

# Game design



<{Developer**R**/>

# Перша гра



# Повторення

## Що таке ігровий об'єкт в Unity?

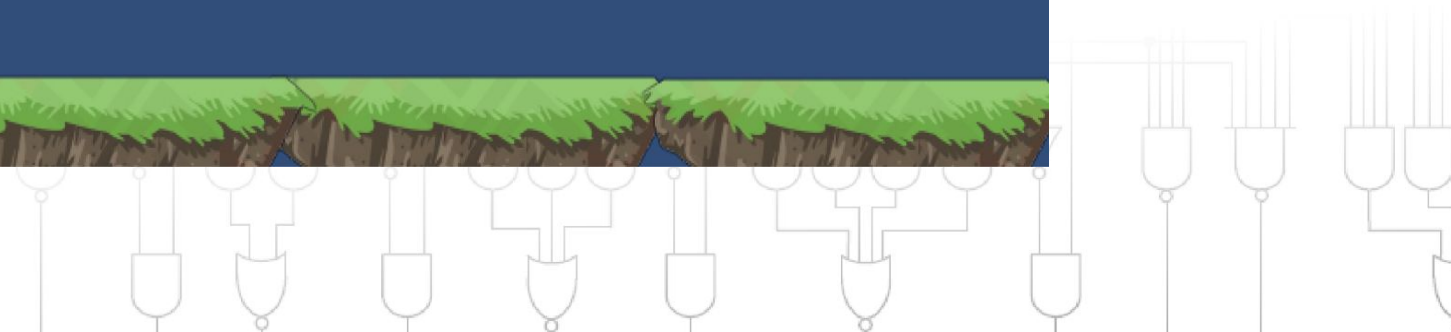
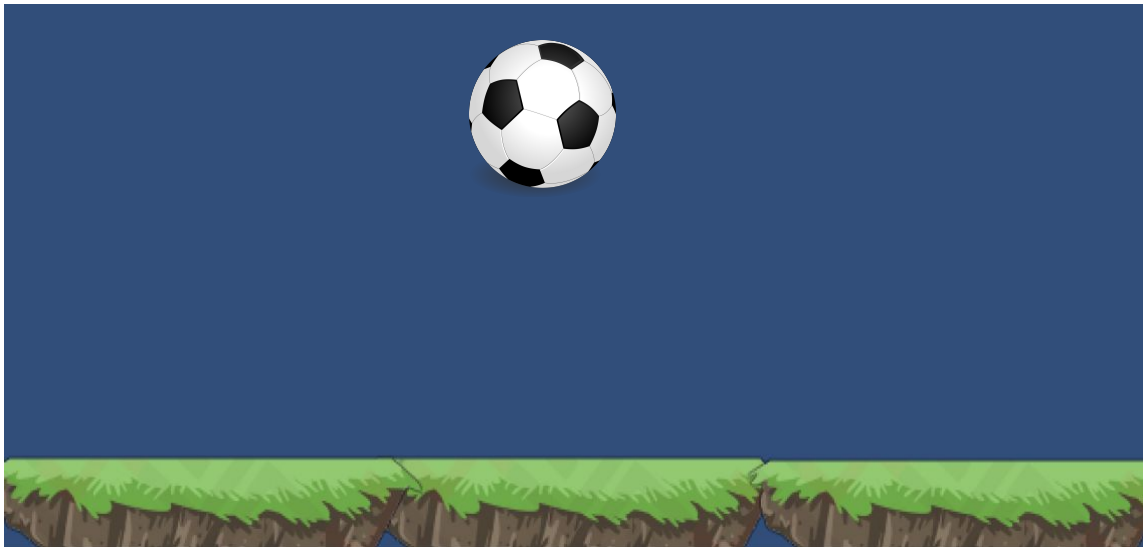
Ігрові об'єкти - це контейнери. Порожня коробка, яка може містити всередині різні елементи, такі як острів з зааданими тінями або фізично коректний автомобіль.



# Повторення

**За що відповідають компоненти Collider?**

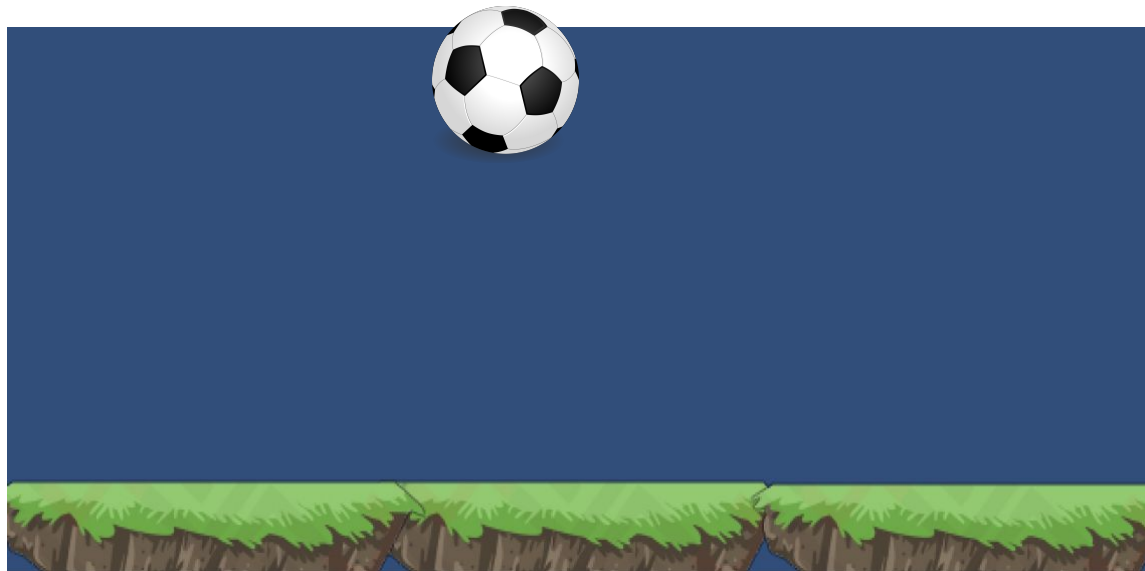
*Collider відповідають за взаємодію ігрових об'єктів. Ці компоненти, фактично, «матеріалізують» об'єкти.*



