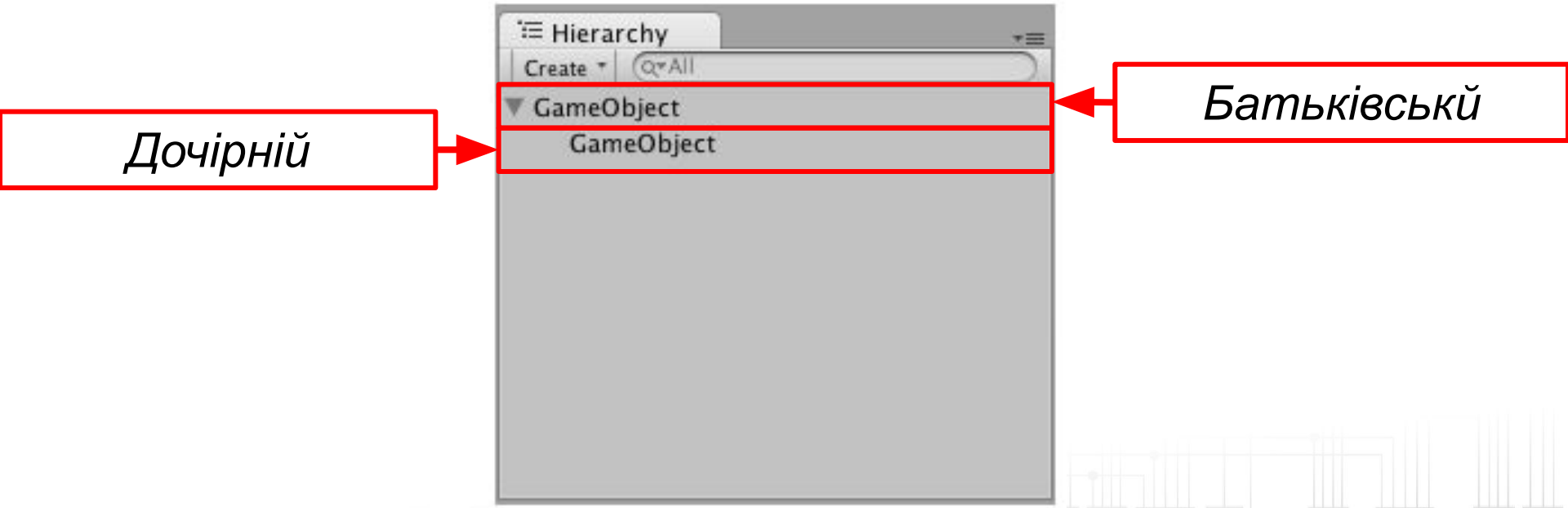
Платформа Unity

Якщо ви плануєте використовувати декілька однакових об'єктів, то щоб кожного разу не налаштовувати всі параметри, можна з існуючого об'єкта створити заготовку Prefab:



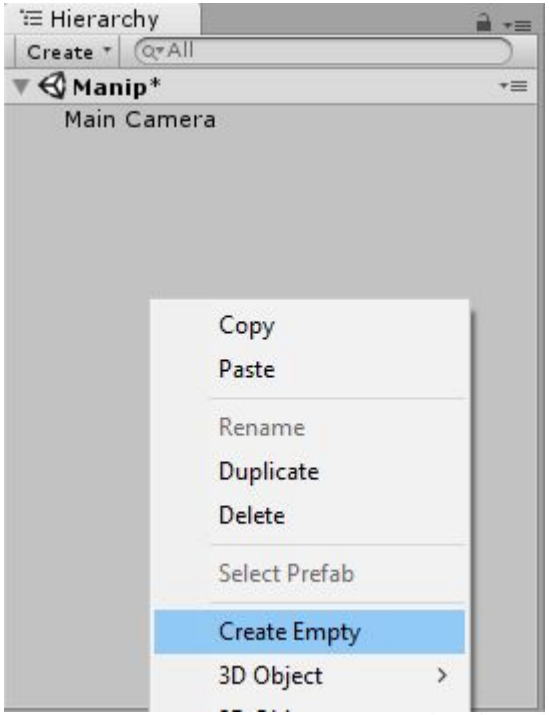
Дочірні об’єкти в Unity

Unity використовує концепцію під назвою Parenting. Для того, щоб зробити будь-який GameObject дочірнім для іншого, перемістіть бажаний дочірній об'єкт на відповідний батьківський в hierarchy.

