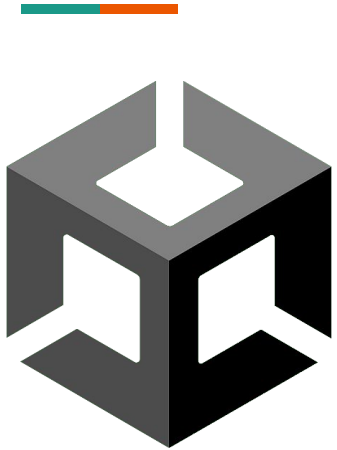
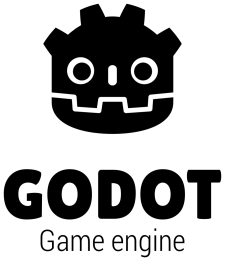
Архитектура Unity. Сравнение с другими движками

Editor, Scenes, GameObjects, Components




UNREAL
ENGINE

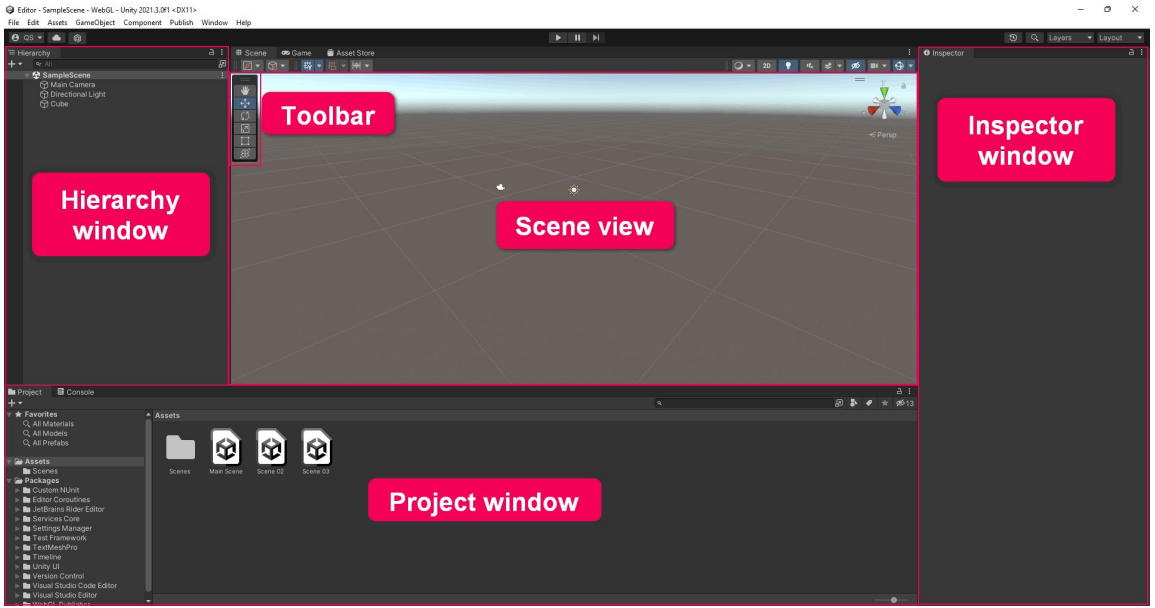

CRYENGINE®


GODOT
Game engine

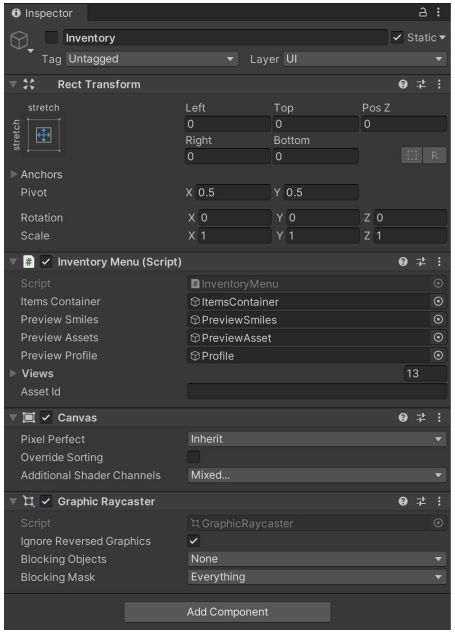

COCOS


GameMaker™

Unity Editor



GameObject



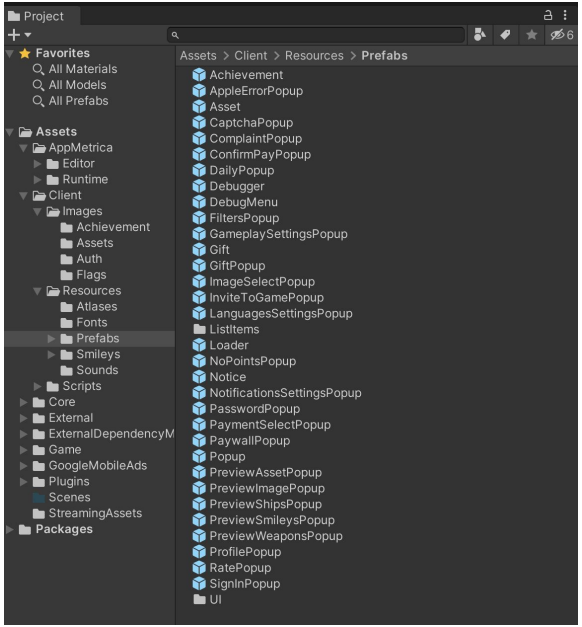
Transform

Components

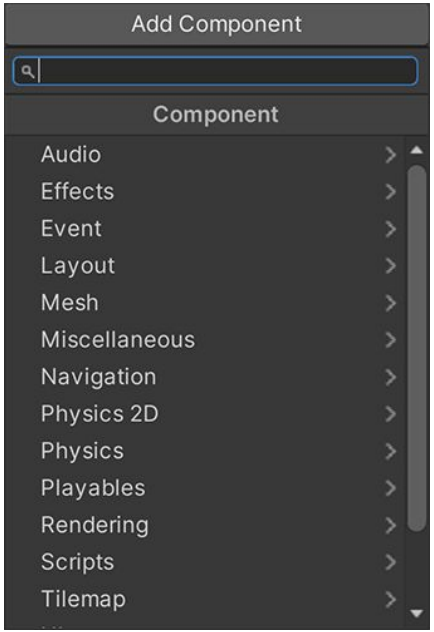
Prefab

GameObject vs Prefab

Объект на сцене
vs
Сохраненный шаблон



Components



Core components

Transform, Rigidbody, Collider, Camera, etc

User components

Any C# scripts

GameObject

├── Transform

├── Renderer

├── Collider

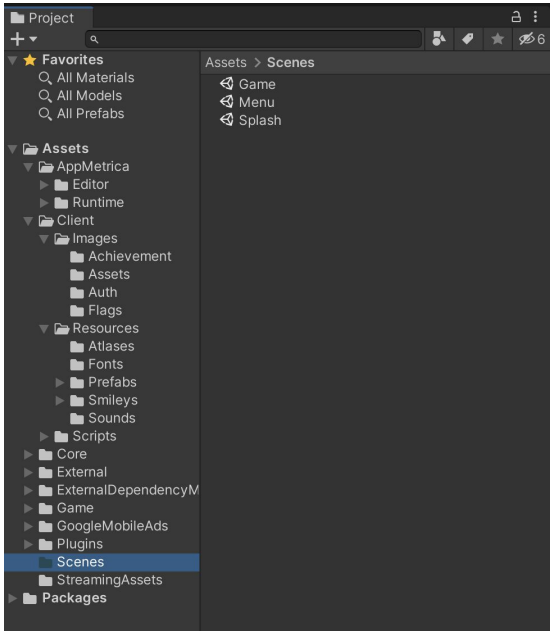
└── Health (скрипт)