# Повторення

**Як задати коефіцієнт пружності та силу опору(тертя)?**

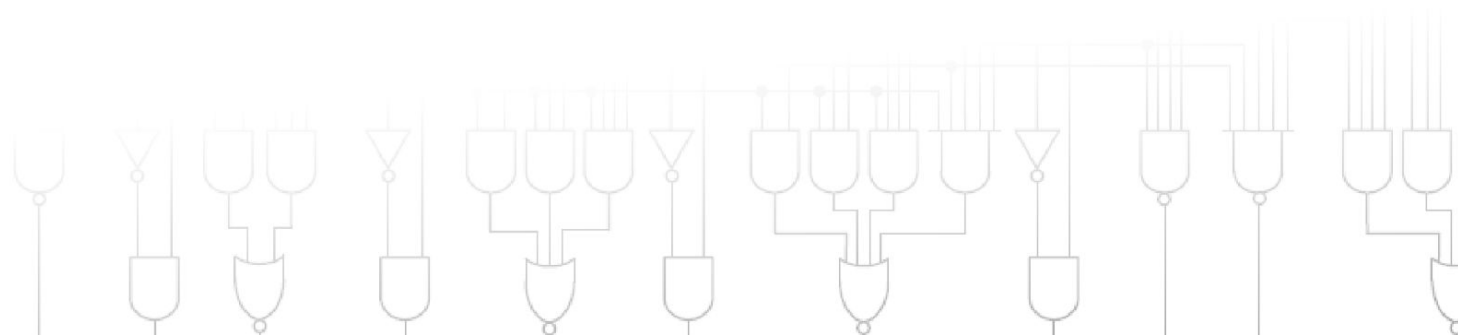
*Ці параметри задаються за рахунок присвоєння ігровим об'єктам параметрів фізичного матеріалу *Physical Material 2D*.*





# Скрипти С#

**Якщо діти на попередньому занятті встигли попрацювати зі скриптами – то скрийте слайди 5-14 включно**



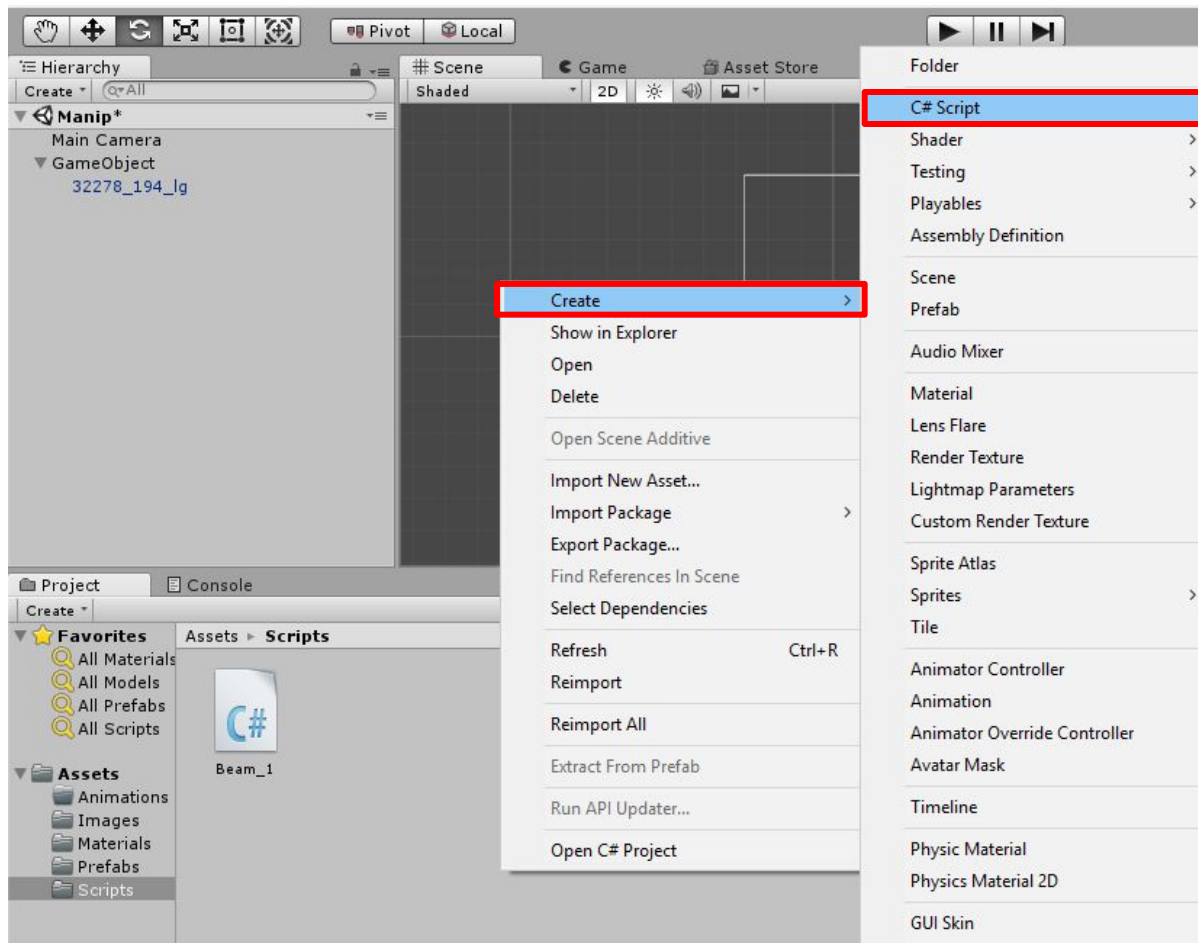
# Скрипти С#

- *Відкрийте проект з минулого уроку.*
- *Створіть горизонтальну поверхню, по якій зможе рухатись ігровий об'єкт. Та задайте всі необхідні колаедри.*