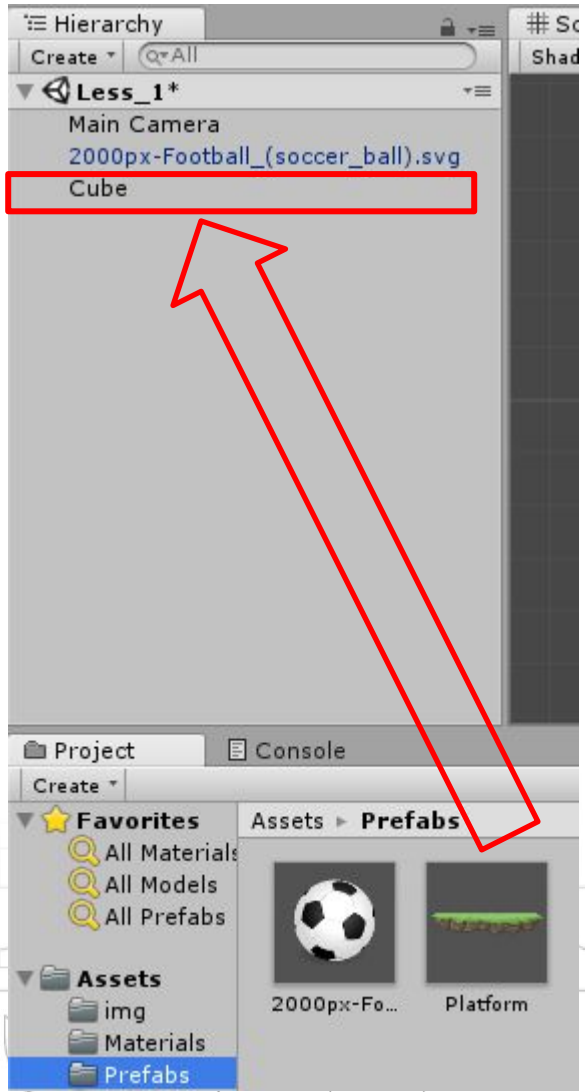


Взаємний рух об’єктів в Unity

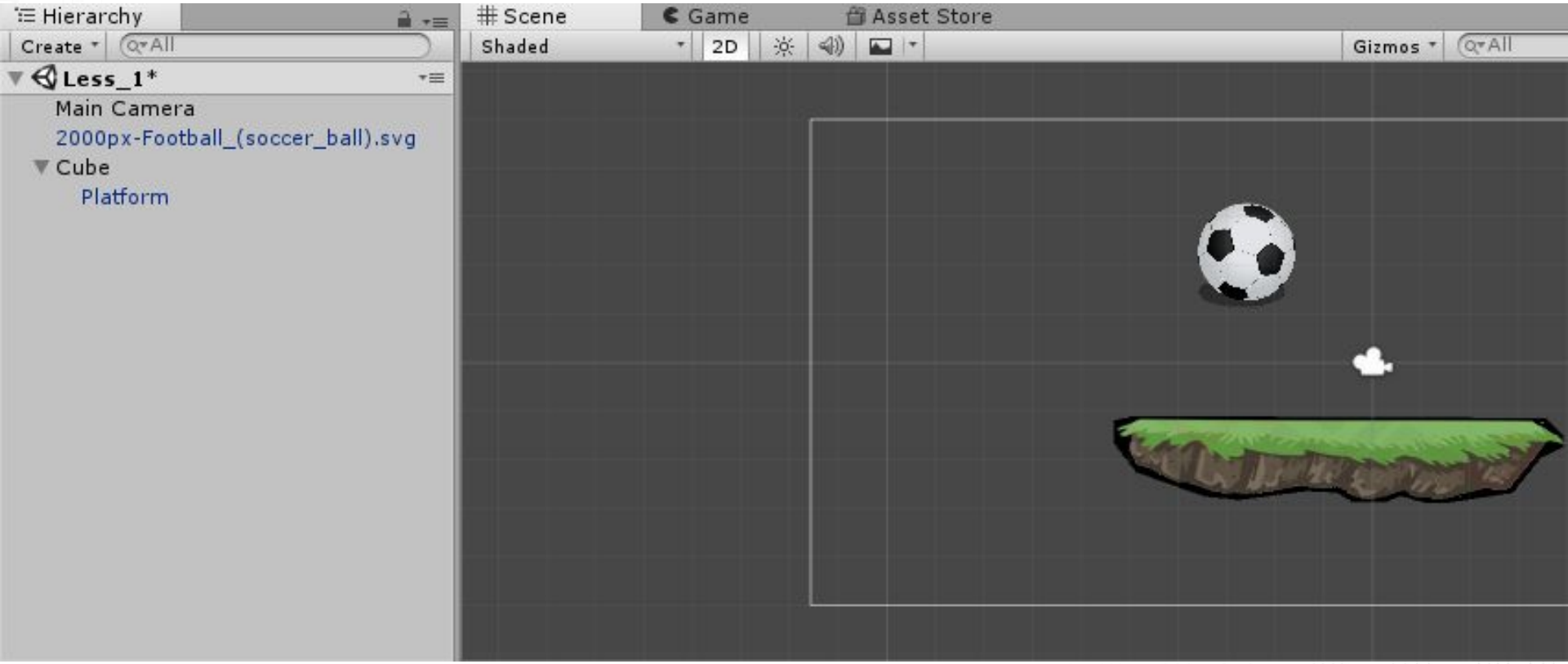
Створимо порожній об’єкт, відносно якого буде виконуватись рух, назвемо його Cube:



Перетягніть заготовку на створений порожній об’єкт



Взаємний рух об’єктів в Unity

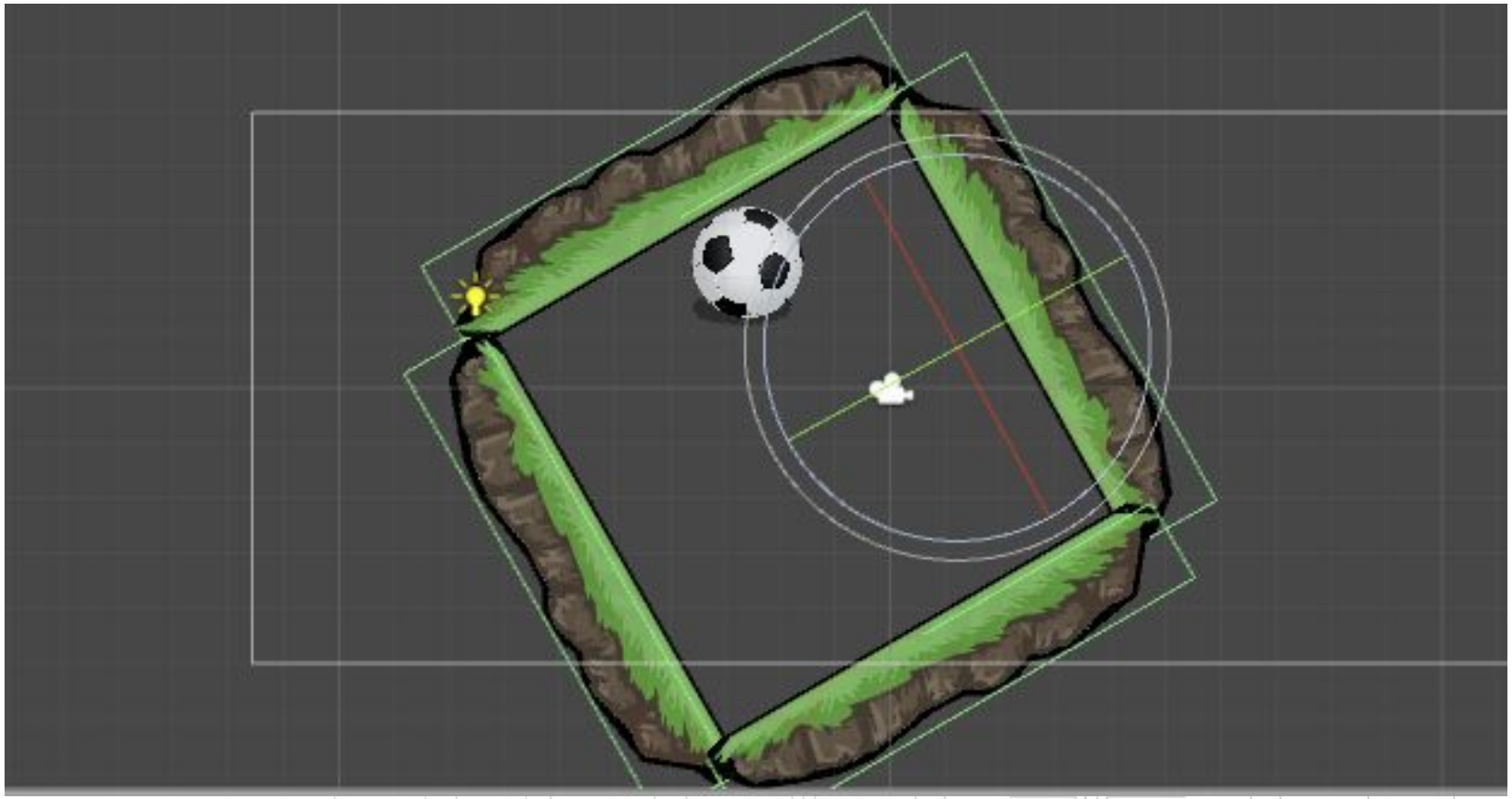


Взаємний рух об’єктів в Unity

Додайте ще три платформи, щоб вийшов квадрат:

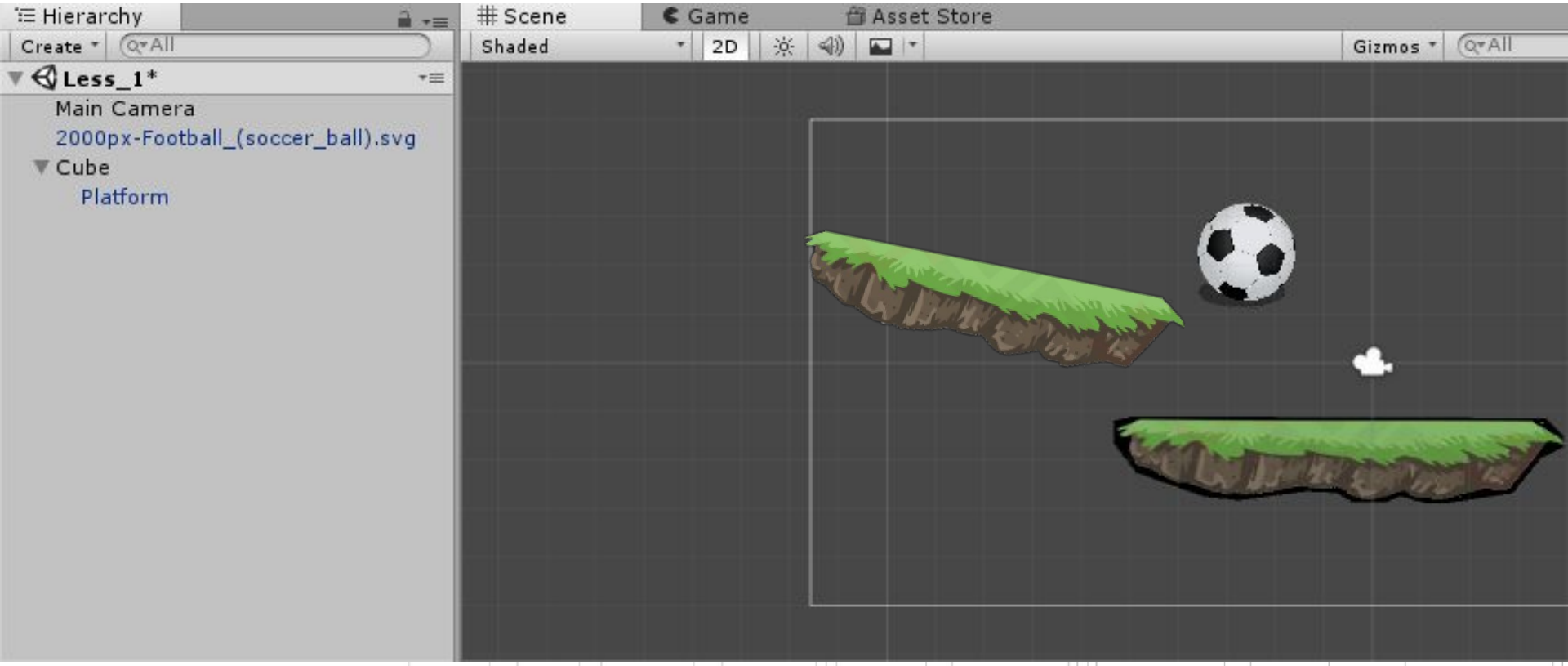
Взаємний рух об’єктів в Unity

Виділіть батьківський об’єкт (Cube) та поверніть його на певний кут



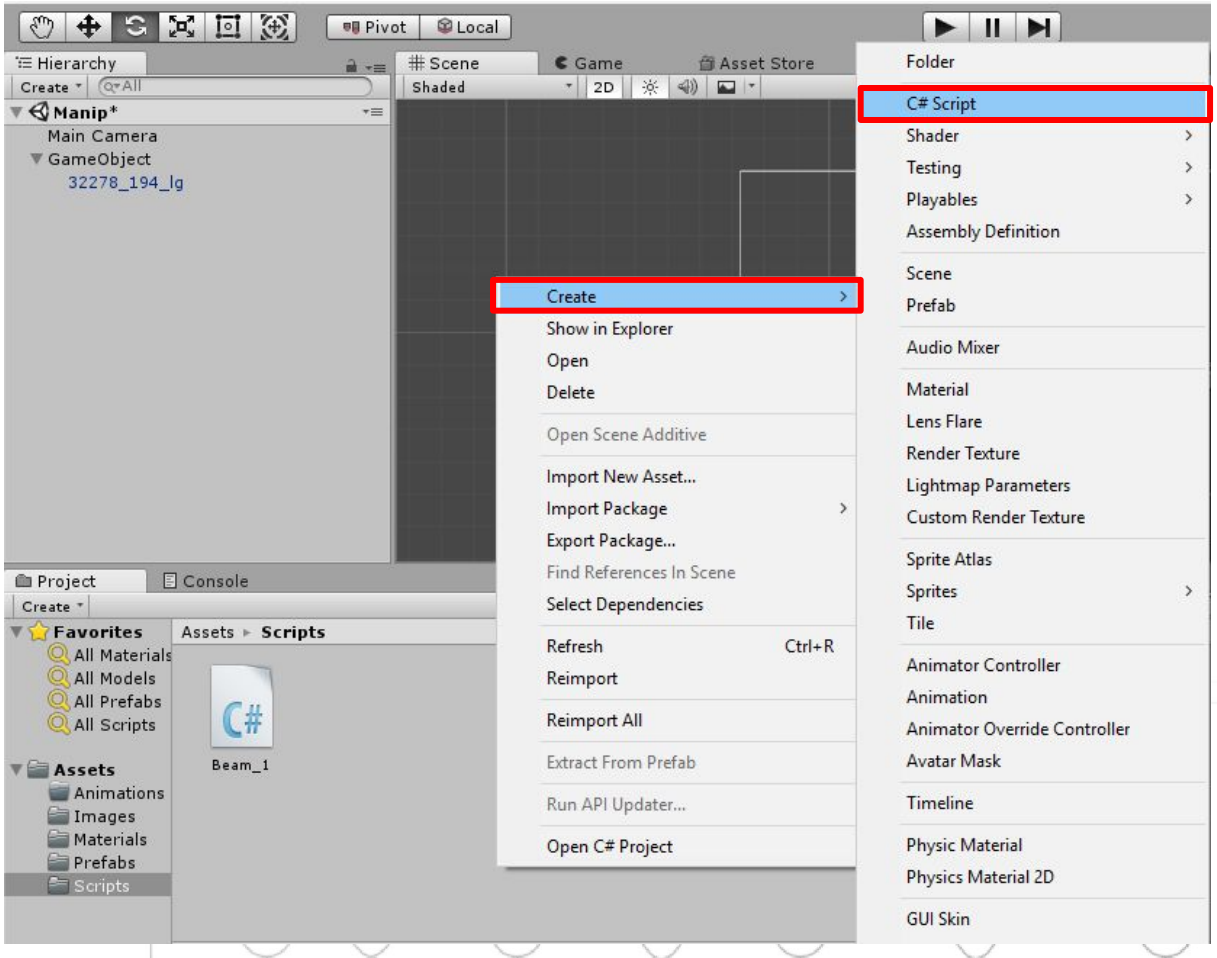
Взаємний рух об’єктів в Unity

Створіть систему похилих площин, щоб змоделювати процес ступінчастого падіння



Скрипти C#

