Справочник атрибутов Unity

Атрибут	Где применяется	Что делает	Пример
`Header("Название")`	поле	Добавляет заголовок в инспекторе	`[Header("Настройки скорости")] public float speed;`
`Tooltip("Текст")`	поле	Подсказка при наведении в инспекторе	`[Tooltip("Скорость движения")] public float speed;`
`Range(min,max)`	float/int поле	Слайдер для чисел	`[Range(0,10)] public int health;`
`Min(value)`	число	Минимальное значение	`[Min(1)] public int lives = 1;`
`Multiline`	строка	Многострочное поле	`[Multiline] public string text;`
`TextArea(min,max)`	строка	Растягиваемое текстовое поле	`[TextArea(3,10)] public string dialogue;`
`Space`	поле	Пустая строка в инспекторе	`[Space] public float damage;`
`HideInInspector`	поле	Скрывает поле	`[HideInInspector] public int hidden;`
`SerializeField`	поле	Сериализует приватное поле	`[SerializeField] private int level;`

`NonSerialized`	поле	Не сериализуется	`[NonSerialized] public int temp;`
`ContextMenu("Название")`	метод	Добавляет метод в контекстное меню	`[ContextMenu("Reset Stats")] void ResetStats() { health = 100; }`
`ContextMenuItem("Кнопка","Метод")`	поле	Кнопка рядом с полем	`[ContextMenuItem("Сбросить"," ResetHealth")] public int health; void ResetHealth() { health=100; }`
`Delayed`	число/строка	Применяет значение только после Enter	`[Delayed] public int score;`
`RequireComponent(typeof(...))`	класс	Автоматически добавляет компонент	`[RequireComponent(typeof(Rigidbody))] public class Player : MonoBehaviour {}`
`DisallowMultipleComponent`	класс	Запрещает несколько копий компонента	`[DisallowMultipleComponent] public class Manager : MonoBehaviour {}`
`AddComponentMenu("Menu/Name")`	класс	Добавляет в меню Component	`[AddComponentMenu("Custom/Enemy")] public class Enemy : MonoBehaviour {}`
`ExecuteInEditMode`	класс	Запускает Update в редакторе	`[ExecuteInEditMode] public class Preview : MonoBehaviour {}`
`ExecuteAlways`	класс	Работает и в Play, и в Edit Mode	`[ExecuteAlways] public class Logger : MonoBehaviour {}`
`DefaultExecutionOrder(100)`	класс	Задаёт порядок вызова Awake/Update	`[DefaultExecutionOrder(100)] public class UIManager : MonoBehaviour {}`

`CreateAssetMenu(menuName="...", fileName="...")`	ScriptableObject	Добавляет пункт в Create Asset	`[CreateAssetMenu(menuName=" Config/Enemy", fileName=" NewEnemyConfig")] public class EnemyConfig : ScriptableObject {}`
`FormerlySerializedAs("oldName")`	поле	Сохраняет данные при переименовании	`[FormerlySerializedAs("hp")] public int health;`
`SerializeReference`	поле	Сериализация абстрактных типов	`[SerializeReference] public IAbility ability;`
`CustomEditor(typeof(...))`	класс-редактор	Кастомный инспектор	`[CustomEditor(typeof(Player))] class PlayerEditor : Editor {}`
`CanEditMultipleObjects`	класс-редактор	Редактировать несколько объектов	`[CanEditMultipleObjects] [CustomEditor(typeof(Player))] class PlayerEditor : Editor {}`
`CustomPropertyDrawer(typeof(...))`	класс-Drawer	Свой рендеринг для типа	`[CustomPropertyDrawer(typeof (MyType))] class MyTypeDrawer : PropertyDrawer {}`
`MenuItem("Tools/Command")`	метод	Добавляет команду в меню	`[MenuItem("Tools/Reset Data")] static void ResetData() { Debug.Log ("Reset!"); }`
`InitializeOnLoad`	класс	Запускает статический конструктор при старте Unity	`[InitializeOnLoad] class AutoInit { static AutoInit(){ Debug.Log("Editor started!"); } }`
`InitializeOnLoadMethod`	метод	Запускает статический метод при старте	`[InitializeOnLoadMethod] static void Init(){ Debug.Log("Start!"); }`
`RuntimeInitializeOnLoadMethod`	метод	Запускается при старте игры	`[RuntimeInitializeOnLoadMethod] static void OnGameStart(){ Debug.Log ("Game Start!"); }`

`ImageEffectAllowedInSceneView`	класс-эффект	Включает эффект в Scene View	`[ImageEffectAllowedInSceneView] class MyEffect : MonoBehaviour {}`
`ImageEffectOpaque`	класс-эффект	Эффект до прозрачности	`[ImageEffectOpaque] class MyEffect : MonoBehaviour {}`
`ImageEffectTransformsToLDR`	класс-эффект	HDR → LDR	`[ImageEffectTransformsToLDR] class MyEffect : MonoBehaviour {}`
`BurstCompile`	класс/метод	Компиляция Burst'ом	`[BurstCompile] struct MyJob : IJob {}`
`ReadOnly`	поле в job	Только чтение	`[ReadOnly] public NativeArray<int> data;`
`WriteOnly`	поле в job	Только запись	`[WriteOnly] public NativeArray<int> results;`
`DeallocateOnJobCompletion`	поле в job	Освободить память после job	`[DeallocateOnJobCompletion] public NativeArray<int> results;`
`NativeDisableParallelForRestriction`	поле в job	Разрешает небезопасный доступ	`[NativeDisableParallelForRestriction] public NativeArray<int> arr;`
`UnityTest`	метод	Coroutine-тест	`[UnityTest] public IEnumerator MyTest(){ yield return null; }`
`Test`	метод	NUnit-тест	`[Test] public void AddTest(){ Assert.AreEqual(2,1+1); }`

`SetUp / TearDown`	метод	Подготовка / очистка перед тестами	`[SetUp] void Init(){}`
`OneTimeSetUp / OneTimeTearDown`	метод	Запускается один раз	`[OneTimeSetUp] void InitAll(){}`
`Ignore("Причина")`	метод	Пропускает тест	`[Ignore("Пока не работает")] [Test] public void TestSkip(){}`