# Скрипти С#

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



# Скрипти С#

*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 () {

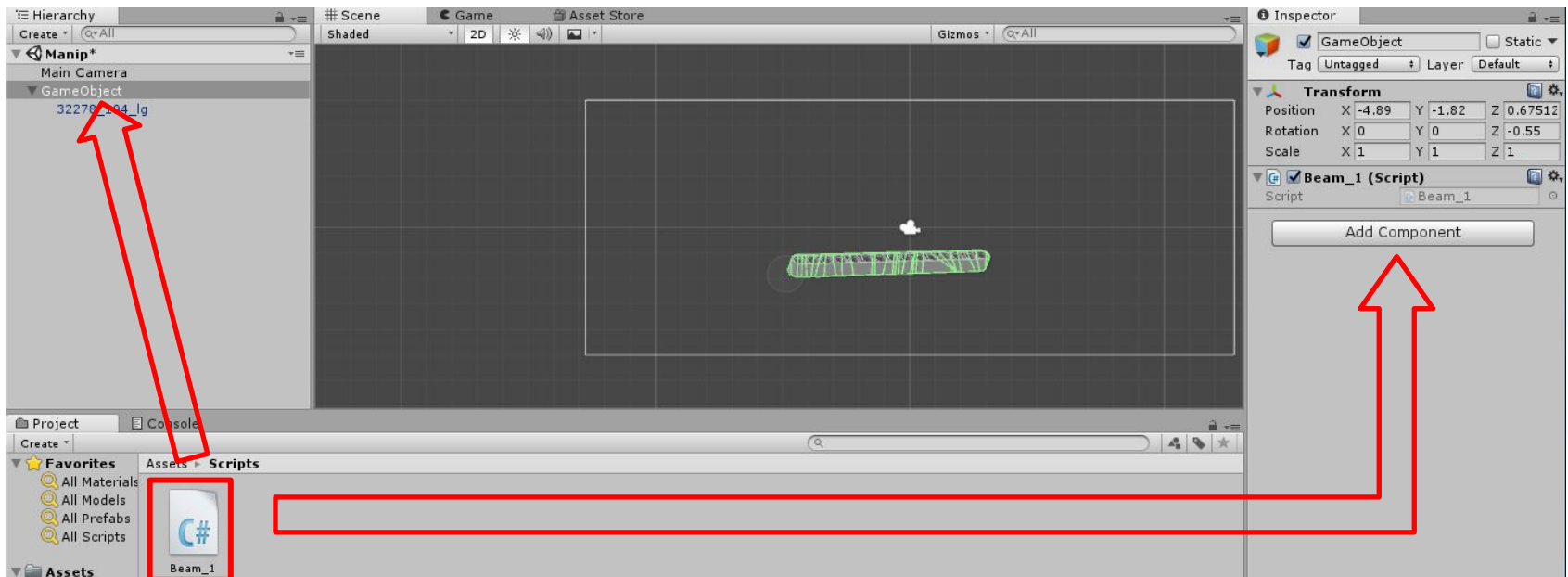
    }

}
```



# Скрипти С#

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 С#

## Змінна

*Це комірка пам'яті в якій зберігаються дані.*

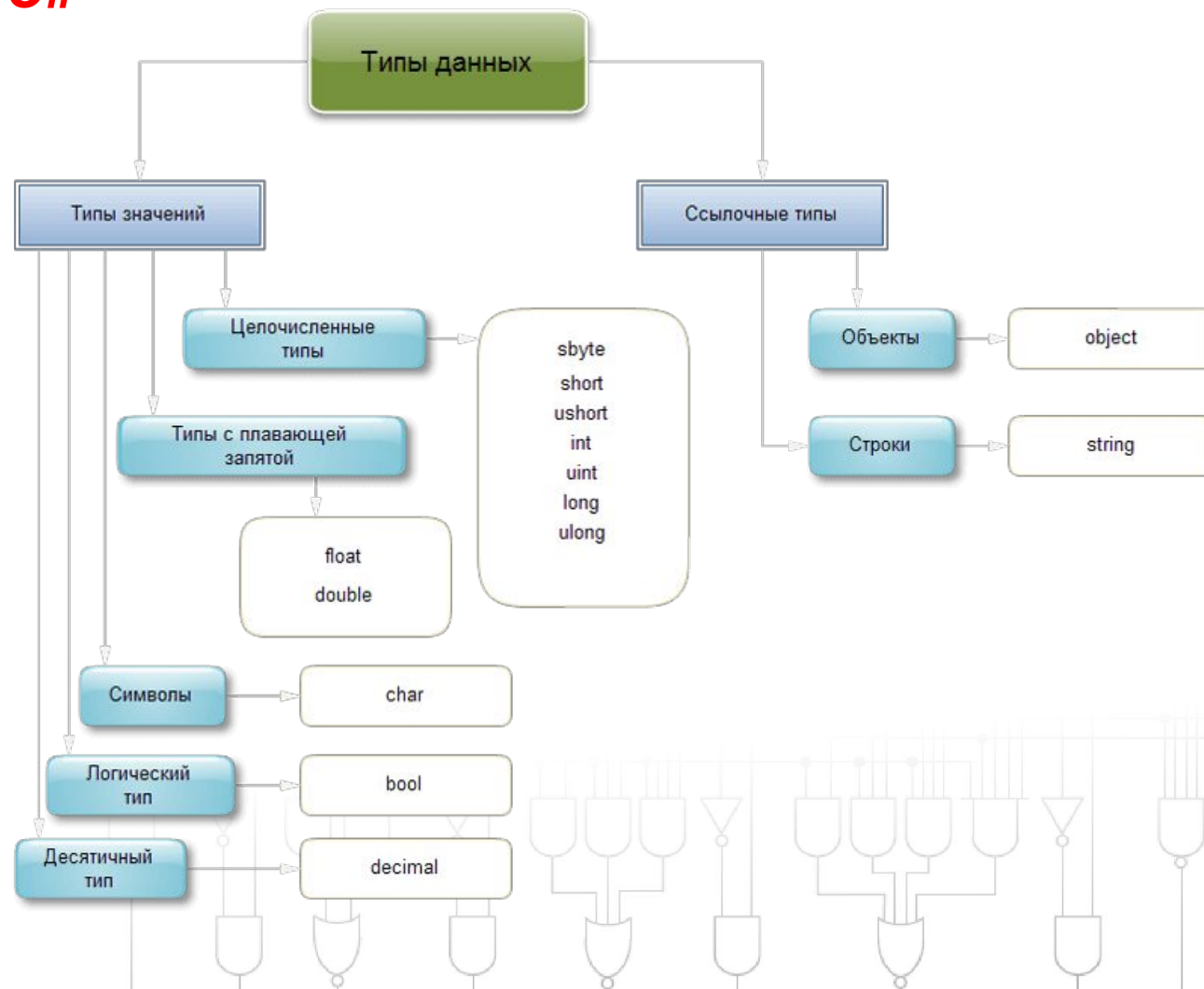


*Послідовність дій при роботі зі змінними:*

- 1. `int x;` - оголошення (декларування);*
- 2. `x = 3;` - присвоєння (ініціалізація).*

