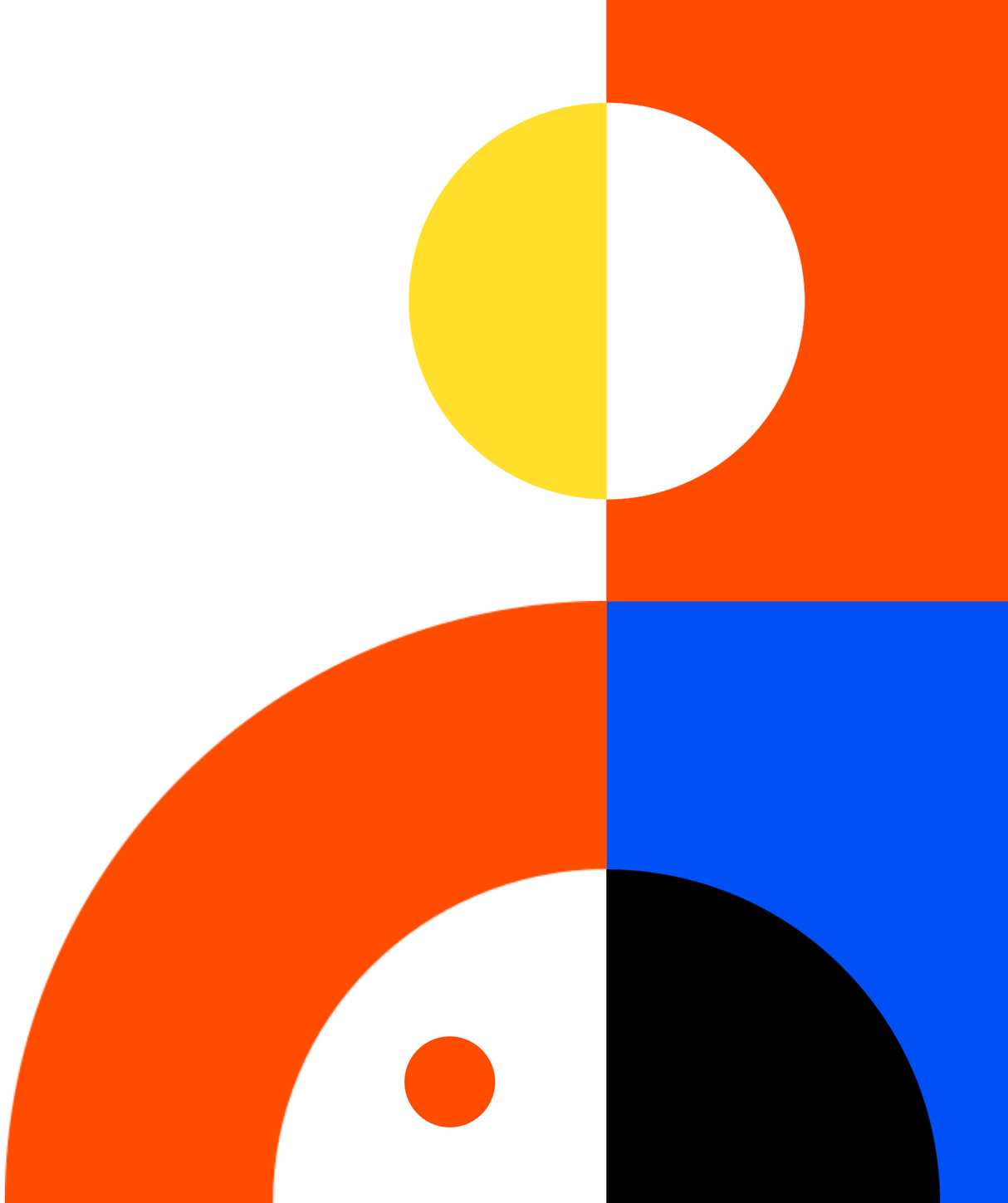