Створимо скрипт. Для цього перейдемо в папку Scripts->пкм-> Create->C# Script, задайте назву Скрипта:



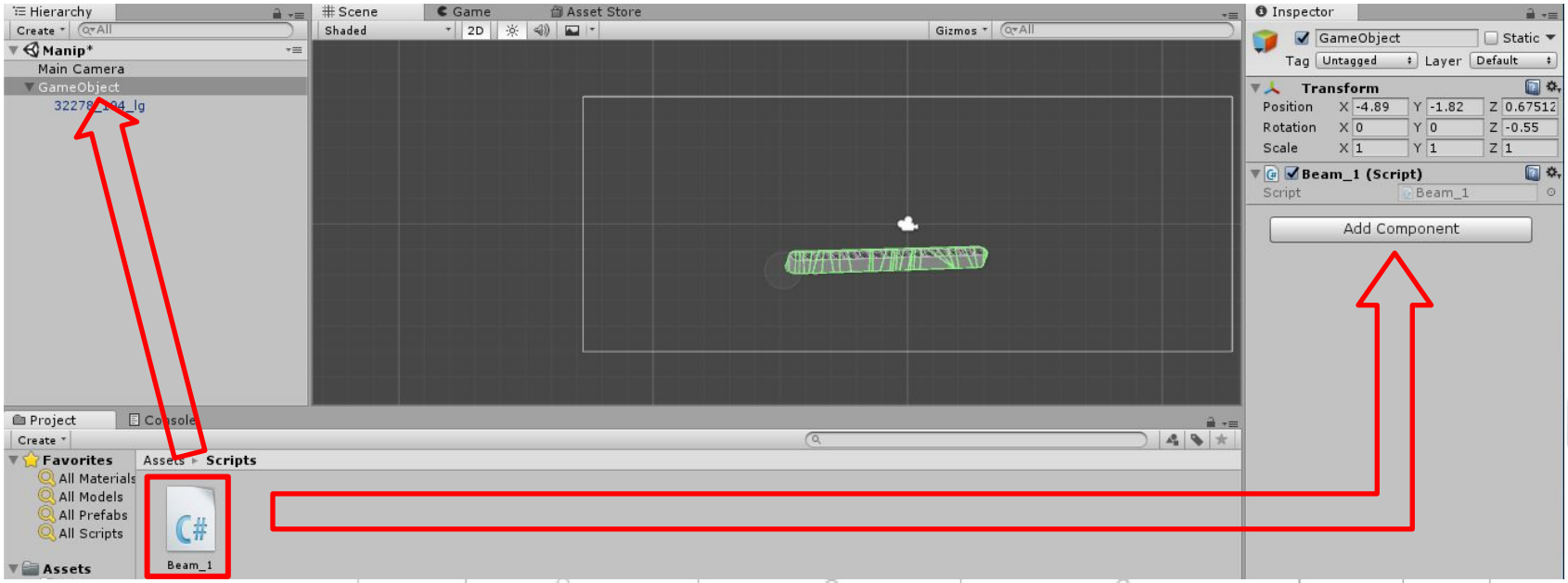
Скрипти C#

Відкрийте скрипт (подвійний клік на файлі):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Beam_1 : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