# Скрипты С#

## Типы данных С#





# 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.*

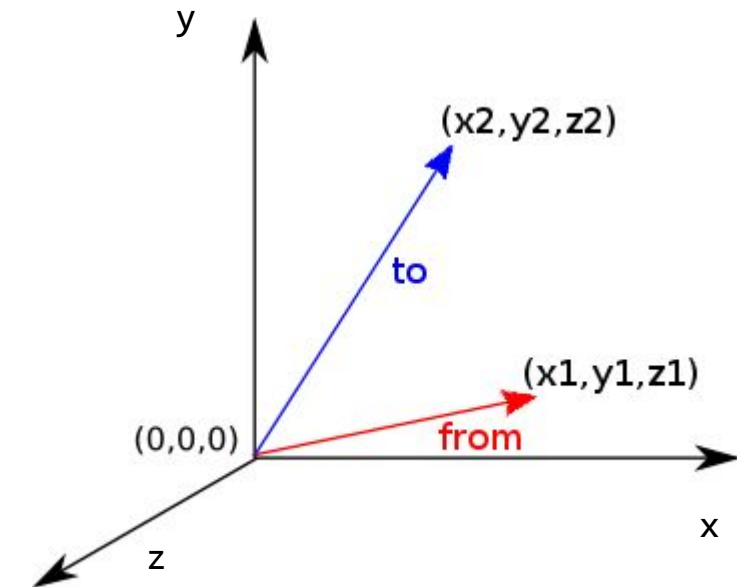
# Скрипти С#

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 С#

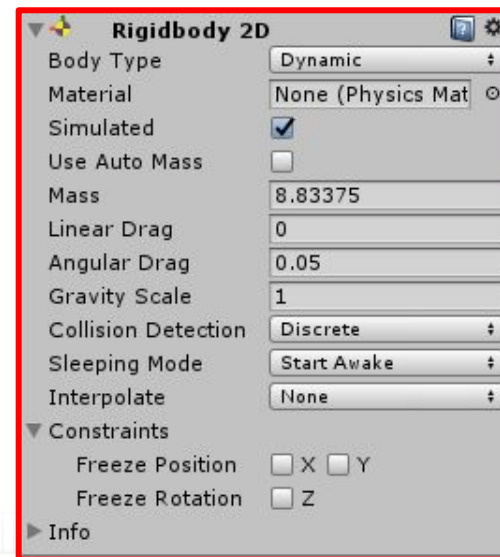
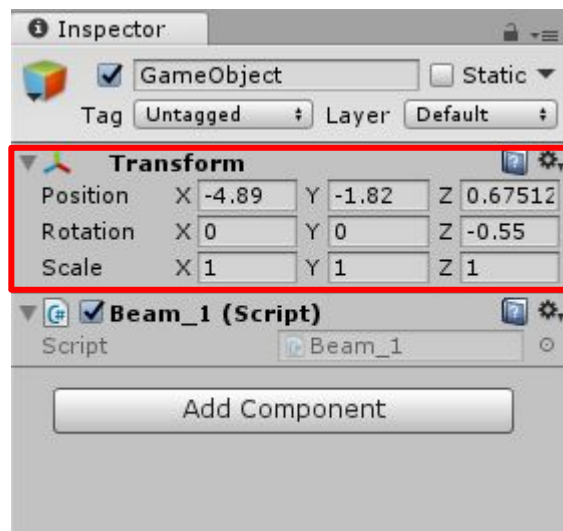
*Програма виглядатиме наступним чином:*

```
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 С#

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

У редакторі Unity ви змінюєте властивості Компонента використовуючи вікно Inspector.

