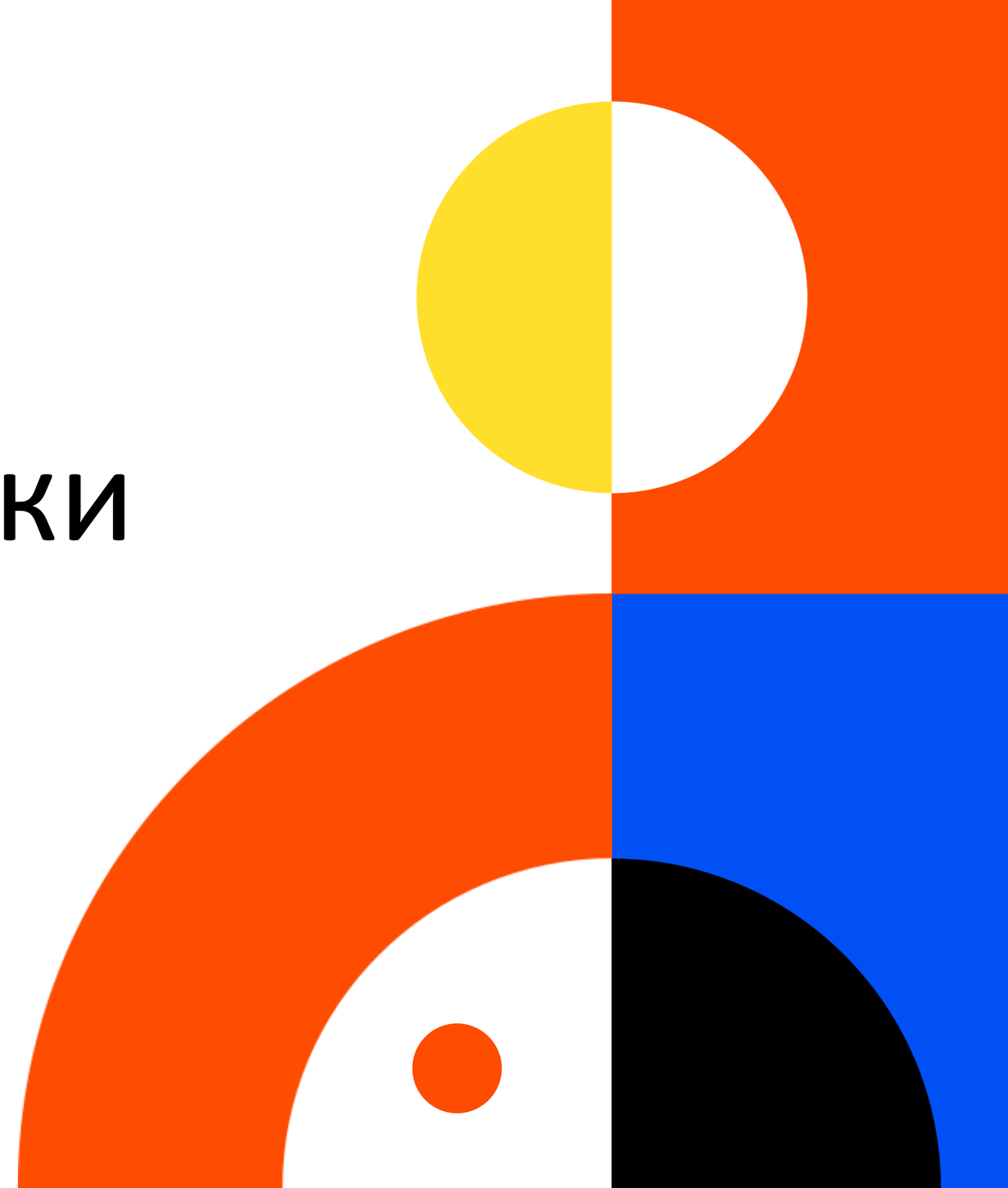
Лекція №4

Основи розробки 2D-ігор в Unity

2025



План

1. Trigger (Collider2D) та обробка взаємодій
2. Зміна сцен в Unity: SceneManager
3. Створення рівня за допомогою TileMap
4. Анімації в Unity. Animator

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.