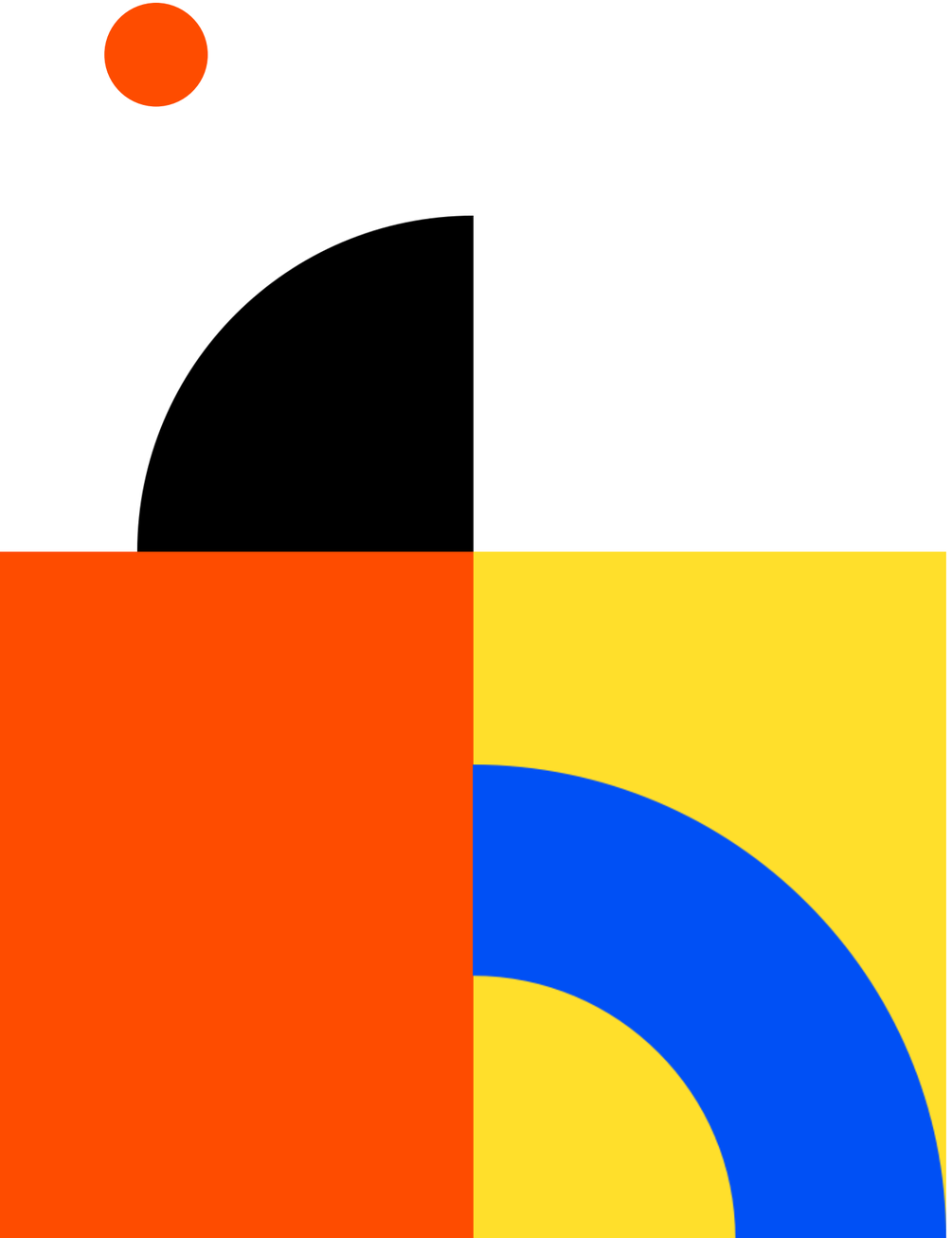
Лекція №9