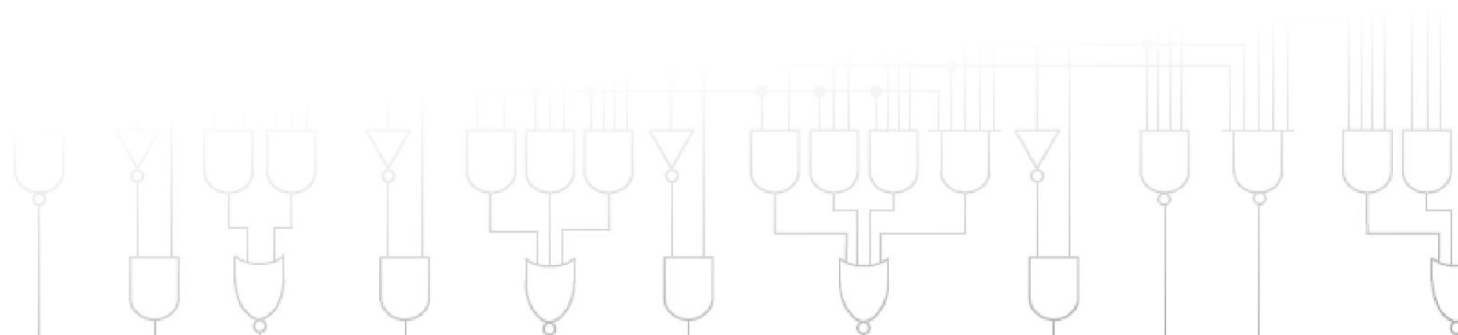


Але властивості компонентів для керування ігровими об'єктами можна змінювати і за допомогою скриптів.



# Завдання

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



# Завдання

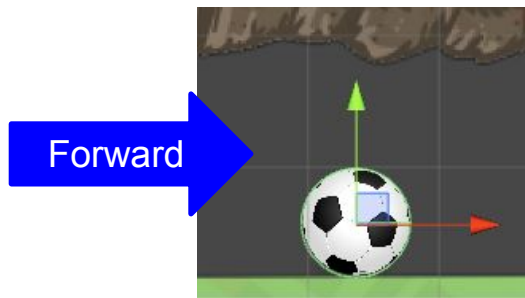
При створенні гри «Лабіринт» для детектування контакту об'єкта зі стінами зробить наступне:

- *Задайте стінкам колаедри (Polygon/Box).*
- *Ігровому об'єкту задайте колаедри та RigidBody 2D.*



# Завдання

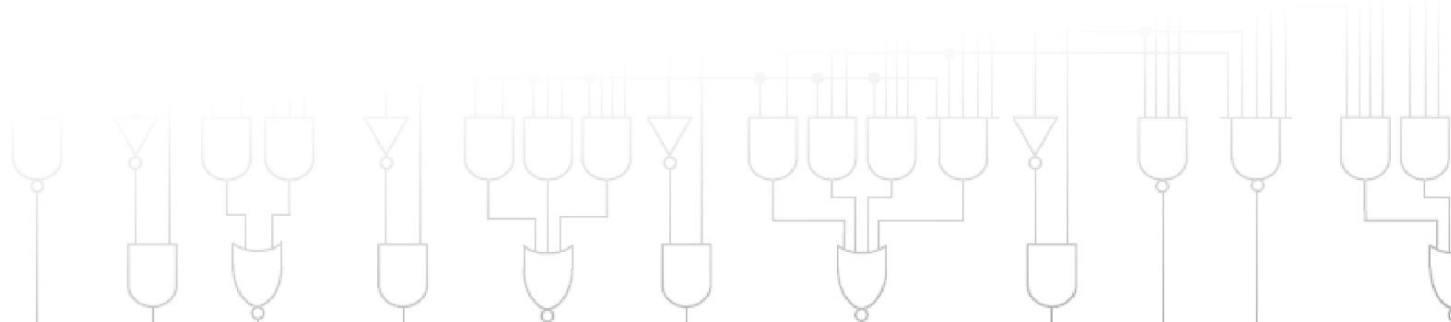
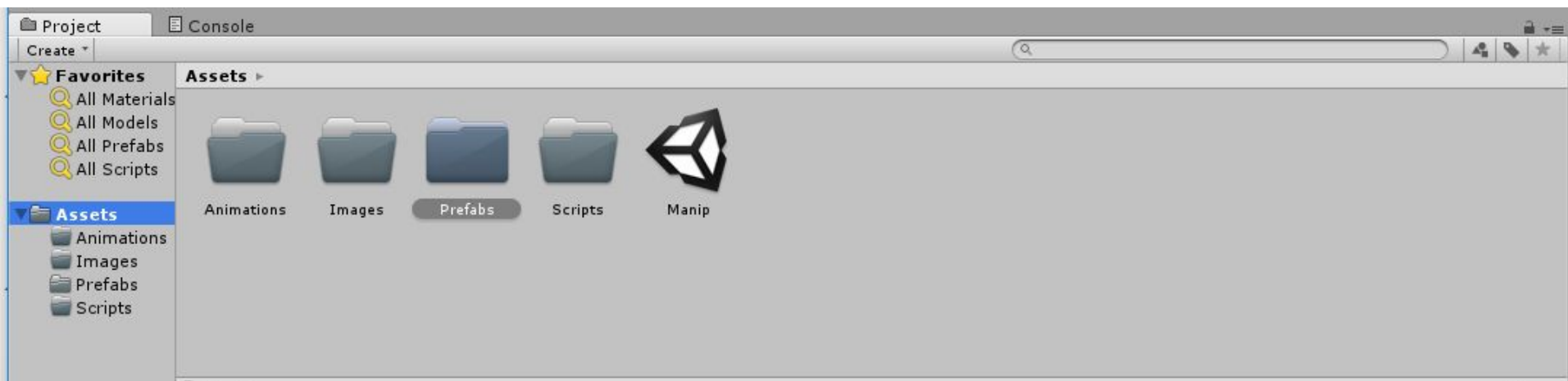
Оскільки напрям руху `transform.Translate(Vector3)` підв'язано до системи координат об'єкта зі скриптом, то при зміні кута нахилу зміняться і напрями.



- 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`.