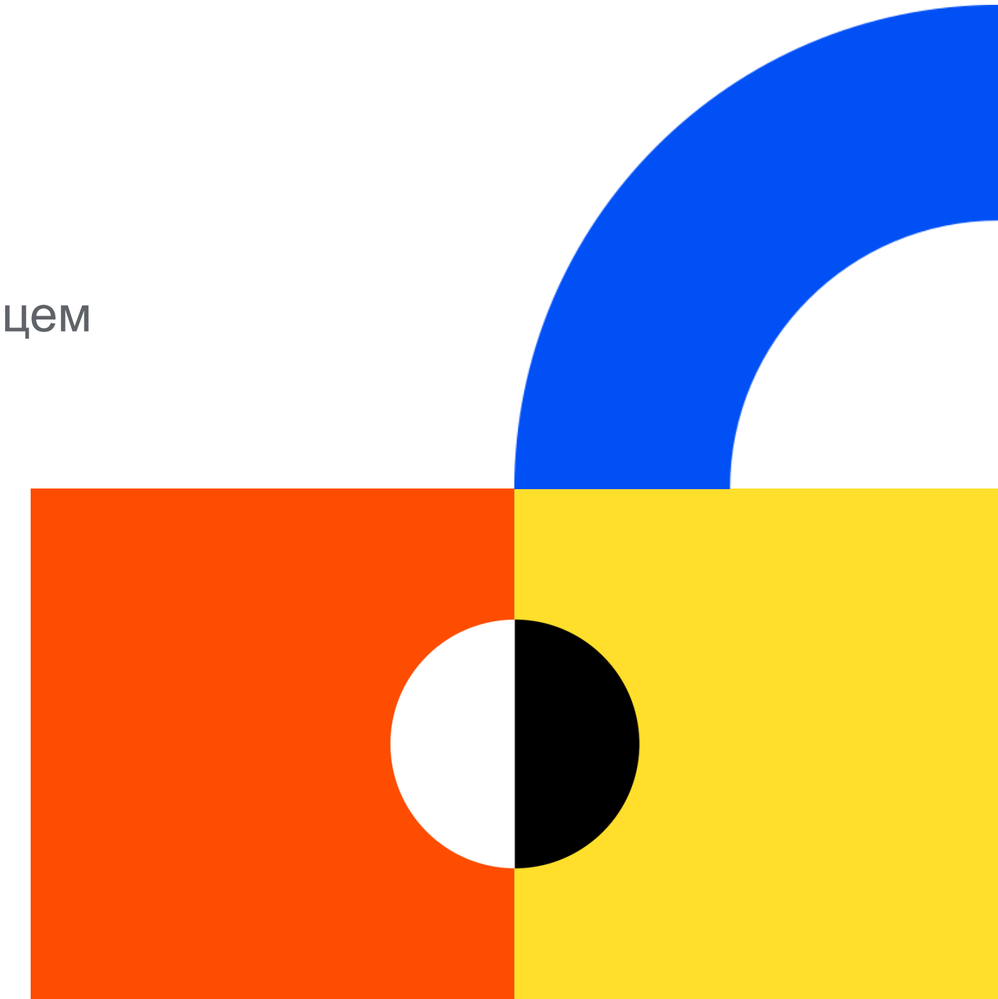
Івенти OnCollision та OnTrigger

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;

