Лекція №3-4

Вступ до 2D фізики в Unity

2025

План

1.Введення в фізику Unity. FixedUpdate
2. Компоненти Collider2D/Rigitbody2D
5. Переміщення за допомогою Rigitbody2D (MovePosition/AddForce/Velocity)
6. Фізичний матеріал PhysicsMaterial2D

Введення у фізику Unity та FixedUpdate

Фізика в Unity — це симуляція реальних законів руху (гравітація, зіткнення, сили) в ігровому світі. Щоб ця симуляція була стабільною та точною, вона працює на окремому циклі, незалежному від графічного оновлення.

Метод FixedUpdate()

Кожен скрипт у Unity має два основні цикли оновлення:
Update() викликається раз на кадр (framerate-dependent). Використовується для введення даних гравця, анімацій та нефізичної логіки.
FixedUpdate() викликається з фіксованою частотою, незалежно від частоти кадрів (framerate-independent). ***Це основний цикл для всіх фізичних розрахунків.***

Чому FixedUpdate?
Фізичні розрахунки повинні відбуватися з постійною частотою (наприклад, 50 разів на секунду), щоб уникнути помилок в інтеграції (наприклад, об'єкт може "проскочити" крізь інший об'єкт на низькій частоті кадрів).

Ключове правило. Усі команди, що стосуються фізики (наприклад, застосування сили, зміна швидкості Rigidbody), повинні викликатися всередині FixedUpdate().

Компоненти Collider2D та Rigidbody2D

Для роботи з 2D фізикою в Unity необхідно два основні компоненти:

- **1. Rigidbody2D (Тверде тіло 2D)**
- **2. Collider2D (Колайдер 2D)**

Rigidbody2D (Тверде тіло 2D)

Цей компонент перетворює ігровий об'єкт на фізичне тіло, яким керує фізичний рушій Unity.

- **Функціональність.** Дозволяє об'єкту реагувати на сили (гравітацію, імпульс) і взаємодіяти з іншими фізичними об'єктами.
- **Властивості:**
  - **Mass (Маса) -** впливає на інерцію та силу, необхідну для прискорення.
  - **Drag (Опір) -** повітряний опір, який уповільнює об'єкт.
  - **Gravity Scale (Масштаб гравітації) -** множник для світової сили тяжіння. 1 – нормальна гравітація, 0 – відключена, -1 – зворотна гравітація.
  - **Body Type (Тип тіла):**
    - **Dynamic - с**тандартний тип. Об'єкт, керований силами та скриптами.
    - **Kinematic - к**ерується тільки скриптами (наприклад, рухома платформа). На нього не діє гравітація.
    - **Static** - не рухається і не реагує на сили. Ідеально підходить для нерухомих елементів світу (підлога, стіни).