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

    }
}
```

Скрипти C#

Скрипт визначає тільки план компонента, а отже, його код не буде активований до тих пір, доки екземпляр скрипта не буде приєднаний до ігрового об'єкту. Ви можете прикріпити скрипт перетягуванням його із ***Assets*** *на ігровий об'єкт в панелі* ***Hierarchy*** *або через вікно* ***Inspector*** *обраного ігрового об'єкта.*



Скрипти C#

Змінна

Це комірка памяті в якій зберігаються дані.

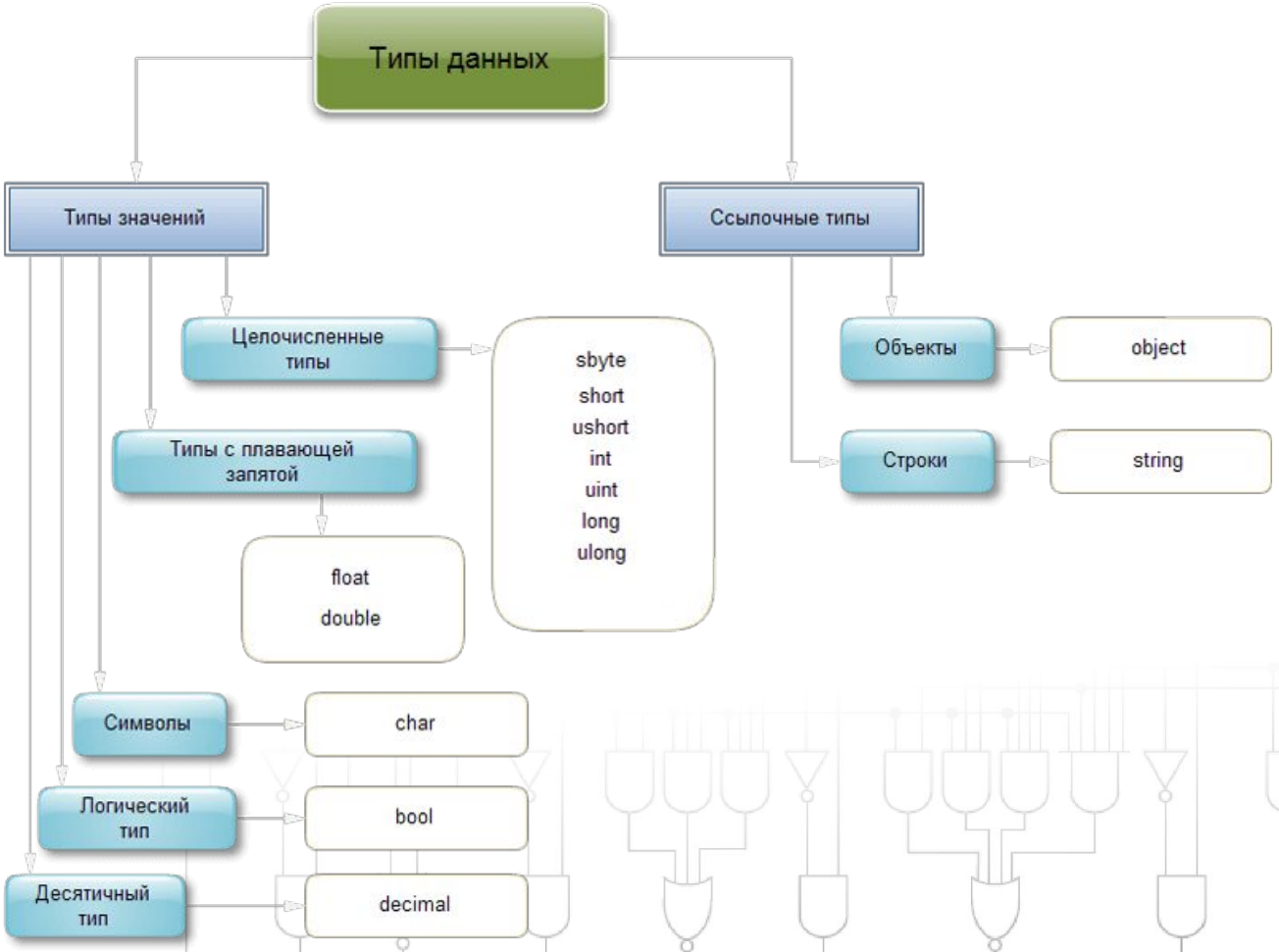
Послідовність дій при роботі зі змінними:

1. int x; - оголошення (декларування);

2. x = 3; - присвоєння (ініціалізація).

Скрипти C#

Типи даних C#



Скрипти C#

Створимо public змінну, що зберігатиме в собі швидкість зміщення об'єкту:

```
 public float speed;
```

Замінемо функцію void Update() на void FixedUpdate():

```
void FixedUpdate () {
    /*~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
    **********Виконуваний код**********
    ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ */
}
```

Скрипти C#

Задамо зміщення об'єкта вліво – вправо при натисканні клавіш A, D на клавіатурі. Для цього скористаємось класом UnityEngine – ***Input.***

`Input.GetKey(KeyCode.A)` - *Повертає true, поки натиснута кнопка A*

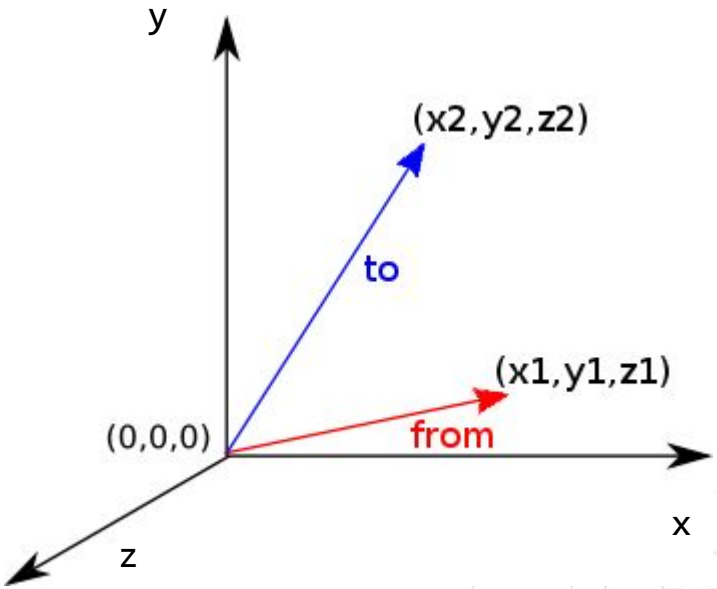
`Input.GetKey(KeyCode.D)` - *Повертає true, поки натиснута кнопка D*

Для реалізації вибору між діями, які ми виконуємо при натисненні кнопки, використаємо умовний оператор if-else.

Скрипти C#

Щоб задати напрямок переміщення скористаємось структурою Vector 3:

```
Vector3(x, y, z)
```



- *Ця структура використовується всередині Unity для передачі 3D-позицій та напрямків. Вона також містить функції для виконання загальних векторних операцій.*

Скрипти C#

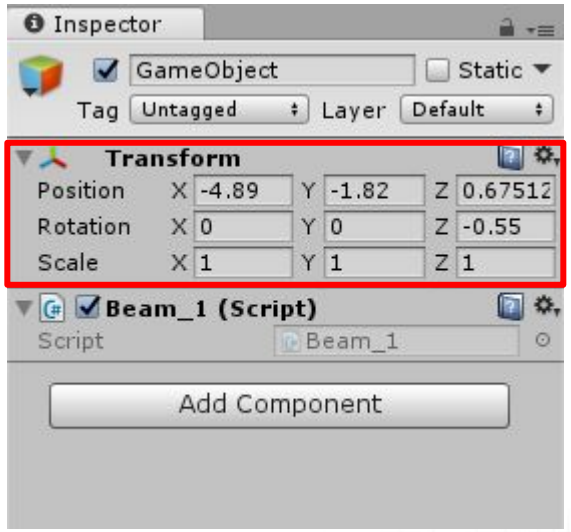
Програма виглядатиме наступним чином:

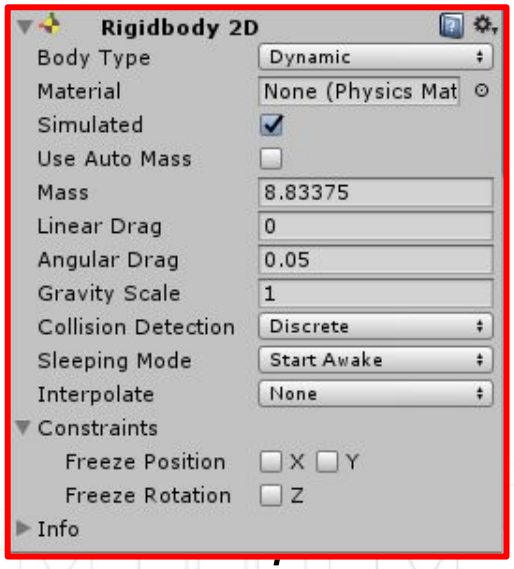
```
void FixedUpdate () {
        if (Input.GetKey(KeyCode.A)) {
            transform.Translate(-speed, 0, 0);
        }
        else if (Input.GetKey(KeyCode.D)) {
            transform.Translate(speed, 0, 0);
        }
        else {
            transform.Translate(0, 0, 0);
        }
    }
```

Скрипти C#

Але на даному етапі ми лише задаємо зміщення по координатним осям, ігровий об'єкт при цьому залишається незмінним.

У редакторі Unity ви змінюєте властивості Компонента використовуючи вікно Inspector.





Але властивості компонентів для керування ігровими об'єктами можназмінювати і за допомогою скриптів.

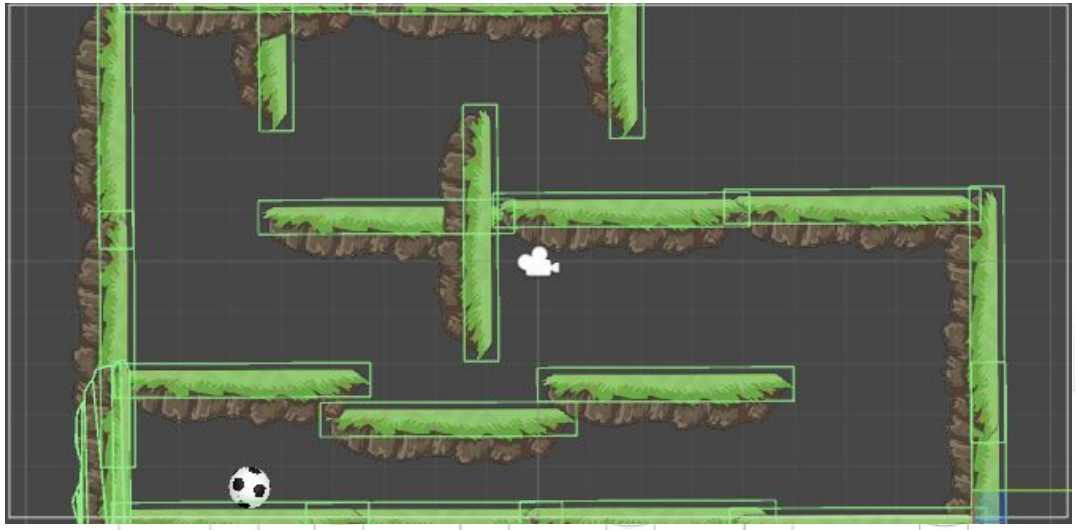
Завдання

- *Самостійно пропишіть зміщення по осі y.*
- *Створіть лабіринт.*

Завдання

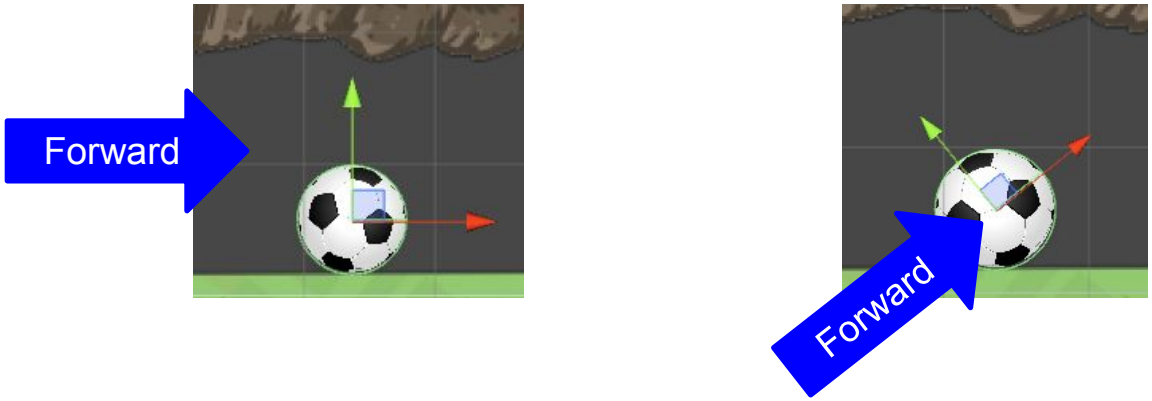
При створенні гри «Лабіринт» для детектування контакту об’єкта зі стінами зробіть наступне:

- *Задайте стінкам колаедри (Poligon/Box).*
- *Ігровому об’єкту задайте колаедри та Rigidbody 2D.*



Завдання

Оскільки напрям руху transform.Translate(Vector3) підвязано до системи координат об’єкта зі скриптом, то при зміні кута нахилу зміняться і напрями.



- *Щоб цього позбутися – слід помістити ігровий об’єкт в порожній об’єкт і скрипт приєднати до нього*
- *Або ж в Inspector-> Rigidbody2D-> Constraints->Freeze Rotation.*