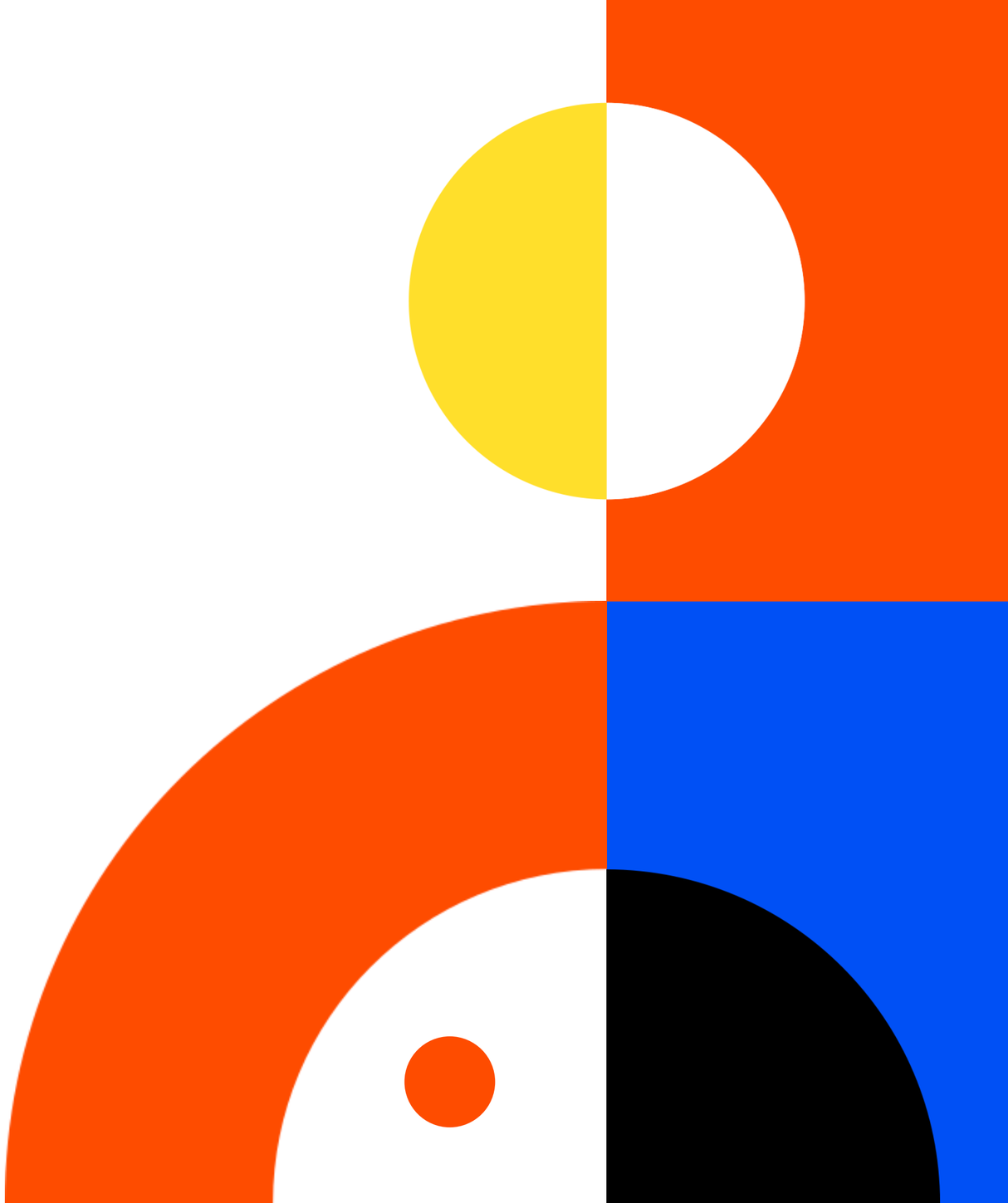
Лекція №2

Скрипти, GameObjects та компоненти

2025



План

1. Скрипти та компонентна система Unity
2. Життєвий цикл скрипта. Магічні функції Інтерфейс Unity.
3. Компонент Transform

Скрипти та компонентна система Unity

Unity побудований не на класах, що успадковують функціонал, а на **об'єктах, що збирають функціонал** — це і є **Компонентна система**.

GameObject

GameObject (Ігровий об'єкт) — це найбазовіша сутність в Unity. Це контейнер, який сам по собі не має жодної поведінки чи форми. Він просто існує у світі (сцені).

Приклад. Порожній GameObject не є ні гравцем, ні деревом, ні камерою. Він стає чимось лише тоді, коли до нього додають компоненти.

Компоненти

Компонент — це "будівельний блок", що додає GameObject певний функціонал, зовнішній вигляд або логіку.

Кожен GameObject **повинен** мати принаймні один компонент: **Transform**.

.

Приклади компонентів

Mesh Renderer додає об'єкту візуальну форму (якщо це 3D).