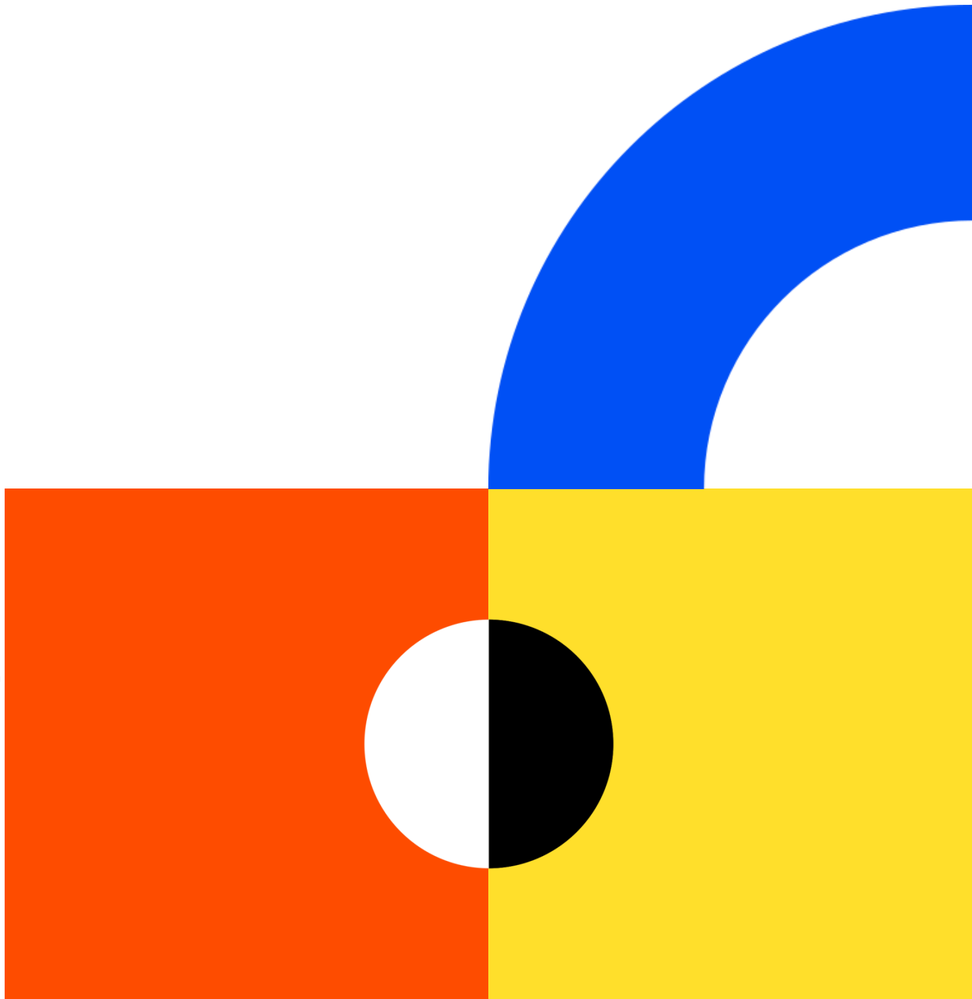
Ключові методи:

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);
    }
}
```



Створення рівня за допомогою TileMap

TileMap (Тайлова Карта) — це інструмент Unity, спеціально розроблений для швидкого створення 2D-рівнів, особливо для платформерів та ізометричних ігор, використовуючи маленькі зображення-плитки (тайли).

Основні компоненти

Grid (Сітка) - базовий об'єкт, який створює сітку координат у сцені.

Tilemap - Компонент, який знаходиться на дочірньому об'єкті Grid. Він зберігає інформацію про те, який тайл розташований у якій клітинці сітки.

Tile Palette (Палітра Тайлів) - вікно у редакторі (Window -> 2D -> Tile Palette), де зберігаються всі доступні тайли. З неї ви берете тайли і малюєте ними на Tilemap.

Анімації в Unity. Animator

Система **Animator** є серцем керування анімаціями в Unity. Вона дозволяє створювати складні переходи між різними станами (наприклад, "Спокій", "Ходьба", "Стрибок").

Ключові компоненти

1. **Animator Component.** Прикріплюється до об'єкта, який має анімуватися. Це точка входу для системи.
2. **Animator Controller.** Це основний "графік" або скінченний автомат. Він візуально представляє всі стани анімації та умови переходу між ними.
3. **Animation Clips.** Власне файли, що містять ключові кадри (рух, зміна спрайтів, зміна розміру тощо). Вони вставляються як "стани" у Controller.
4. **Parameters (Параметри).** Змінні (Float, Int, Bool, Trigger), які ви використовуєте у коді, щоб повідомити Controller, що потрібно змінити стан.

Логіка переходу

У скрипті ви змінюєте параметр в Animator, і Controller автоматично переходить до нового стану, якщо умови переходу виконані.

Приклад керування аніматором