Components

```
public class Enemy : MonoBehaviour {
    void OnCollisionEnter(Collision other) {
        Health h = other.gameObject.GetComponent<Health>();
        if (h != null) h.TakeDamage(10);
    }
}
```

```
public class Health : MonoBehaviour {
    public int hp = 100;
    public void TakeDamage(int amount) {
        hp -= amount;
        if (hp <= 0) Destroy(gameObject);
    }
}
```

Scenes

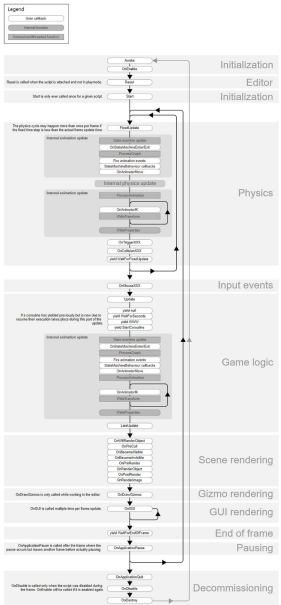


LoadSceneMode: single, additive

DontDestroyOnLoad

Build settings: scenes list

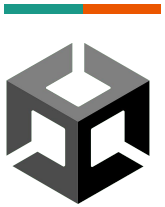
MonoBehaviour Lifecycle



Awake → OnEnable → Start → Update →

→ FixedUpdate → LateUpdate →

→ OnDisable → OnDestroy

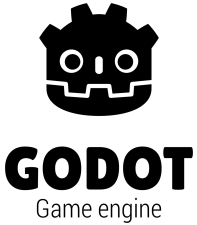


GameObject + Component (C#)

