Basic script pack 1.0:

Documentation.

Table of contents:

Table of contents description:

Introduction:

This script pack was made to cover the basic scene event needs of any starting/junior developer that don’t want to get bloated packs, which functionality they won’t use or are scared of its size, understanding deep concepts, reading documentation, etc.

Like: “I’m just making a walking simulator/horror, I need trigger events only!”

Or: “All I want in UGUI is just the ability to drag a window. And a TextMeshPro toggle.”

The scripts inside usually are *less than 100 lines.*

Most of the scripts usage in Inspector is self-explanatory.
Not all have a detailed how-to in this documentation.
There are many Tooltips in the inspector window on hover.
Contains some code comments.

1. Editor tools (1)

NameModifier.cs

How to use:

1. Find Tools, which in upper left quarter in Unity
2. Click it
3. Click Name Modifier
4. Choose multiple objects with Ctrl + LMB/Shift etc.
5. Drag down the border if “Apply” button is not visible
6. See the preview results in same window
7. Apply. Undo: Ctrl + Z. Redo: Ctrl + Y.

Namespace: Basic.Editor

Variables:

```
private string _prefix = "";
private string _suffix = "";
private string _removeText = "";
private bool _previewMode = true;
```

Methods:

```
private static void ShowWindow()
private void OnGUI()
private string GetModifiedName(string originalName)
```

Local variables:

```
string newName = GetModifiedName(obj.name);
string newName = originalName;
```

2. Game sound (1)

CollisionSoundHandler.cs

Info:

This is arcade realisation for the most diverse sounding, modify code/edit variables in inspector for a more realistic feel.

Known issues:

- **Pitch randomizes too high for some cases**
- **on default settings.**

Variables	Names	Line
private AudioClip[]	_impactSounds	10
private float	_baseVolume	11
private float	_basePitch = 1f;	13
private float	_pitchRandomization	14
private float	_minVelocity = 0.1f;	17
private float	_maxVelocity = 10f;	18
private float	_velocityToPitchMultiplier = 0.1f;	19
private float	_velocityToVolumeMultiplier = 0.1f;	20
private float	_minTimeBetweenSounds = 0.1f;	23
void	Awake()	28

void	OnCollisionEnter(Collision collision)	35
Local variables	Names	Line
float	collisionForce	39
float	velocityPitchModifier	43
float	randomPitchModifier	44
float	finalPitch	45
float	velocityVolumeModifier	48
float	finalVolume	49
AudioClip	randomSound	52

The most important part:

```
// Calculate pitch
float velocityPitchModifier = Mathf.Clamp(collisionForce *
_velocityToPitchMultiplier, 0.1f, 2f);
float randomPitchModifier =
Random.Range(-_pitchRandomization,
_pitchRandomization);
float finalPitch = _basePitch + velocityPitchModifier +
randomPitchModifier;

// Calculate volume
float velocityVolumeModifier = Mathf.Clamp(collisionForce
/ _maxVelocity, 0.1f, 1f);
float finalVolume = _baseVolume + (velocityVolumeModifier
* _velocityToVolumeMultiplier);
```

3. Timers (6)

KinematicFalseTimer.cs

Namespace: Basic

Variables:

```
[SerializeField] private Rigidbody[] _rigidbodies;
[SerializeField] private float _timer = 3f;
[SerializeField] private bool _deactivateAfterUse = false;
private float _interval;
private bool _isActivated;
```

Methods:

```
private void Update()
```

SceneTimer.cs

Info:

Very simple and uses the classic Update() timer. It is supposed to run in an enabled state and can disable itself (Probably you don’t need it, as it will be destroyed in the previous scene).

Namespace: Basic.Timers

Variables:

```
private bool _disableScriptAfter;
[SerializeField] private float _timer = 13f;
private float _interval;
```

Class:

```
[SerializeField] private string _sceneName = "1";
```

SelfDestroy.cs

The most simple one, just a Destroy(gameObject, _timer) call in Start() with 0f by default in Inspector.

SetActiveTimer.cs

Info:

Another Update() logic timer with the ability to disable itself or its gameObject.

Classes:

[SerializeField] private GameObject[] _targetObjects;

[SerializeField] AudioSource _audioSource;

Class array:

[SerializeField] AudioSource[] _additionalSources;

Variables:

[SerializeField] private float _timer = 3f;

[SerializeField] private bool _deactivateGameObjectAfterUse;

[SerializeField] private bool _disableScriptAfterUse;

[SerializeField] private bool _isUsingAudioTime;

float _requiredSingleClipLength;

private float _interval;

Methods:

private void Update()

void CheckDisablesAfterUse()

Local variables:

float totalExtraTime;

UEventTimer_Coroutine.cs

Info:

Coroutine version of UnityEvent timer, do not mass use to not allocate memory. Better readability for easier modifying by others, use System.Threading.Tasks one instead below. Highly unrecommended in <= Unity 2019.