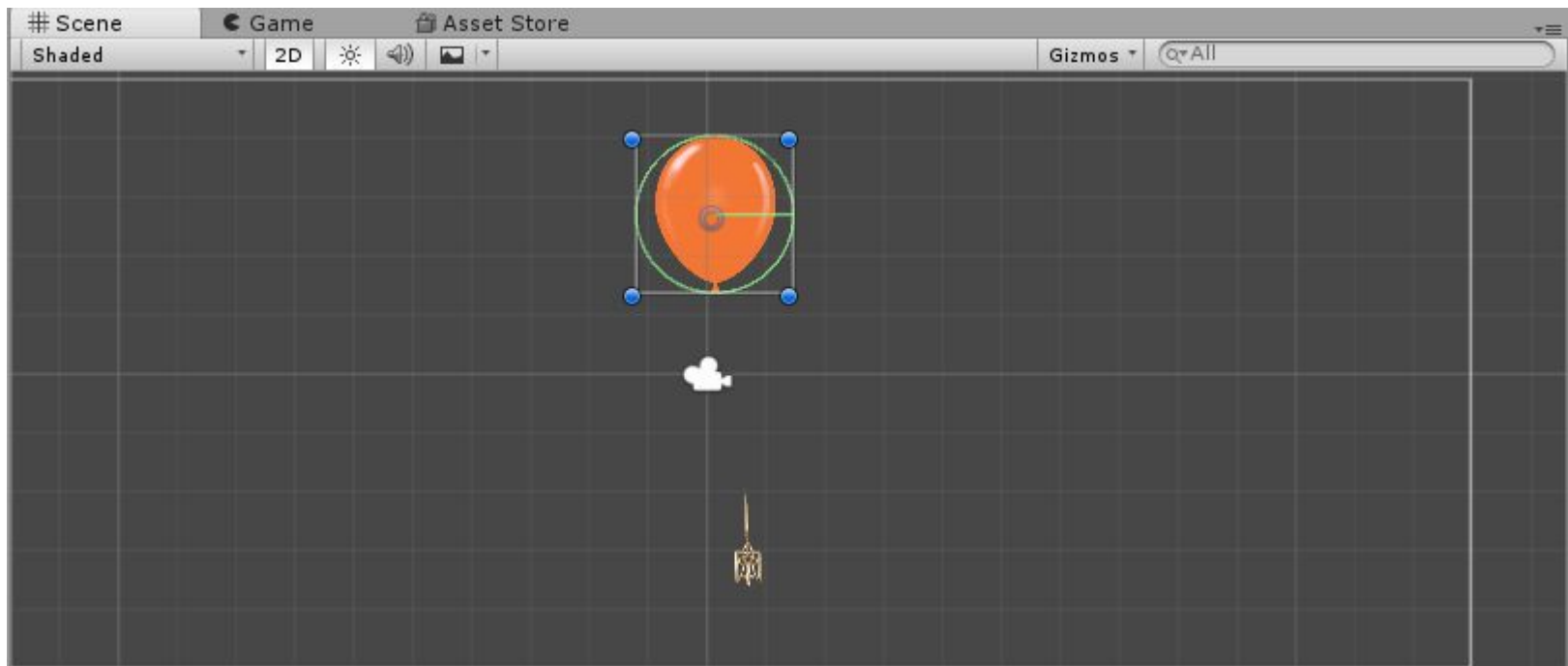
# Перша гра

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



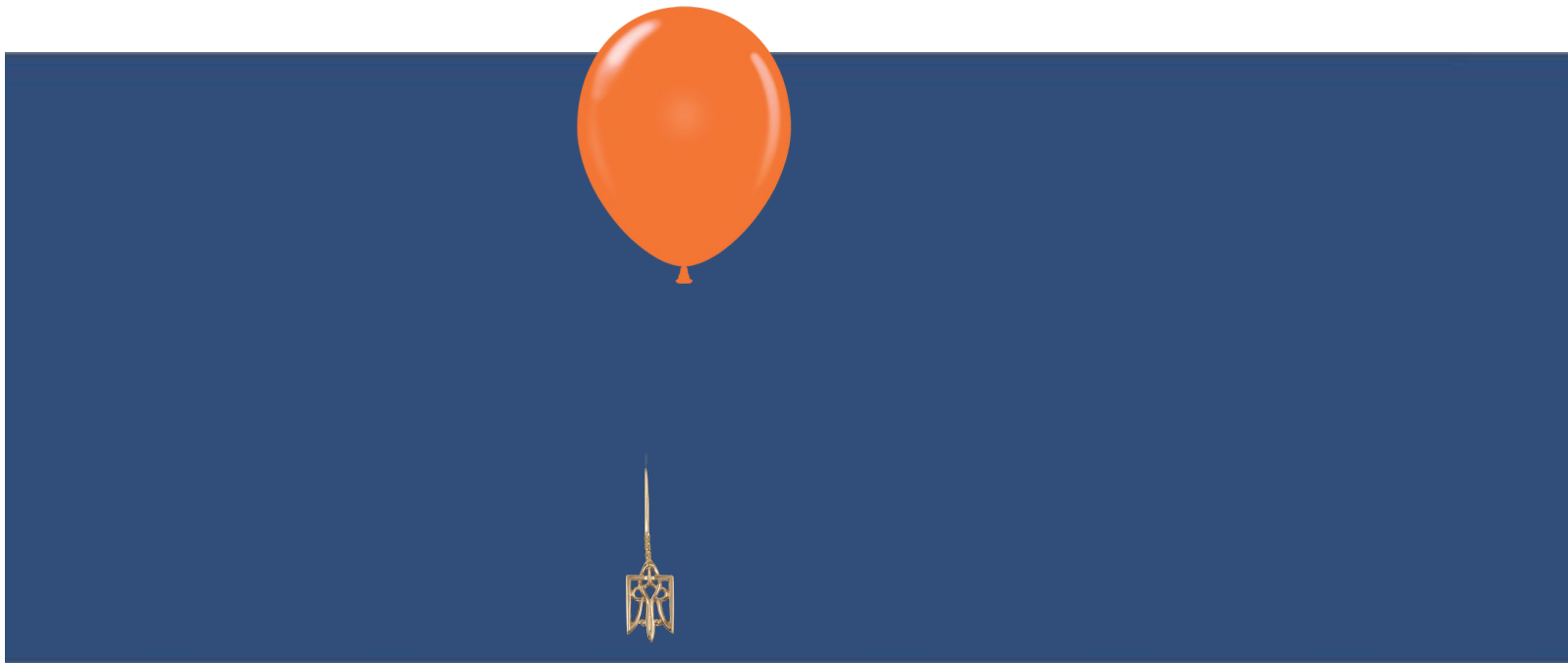
# Перша гра

*Додайте ваші ігрові об'єкти (можна з минулого заняття), в наведеному прикладі – це булавка та кулька:*