Sprite-based (C++/JS/Lua)

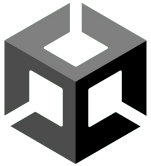
Actor + Component (C++/Blueprints)

GameObject + Component (Lua)



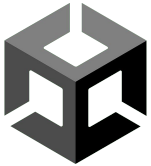
Node-based (GDScript, C#)

Object-oriented (GML)



Простая схема: GO + Components
Низкий порог входа
Кроссплатформенность
C#

Actors + Components
AAA графика из коробки
Сложнее и тяжелее на старте
C++ и Blueprints

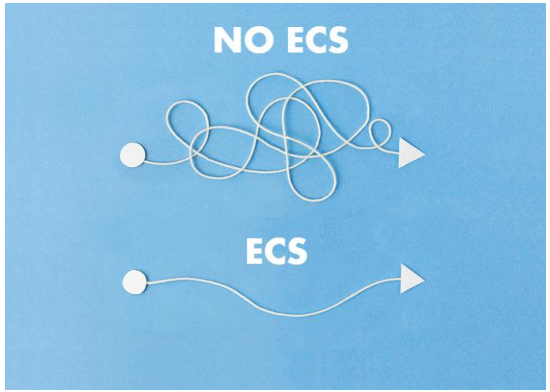


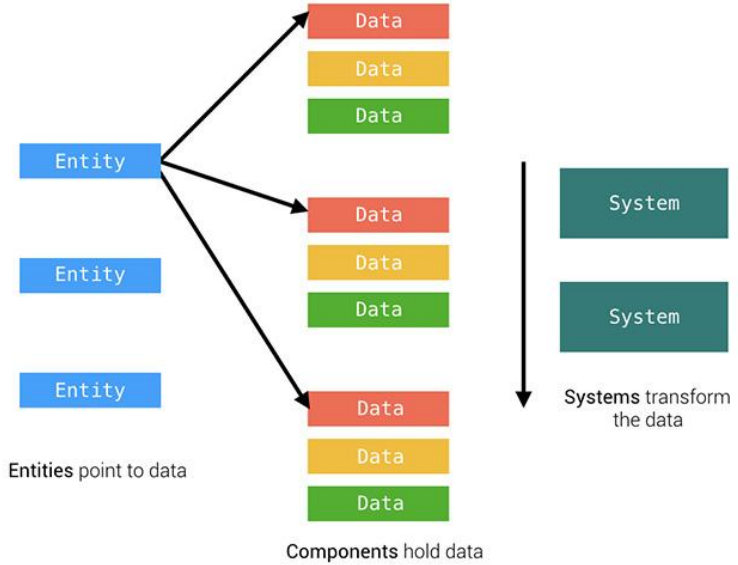
GO + Components
Сцена = объединение объектов
URP, HDRIP
C#
Большое комьюнити

GODOT
Game engine

Node-based
Сцена = дерево узлов
Пока больше для 2D
Open source
GDScript, C++, C#

Бонус. ECS


NO ECS
ECS


Entity
Entity
Entity
Data
Data
Data
Data
Data
Data
Data
Data
Data
System
System
Entities point to data
Components hold data
Systems transform the data

Последний слайд

Unity = GameObject + Components

Scenes = контейнер уровней

У всех движков — компонентная архитектура, но разная реализация

Unity → оптимальный баланс между простотой и гибкостью

Ещё один

2) CSharp в Unity. Жизненный цикл объектов (awake, start, ...), классы, наследование. 3) Сцены и объекты. Префабы, иерархия сцены. Управление ресурсами. 4) Тестирование и дебаг 5) Ввод, Input System vs старый Input. 6) Физика: Rigidbody, Collider, Triggers. 7) Камеры, проекции: перспективная, оргографическая. Cinemachine. 8) Canvas, UI элементы, резиновый интерфейс, события. 9) Звуки и музыка: AudioSource, AudioListener, менеджер звуков. 10) Анимация: Animator и твинеры. 11) Архитектура, паттерны: компонентный подход, singleton, event bus, scriptable object, ecs, fsm, nav mesh, поведенческие деревья 12) Сохранение данных: PlayerPrefs, JSON, SO 13) Сборка под различные платформы: pc, ios, android, webgl. Отличия. 14) Оптимизация. Profiler, батчинг, кеширование, пул, атласы, события 15) Обзор сетевых решений. Клиент-серверная архитектура. 16) Интеграция сервисов и SDK