Resources

2025



План



1. Як зробити білд проєкту
2. Resources
3. StreamingAssets
4. ScriptableObject

Як зробити білд проєкту

Білд (Build) — це процес збирання вашого Unity-проєкту у готову гру або застосунок, який може запускатися **поза редактором Unity**.
Unity компілює:

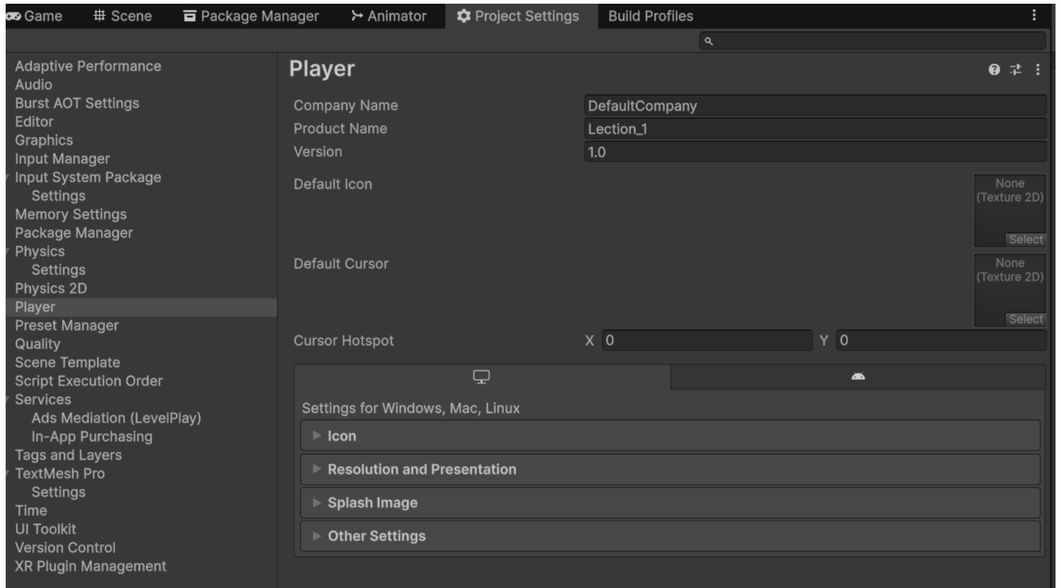
- усі сцени, зазначені в **Build Settings**;
- усі скрипти, ресурси та залежності;
- додає рушій Unity (runtime);
- створює виконуваний файл (exe, apk, xcode project, webgl тощо).

Іншими словами — білд = фінальна версія гри, готова для публікації або тестування.

Відкрити: **Edit → Project Settings → Player**

Найважливіші параметри:

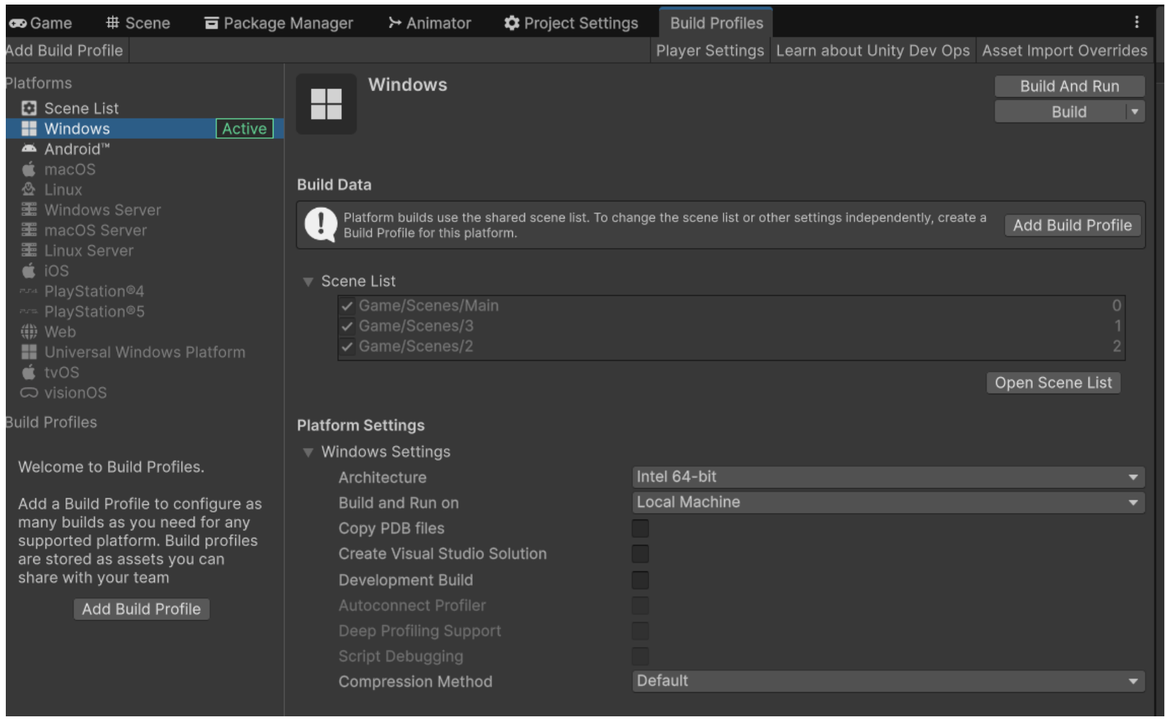
- **Company Name / Product Name** — назви, що відображаються у вікні гри або пакеті APK.
- **Default Icon / Splash Screen** — логотипи та екрани завантаження.
- **Resolution and Presentation** — роздільна здатність і орієнтація екрану.
- **Other Settings → Scripting Backend** — Mono або IL2CPP.
- **API Compatibility Level** — зазвичай .NET Standard 2.1.
- **Target Platform** — Android, Windows, WebGL тощо.