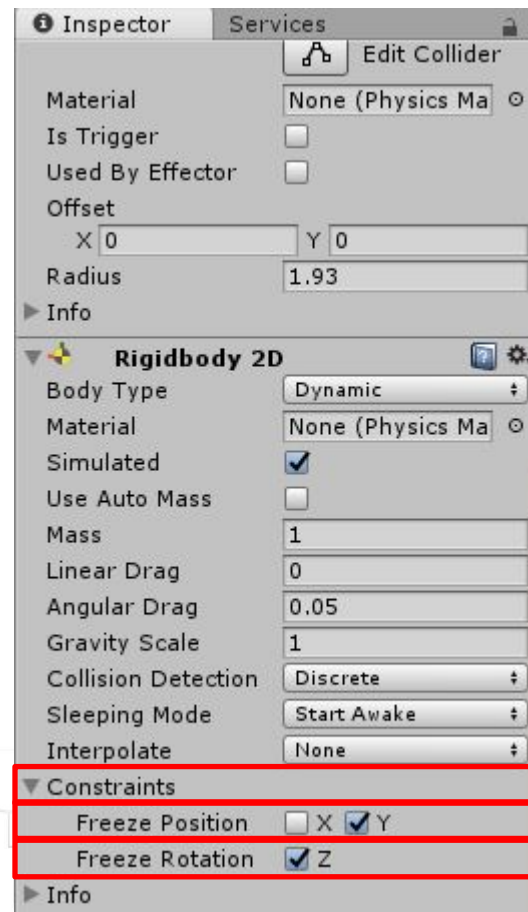
# Перша гра

Задайте ігровим об'єктам основні фізичні параметри *Physics 2D* > *Rigidbody2D* *Physics 2D* > *Polygon Collider(Circle)*. Запустіть і перевірте:



# Перша гра

Нам слід змінити налаштування *Rigidbody2D*, щоб об'єкти не падали під дією гравітації, а керувались скриптами:



# Перша гра

*В папці Scripts створюємо 2 скрипти для кульки та голки:*



*Нам треба, щоб скрипт Baloon задавав повільне падіння кульки згори вниз, а Needle задавав зміщення голки вліво – вправо.*



# Перша гра

*В скрипті Valoon задамо public змінну, що контролюватиме швидкість падіння:*

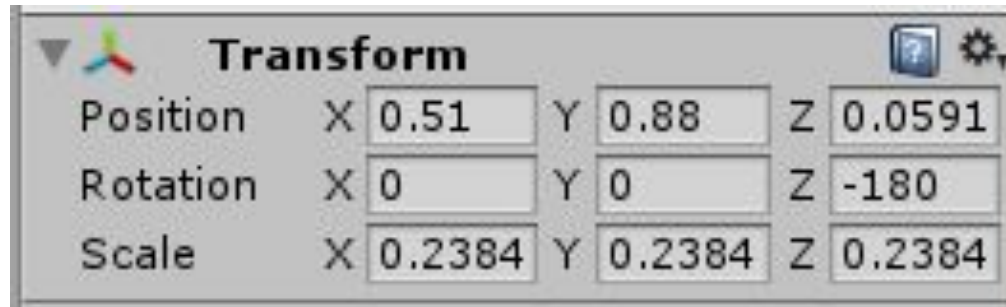
```
public float Fallspeed;
```

*Тіло програми змінимо на void FixedUpdate () і додамо в нього зміну положення по осі y:*

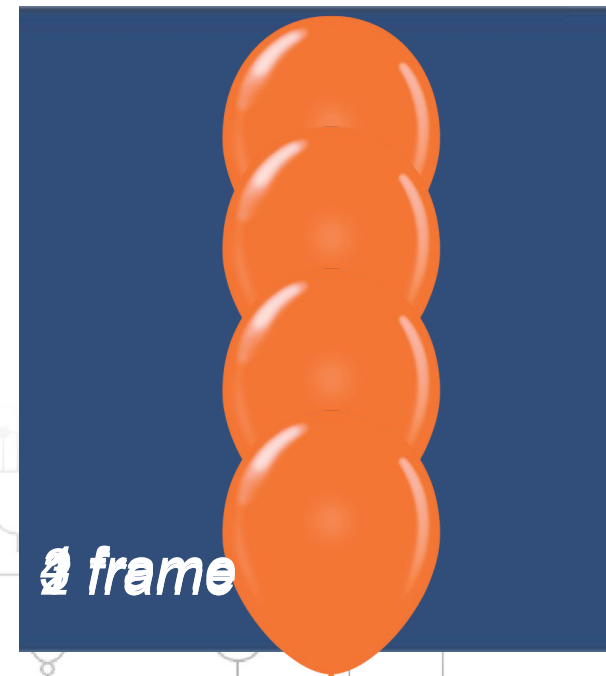
```
void FixedUpdate()  
{  
    transform.Translate(0, -Fallspeed, 0);  
}
```

# Перша гра

Метод `transform` відповідає за зміну відповідних параметрів:

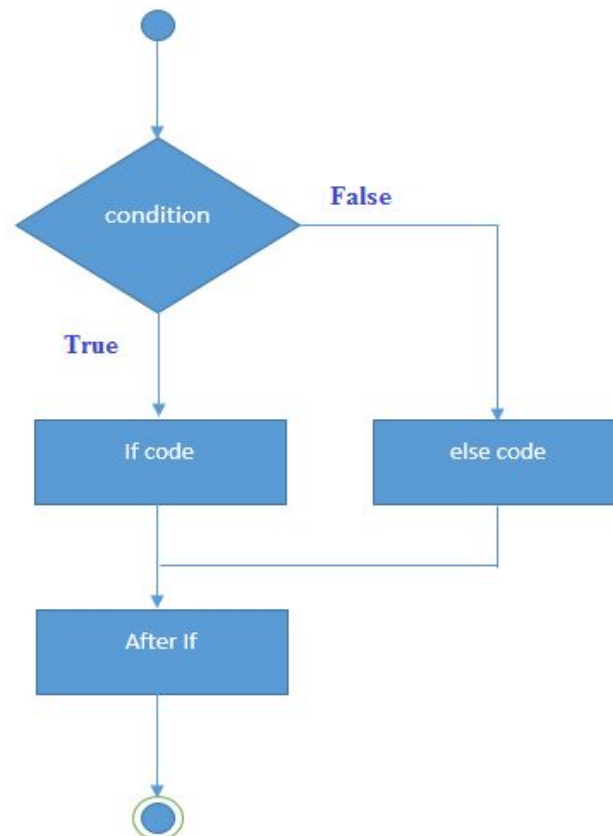


Зокрема `transform.Translate` переносить об'єкт в заданому напрямі на задану кількість пікселів, а оскільки метод `FixedUpdate` викликається кожного фрейму, то фактично – це зміщення за фрейм



# Перша гра

В скрипті Needle задамо зміщення (`transform.Translate`) по натисненні кнопок A, D, для цього слід використати умовний оператор `if - else`:



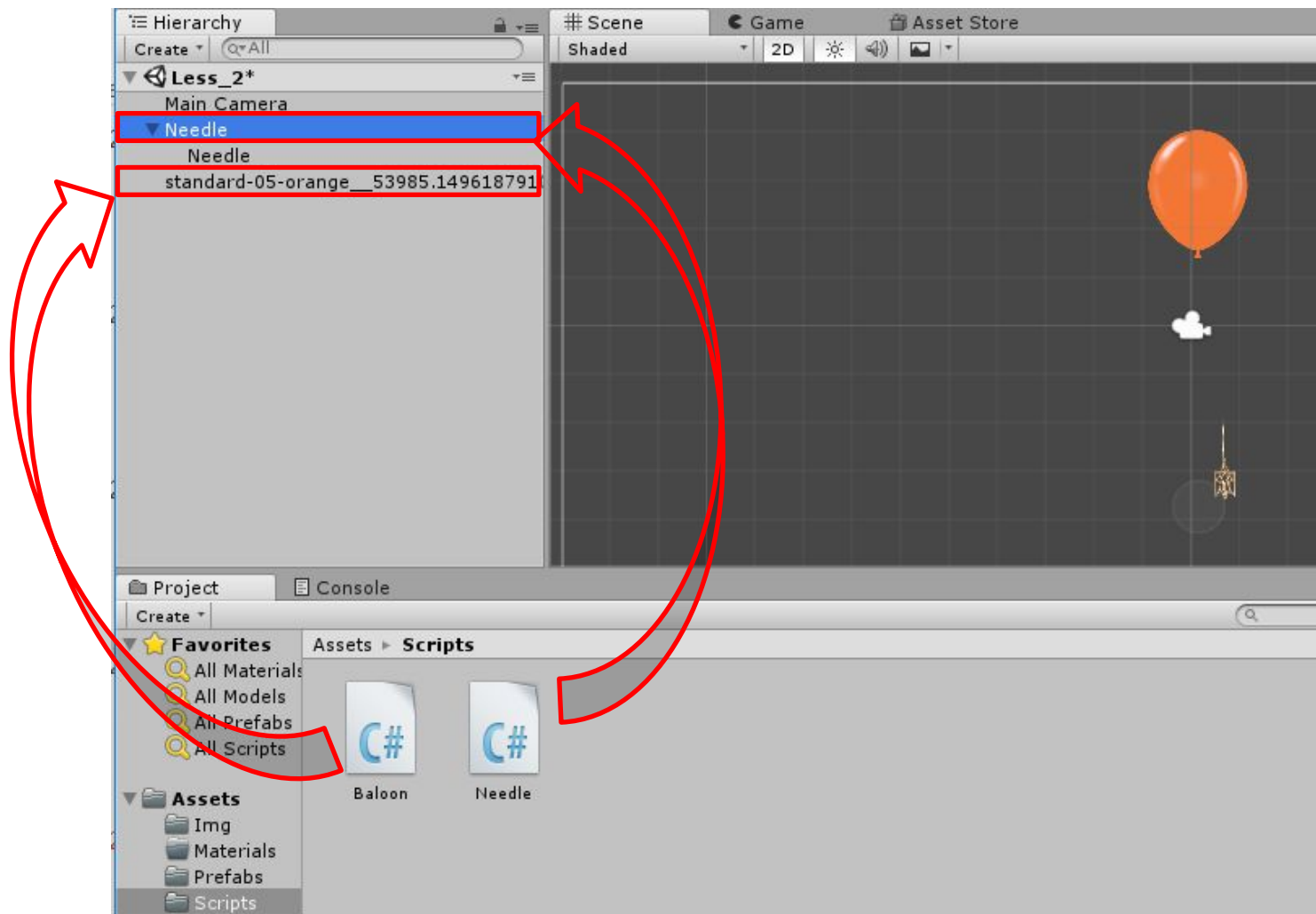
# Перша гра

*В скрипті дана конструкція виглядатиме так (не забудьте проініціалізувати змінну `public float speed` до `void Setup()`):*

```
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);
}
```

# Перша гра

*Під'єднайте скрипти до відповідних об'єктів:*



# Перша гра

*З префабу перетягніть ще 5-10 кульок, розташуйте їх над камерою на різній висоті, але так, щоб вони по горизонталі не виходили за межі камери.*