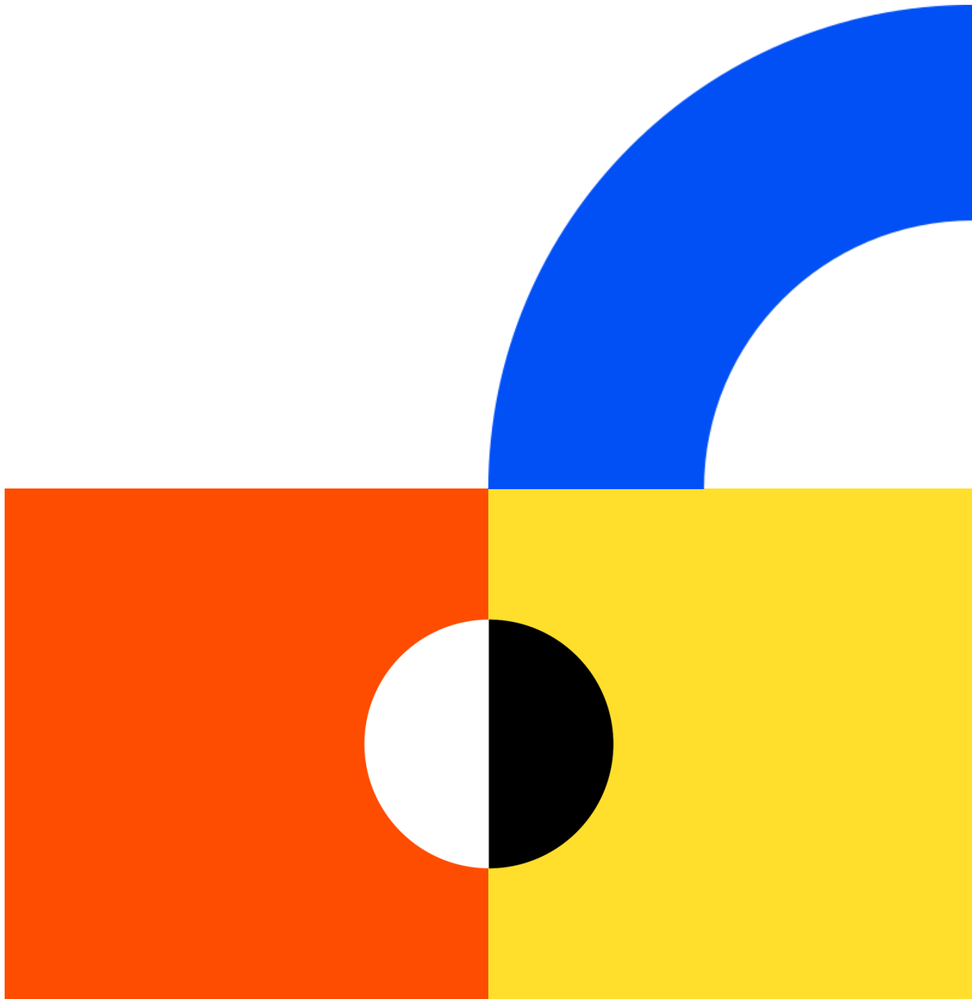
Collider2D (Колайдер 2D)

Цей компонент визначає форму об'єкта для фізичних зіткнень. Об'єкт не може зіткнутися з іншим об'єктом, якщо не має колайдера.

- **Поширені типи:** BoxCollider2D, CircleCollider2D, PolygonCollider2D.
- **Властивість Is Trigger:**
  - Якщо **вимкнено** (false) - колайдер є твердим тілом і запобігає проникненню інших об'єктів.
  - Якщо **увімкнено** (true) - колайдер стає тригером. Він не зупиняє рух, але викликає події (OnTriggerEnter2D, OnTriggerExit2D), які можна використовувати для збору предметів, виявлення гравця тощо.

Переміщення за допомогою Rigidbody2D

Для переміщення фізичних об'єктів **ніколи не можна** використовувати пряму зміну transform.position. Це призводить до нестабільності фізичного рушія. Замість цього використовуйте методи Rigidbody2D



MovePosition(Vector2 position)

Для точного переміщення об'єкта до нової позиції. Unity розраховує рух між поточною та цільовою позиціями під час фізичного циклу, забезпечуючи коректне зіткнення.

Підходить для руху, керованого користувачем або ШІ, де потрібна висока точність.

```
void FixedUpdate() {
// Рух вправо
 Vector2 targetPosition = rb.position + Vector2.right * speed
* Time.fixedDeltaTime;
rb.MovePosition(targetPosition);
}
```

AddForce(Vector2 force, ForceMode2D mode)

Застосовує силу до Rigidbody, змушуючи його прискорюватися. Це імітує реальні фізичні ефекти (поштовх, вибух, постійний двигун). **Підходить для с**трибків, ракетних двигунів, зіткнень.

- **ForceMode2D.Force -** стандартна сила. Застосовується постійно. Рух залежить від маси.
- **ForceMode2D.Impulse** - миттєвий удар (імпульс). Ігнорує час кадру. Ідеально для стрибків або одноразових поштовхів.

```
void Jump() {
// Застосовуємо імпульс вгору rb.AddForce(Vector2.up *
jumpForce, ForceMode2D.Impulse);
}
```

Velocity

Пряма установка або читання поточної швидкості Rigidbody. Використовується, коли вам потрібно миттєво змінити швидкість або контролювати максимальну швидкість об'єкта.

```
void FixedUpdate(){
 float horizontalInput = Input.GetAxis("Horizontal");
//Встановлюємо нову швидкість по X, зберігаючи швидкість по Y
rb.velocity = new Vector2(horizontalInput * maxSpeed, rb.velocity.y);
}
```

Фізичний матеріал PhysicsMaterial2D

PhysicsMaterial2D (Фізичний матеріал 2D) — це окремий актив (Asset), який можна створити в Unity (Create -> 2D -> Physics Material 2D). Він використовується для визначення того, як об'єкт взаємодіє з іншими об'єктами під час зіткнень.