Відкрити: **File → Build Profiles**

Основні елементи:

- **Scenes in Build** — список сцен, які потраплять у білд.
  – Додати можна натиснувши *Add Open Scenes*.
- **Platform** — вибір цільової платформи.
  – Windows / Mac / Linux
  – Android / iOS
  – WebGL
- **Switch Platform** — перехід до вибраної платформи (Unity перекомпілює ассети).
- **Build / Build And Run** — запуск процесу білду.



Ресурси гри без прямих посилань

Unity дозволяє зберігати додаткові ресурси для доступу під час виконання (runtime) навіть після збірки гри.

Для цього існують дві спеціальні директорії:

- **Resources/** — для ресурсів, що вбудовуються в білд.
- **StreamingAssets/** — для зовнішніх файлів, які не обробляються Unity і копіюються у білд без змін.

Resources

Resources – це особлива папка в Unity в яку можна зберігати різні ассети які завжди потраплять в білд (навіть якщо на сцені немає посилання на цей ассет). Данні ассети можна отримати через спеціальний клас Resources:

```
var prefab = Resources.Load<GameObject>("Enemies/Orc");
```

В проєкті може бути декілька папок Resources в різних місцях.

Особливості:

- Завантаження здійснюється під час виконання (Resources.Load, Resources.LoadAsync).
- Підтримуються типи: Texture2D, AudioClip, Prefab, ScriptableObject, тощо.
- Не потребує посилань у сцені — зручно для динамічного контенту.

Недоліки:

- Усе в Resources/ потрапляє у пам’ять білду → збільшує розмір.
- Не можна оновити або замінити ресурс після збірки.

StreamingAssets

Ця папка призначена для **сирих файлів**, які Unity не конвертує. Вона підходить для:

- відео, JSON-конфігурацій, CSV-баз, базових текстур;
- збережень користувача або даних, що завантажуються зовні.

Юніті зберігає каталлоги і файли без змін, але при білді зберігає їх в спеціальну папку, шлях до якої можна отримати через Application.streamingAssetsPath. До данних файлів можна отримати доступ через звичайні методи читання файлів.

Особливості:
•Вміст копіюється “як є” до папки гри (або apk на Android).
•Доступ шляхом Application.streamingAssetsPath.
•Підходить для оновлення без перекомпіляції білду.

Порівняння способів збереження ресурсів

Параметр	Resources	StreamingAssets	Інші ресурси в папці Assets
Формат зберігання	Обробляється Unity	Необроблений	Обробляється Unity
Завантаження	Resources.Load	через File.ReadAllText, WWW, UnityWebRequest	Через пряме посилання або SerializeField
Потрапляє в білд	Так	Так	Лише якщо є пряме посилання в сцені чи коді
Можна оновити без білду	Ні	Так	Ні
Основне застосування	Префаби, текстури, ScriptableObjects	JSON, відео, зовнішні файли	Статичні ресурси, пов’язані напряму зі сценами або префабами

ScriptableObject

У Unity об’єкти типу **ScriptableObject** дозволяють створювати зручні, серіалізовані контейнери даних, які зберігаються як окремі .asset файли. Вони не прив’язані до сцен чи об’єктів і є чудовим способом реалізації **ігрової бази даних**.

- У складному проєкті часто потрібно централізовано зберігати:
- характеристики персонажів (здоров’я, швидкість, ціна, шкода);
- властивості предметів або зброї;
- інформацію про рівні, ворогів, місії тощо.
- Якщо ці дані тримати всередині MonoBehaviour або сцен — це призводить до дублікатів, складного оновлення й високої залежності між об’єктами.

ScriptableObject дозволяє зберігати дані у вигляді **“data-driven” архітектури**.

```
using UnityEngine;

[CreateAssetMenu(fileName = "ItemData", menuName = "Database/Item")]
public class ItemData : ScriptableObject
{
    public string id;
    public string displayName;
    public Sprite icon;
    public int price;
    public int damage;
}
```

Переваги ScriptableObject

- Менше споживання пам’яті — дані не дублюються в кожному префабі.
- Можна редагувати в інспекторі без відкриття сцени.
- Підтримують серіалізацію та інтеграцію з Addressables, Resources.
- Ідеально підходить для **data-driven архітектури** і систем типу ECS.

Поради

- Не зберігайте **стан** (наприклад, поточне HP) — лише **конфігурації**.
- Для runtime-змін створюйте копії в пам’яті (Instantiate ScriptableObject).
- Можна реалізувати пошук по ID або використати Dictionary<string, ItemData> при ініціалізації.