```
using UnityEngine;
public class PlayerAnimator : MonoBehaviour{
    private Animator animator;
    private float horizontalInput;
    void Start(){ animator = GetComponent<Animator>(); }
    void Update() {horizontalInput = Input.GetAxis("Horizontal");
        // 1. Встановлюємо булевий параметр 'IsRunning'
        // Якщо гравець рухається (горизонтальний ввід не 0), IsRunning = True
        bool isRunning = Mathf.Abs(horizontalInput) > 0.01f;
        animator.SetBool("IsRunning", isRunning);
        // 2. Встановлюємо Float параметр 'Speed‘.
         // Якщо потрібно мати плавніший перехід або використовувати Blend Tree
        animator.SetFloat("Speed", Mathf.Abs(horizontalInput));
        // 3. Trigger для одноразової події (наприклад, атака)
        if (Input.GetKeyDown(KeyCode.Space)){
            animator.SetTrigger("Attack");
        }   }}
```

Приклад керування аніматором

У прикладі:

У Animator Controller ви створюєте параметр IsRunning (Bool).

Ви створюєте перехід між станами "Idle" (Спокій) та "Run" (Біг).

Умова переходу з "Idle" в "Run" — це коли IsRunning дорівнює True.

Умова переходу назад — коли IsRunning дорівнює False.

Переваги TileMap

- **Швидкість.** Можна створювати великі рівні, малюючи як пензлем.
- **Оптимізація.** Unity автоматично об'єднує колайдери багатьох тайлів в один великий колайдер (TilemapCollider2D), що значно підвищує продуктивність.
- **Шари.** Можна використовувати кілька Tilemap-шарів (наприклад, "Фон", "Платформи", "Передній план"), щоб створювати глибину та складність сцени.

Приклад застосування TileMap

	Опис
Стиль	Super Mario Bros., Celeste, Terraria.
Шари TileMap	1. Платформи (Foreground/Collision) малюються тайли, по яких рухається гравець (земля, цегла, блоки). На цьому шарі додається TilemapCollider2D. 2. Фон (Background) хмари, задні стіни, декорації, які не мають колайдера. 3. Декорації (Details) дрібні елементи, трава, світильники, які малюються поверх платформ, але також не мають колайдера.
Ключова перевага	Unity автоматично генерує єдиний, оптимізований колайдер для всього шару платформ, що значно підвищує продуктивність порівняно з додаванням BoxCollider2D до кожного окремого блоку.

Приклад застосування TileMap

	Опис
Стиль	Ранні RPG (наприклад, Diablo 1), деякі стратегії.
Особливості	Тайли мають ромбоподібну або діагональну форму. Це вимагає спеціального компонента Isometric Tilemap (або Hexagonal Tilemap для гексагональних сіток).
Ключова перевага	Керування порядком (Sorting Order): Завдяки ізометричному TileMap Unity автоматично сортує об'єкти (гравця, ворогів) так, щоб вони правильно перекривали стіни чи меблі, створюючи ілюзію глибини. Об'єкти, що мають нижче положення по осі Y, малюються вище.