Основні властивості

Friction (Тертя):

- Визначає, наскільки сильно об'єкти сповільнюються, коли вони ковзають один по одному.
- Діапазон від 0 (повне ковзання, як лід) до 1 (високе тертя, як гума).

Bounciness (Пружність):

- Визначає, наскільки сильно об'єкт відскочить після зіткнення.
- Діапазон від 0 (не відскакує) до 1 (ідеальний відскок, втрата енергії відсутня).

Застосування

Ви створюєте цей матеріал і просто перетягуєте його на поле **Material** у компоненті **Collider2D** ігрового об'єкта.

Приклади використання:

- Створення **слизького льоду** (Friction = 0.0, Bounciness = 0.0).
- Створення **батута** (Friction = 0.4, Bounciness = 0.9).

Ключові моменти, які потрібно запам'ятати:

- Використовуйте **FixedUpdate()** для всіх фізичних розрахунків.
- **Rigidbody2D** робить об'єкт фізичним.
- **Collider2D** дає об'єкту форму для зіткнення.
- **MovePosition**, **AddForce** та **Velocity** — це правильні способи переміщення Rigidbody.
- **PhysicsMaterial2D** контролює тертя та пружність при зіткненнях.

Trigger (Collider2D) та обробка взаємодій

Взаємодія між об'єктами в Unity реалізується за допомогою **Collider2D** та **Rigidbody2D**.

Колайдери визначають фізичні межі об'єкта. Вони потрібні не лише для зіткнень, але й для запуску подій. Кожен 2D-об'єкт, що бере участь у фізиці, повинен мати компонент Collider2D (наприклад, BoxCollider2D, CircleCollider2D).

Режим 'Is Trigger'

Колайдер може працювати у двох режимах:

1. **Зіткнення (Collision).** Якщо опція **Is Trigger вимкнена**, об'єкт є твердим. Фізичний рушій запобігає проникненню інших твердих об'єктів.

2. **Тригер (Trigger).** Якщо опція **Is Trigger увімкнена**, колайдер стає невидимою "зоною". Об'єкти можуть проходити крізь нього, але фізичний рушій генерує події (івенти), коли вони входять, перебувають або виходять із цієї зони.