Sprite Renderer додає 2D-зображення (спрайт).

Rigidbody додає фізичні властивості (гравітація, маса).

Скрипт Ваш власний код, який додає унікальну ігрову логіку.

Scripts

У Unity скрипти пишуться мовою **C#** і завжди успадковуються від базового класу **MonoBehaviour**. Кожен такий скрипт є **Компонентом**, який можна прикріпити до будь-якого GameObject.

- Скрипт не працює сам по собі. Щоб код виконав функцію, його потрібно **прикріпити до GameObject** на сцені.

- **this.gameObject**. Всередині будь-якого скрипта (успадкованого від MonoBehaviour) це посилання завжди вказує на той GameObject, до якого прикріплений цей скрипт.

Життєвий цикл скрипта. Магічні функції

MonoBehaviour має набір спеціальних (магічних) функцій, які викликаються Unity автоматично у певні моменти часу. Розуміння послідовності їх виклику є ключовим для правильного ініціалізації та оновлення логіки.

Функції ініціалізації (Startup)

Функція	Виклик	Призначення
Awake()	Один раз, коли скрипт завантажується.	Ініціалізація. Використовується для встановлення внутрішніх посилань (наприклад, GetComponent) та ініціалізації змінних, незалежно від того, чи є скрипт активним.
Start()	Один раз, перед першим кадром.	Початкова логіка. Використовується для логіки, яка залежить від інших об'єктів (наприклад, розташувати гравця на стартовій точці), оскільки на момент Start() усі Awake() вже виконані.

!!!! Awake() завжди викликається раніше, ніж Start() на сцені.

Функції оновлення (Per-frame Updates)

Функція	Виклик	Призначення
Update()	Кожен кадр. Частота залежить від швидкості комп'ютера.	Ігрова логіка, що залежить від часу: Обробка вводу користувача, таймери, рух, що не залежить від фізики.
FixedUpdate()	Фіксований інтервал часу (за замовчуванням 50 разів на секунду).	Фізична логіка: Переміщення Rigidbody, застосування сили, обробка зіткнень. Завжди використовуйте його, коли працюєте з фізикою, щоб уникнути помилок.

Ці функції викликаються кожен кадр (Frame) і є основою ігрової динаміки.

Приклад

```
using UnityEngine;
public class LifecycleExample : MonoBehaviour{
        public float speed = 5f;       // Змінна ініціалізована

    void Awake() {                              // 1. Викликається першою
        Debug.Log("Awake: Компоненти завантажено.");
        // Тут можна знайти посилання на інші компоненти цього об'єкта:
        // Rigidbody2D rb = GetComponent<Rigidbody2D>();
    }

    void Start()    {                           // 2. Викликається перед першим кадром
        Debug.Log("Start: Починаємо гру.");
        // Тут можна задати початкову позицію
        transform.position = Vector3.zero;
    }

     void Update() {                            // 3. Викликається кожен кадр (для вводу користувача)
        // Рух, що не залежить від фізики
        float horizontalInput = Input.GetAxis("Horizontal");
        transform.Translate(Vector3.right * horizontalInput * speed * Time.deltaTime);
    }

       void FixedUpdate()  {                    // 4. Викликається з фіксованою частотою (для фізики)
        // Логіка для Rigidbody, якщо є
        // rb.AddForce(Vector2.up * 10f);
    }
}
```

Компонент Transform

Transform — це найважливіший і невід'ємний компонент будь-якого GameObject. Він визначає **місце, орієнтацію та розмір** об'єкта.

Ключові властивості

1. **Position**. Позиція об'єкта у світових координатах (наприклад, Vector3(5, 0, 0)).

2. **Rotation.** Орієнтація об'єкта (поворот) у вигляді кватерніона.

3. **localScale.** Розмір об'єкта (масштаб) по осях X, Y, Z.

Робота з Transform у коді

Ви завжди маєте доступ до компонента Transform через його властивість transform (з малої літери).

Метод	Призначення	Приклад
transform.position	Зчитує або встановлює світові координати.	transform.position = new Vector3(0, 5, 0);
transform.Translate()	Переміщує об'єкт відносно його поточної позиції.	transform.Translate(Vector3.forward * speed);
transform.Rotate()	Обертає об'єкт.	transform.Rotate(Vector3.up, 90f);
transform.parent	Посилання на батьківський Transform.	transform.parent = anotherObject.transform;

Ієрархія (Parenting)

Компоненти Transform створюють **Ієрархію** об'єктів. Коли об'єкт A є батьківським (Parent) для об'єкта B (Child):

- Рух, обертання або зміна масштабу батьківського об'єкта автоматично **застосовується до всіх дочірніх об'єктів**.
- transform.localPosition та transform.localRotation показують позицію/обертання **відносно батьківського об'єкта**, а не світового простору.

Ієрархія (Parenting)

Розуміння того, як **MonoBehaviour**, **цикл життя** та **Transform** працюють разом, дасть повний контроль над об'єктами у грі.