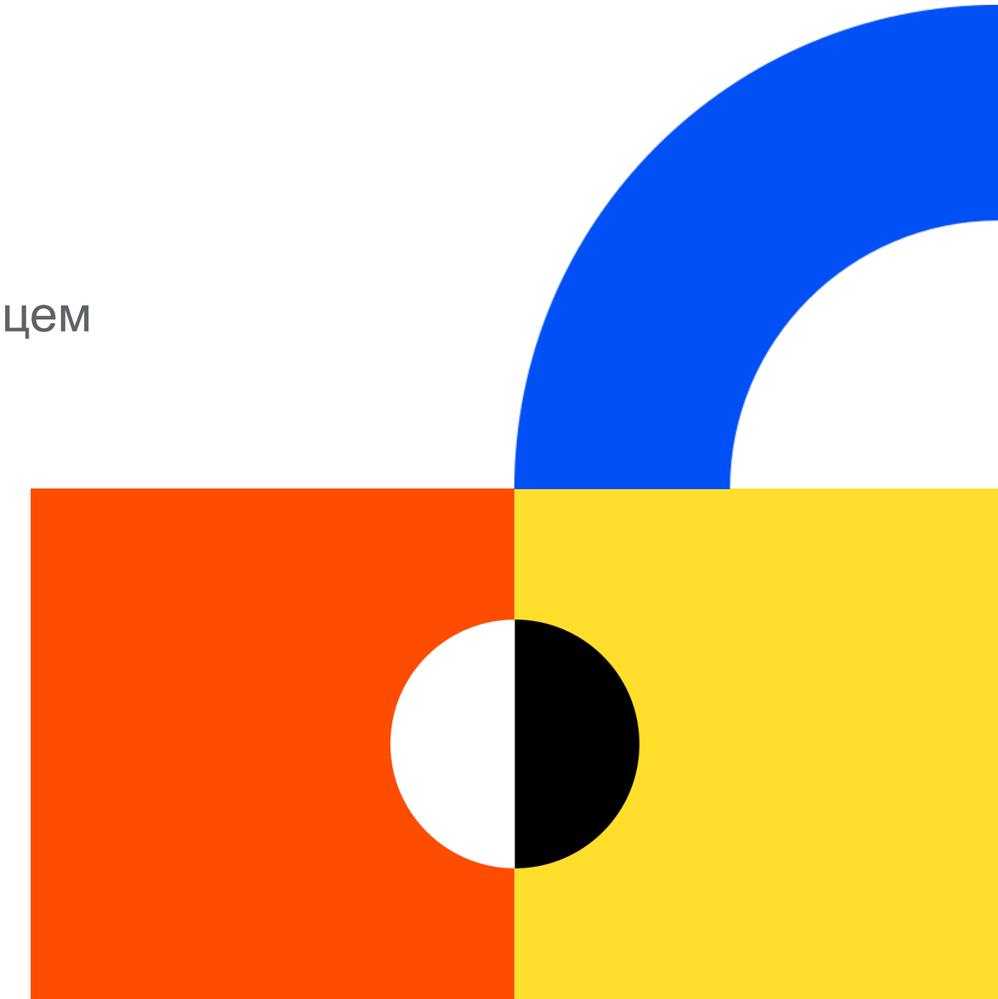
Івенти OnCollision та OnTrigger

Unity надає три основні пари функцій (івентів) для обробки взаємодії. Вони повинні бути реалізовані у скриптах, прикріплених до одного з об'єктів, що взаємодіють.

Тип події	Режим колайдера	Призначення
OnCollisionEnter2D	Collision (Зіткнення)	Об'єкти стикаються. Потрібен Rigidbody2D на обох або на одному з них.
OnTriggerEnter2D	Trigger (Тригер)	Об'єкт входить у зону. Використовується для збору предметів, зон переходу, перевірок.
OnCollisionStay2D	Collision	Об'єкти продовжують стикатися.
OnTriggerStay2D	Trigger	Об'єкт перебуває всередині зони.
OnCollisionExit2D	Collision	Об'єкти розходяться після зіткнення.
OnTriggerExit2D	Trigger	Об'єкт виходить із зони.

Приклад використання OnTriggerEnter2D

```
using UnityEngine;
public class Collectable : MonoBehaviour {
    private void OnTriggerEnter2D(Collider2D other)
    {
        // Перевіряємо, чи об'єкт, який увійшов у тригер, є гравцем
        if (other.CompareTag("Player")){
            Debug.Log("Гравець зібрав бонус!");
             // Тут можна додати логіку нарахування очок
            // GameManager.AddScore(100);
            // Знищуємо об'єкт (бонус)
            Destroy(gameObject);
        }
    }
}
```



Зміна сцен в Unity: SceneManager

Гра, як правило, складається з кількох **сцен (Scenes)**: головне меню, рівень 1, рівень 2, … . Для переходу між ними використовується клас **SceneManager**.

Перш ніж можна буде перемикатися між сценами, їх потрібно **додати до налаштувань збірки (Build Settings)**.

- Відкрийте **File -> Build Settings...**
- Перетягніть необхідні файли сцен у вікно. Unity присвоїть кожній сцені числовий індекс (починаючи з 0).

SceneManager

Зміна сцен

Для роботи з SceneManager потрібно додати простір імен: using UnityEngine.SceneManagement;

Ключові методи:

Метод	Призначення
SceneManager.LoadScene(int index)	Завантажує сцену за її індексом у Build Settings (наприклад, 0 для головного меню).
SceneManager.LoadScene(string name)	Завантажує сцену за її назвою (наприклад, "Level_1").

Приклад скрипта переходу

```
using UnityEngine;
using UnityEngine.SceneManagement; // Обов'язково для роботи з SceneManager

public class SceneLoader : MonoBehaviour{
    // Метод для переходу на наступний рівень за назвою
    public void LoadNextLevelByName(string sceneName) {
        SceneManager.LoadScene(sceneName);
    }
    // Метод для перезапуску поточної сцени
    public void RestartCurrentScene() {
        // Отримуємо індекс поточної сцени
        int currentSceneIndex = SceneManager.GetActiveScene().
        // Завантажуємо сцену за цим індексом
        SceneManager.LoadScene(currentSceneIndex);
    }
}
```