Variables:

[SerializeField] private float _delay = 1f;

[SerializeField] private bool _triggerOnStart;

[SerializeField] private bool _repeating;

[SerializeField] private UnityEvent _onTimerComplete;

Methods:

private void Start()

public void StartTimer()

public void StopTimer()

private IEnumerator SingleTimer()

private IEnumerator RepeatingTimer()

UEventTimer.cs

Info:

Does exactly the same result as the upper version, but is using async Task to imitate timer in sync Start().
Is a 100 lines of code mess, main difference is that methods have Async suffix to their name and they are async Task.

4. Visual (2)

LightFlicker.cs

Namespace: Basic.Lighting

Variables:

```
[Header("Single light still must be assigned")]
[SerializeField] private Light _light;
[SerializeField] private Light[] _lights;
[SerializeField] private float _baseIntensity = 1f;
[SerializeField] private float _flickerIntensity = 0.3f;
[SerializeField] private float _flickerSpeed = 0.1f;
[SerializeField] private bool _smoothTransition;
 private float _randomOffset;
 private float _timer;
```

Methods:

```
private void Awake()
private void Update()
private void RandomFlicker()
private void SmoothFlicker()
```

SecurityCameraDisplay.cs

Namespace: Basic.Renders

How to use:

Info:

- Form Factor for 1x1 scaled quad is 240x240 etc.
- Multiply quad scale X*16 and Y*9 for FHD, etc.
- For quick reset to FHD and keep other:
- Reset - Copy vector - Undo - Paste back

Steps:

1. Create material, do not use the same for all.
2. Make a camera, without Audio Listener.
3. Add script to gO (GameObject)
4. Assign first two steps
5. Customize resolution and color depth (1 still gives color?)
6. Put the material on a quad
7. Press play.

Variables:

```
[SerializeField] private Camera _securityCamera;
[SerializeField] private Material _displayMaterial;
[SerializeField] private Vector3Int _resolutionDepth = new
Vector3Int(1920, 1080, 24);
 [Tooltip("Resets color to white opaque")]
 [SerializeField] private bool
_isResetColorForProperVisibility;
```

5. Triggers (2)

TriggerLoadScene.cs

Info:

- Compares “Player” tag by default in Trigger
- Is async.
- Have a bool to Thread.Sleep() for old school loads feel
- it stops *EVERYTHING* completely for a period of time.
- If not, the _freezeTime is used for Time.scale & Task.Delay().

Variables:

```
[SerializeField] private bool _useOldSchoolThreadSleep;
[SerializeField] private bool _doNotUseAnyTag;
[SerializeField] private string _sceneNameToLoad;

[Range(0, 10)][SerializeField] private float _stopTimer = 1f;
[SerializeField] private CanvasGroup _loadingCanvas
```

Methods:

```
private async void OnTriggerEnter(Collider other)
public async Task LoadSceneSequenceAsync()
```

Local variables:

```
var loadOperation =
SceneManager.LoadSceneAsync(_sceneToLoad);
int ms = Mathf.RoundToInt(_freezeTime * 1000);
```

UEventOnTrigger.cs

Info:

The most useful script, but slower if you need the same actions many times, like set kinematic false.
Only have access to public methods and encapsulations, etc. The easiest way to disable gO, do AudioSource.Play()..
The first word is abbreviated for easier search (“Ue..”).

Namespace: Basic.Triggers

Variables:

[SerializeField] private bool _DisableScriptAfterEnter;
[SerializeField] private bool _DisableGameObjectAfterEnter;

Classes:

public UnityEvent _OnTriggerEnter;
public UnityEvent _OnTriggerExit;
public UnityEvent _OnTriggerStay;

Methods:

private void OnTriggerEnter(Collider other)
private void OnTriggerExit(Collider other)
private void OnTriggerStay(Collider other)
private void CheckDisables()

6. UI (5)

DraggablePanel.cs

Info:

Simple drag a panel script,

also can restrict going out of bounds.

Classes:

public class DraggablePanel : MonoBehaviour,
IBeginDragHandler, IDragHandler

[SerializeField] private RectTransform
_panelRectTransform;
[SerializeField] private Canvas _canvas;

Variables:

[SerializeField] private float _edgePadding = 10f;
private Vector2 _dragStartPosition;
private Vector2 _minPosition;
private Vector2 _maxPosition;

Methods:

private void Awake()
private void CalculateBounds()
public void OnBeginDrag(PointerEventData eventData)
public void OnDrag(PointerEventData eventData)

RawImageDarkenOnPointer_Legacy.cs

Info:

If raw image is null it gets the nearest raw image in object it is in or its child with:

_rawImage = GetComponentInChildren<RawImage>();

(Gets component in parent☝️ if null. Assign to prevent.)

Classes:

public class RawImageDarkenOnPointer_Legacy : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler

[SerializeField] private RawImage _rawImage;

Variables:

[SerializeField] private float _valueReduction = 0.3f;

private Color _originalColor;

Methods:

private void Awake()

public void OnPointerEnter(PointerEventData eventData)

public void OnPointerExit(PointerEventData eventData)

TextDarkenOnPointer_Legacy.cs

Classes:

public class TextDarkenOnPointer_Legacy :
MonoBehaviour, IPointerEnterHandler, IPointerExitHandler

[SerializeField] private Text _text;

Variables:

[SerializeField] private float _valueReduction = 0.3f;

[SerializeField] private float _valueReduction = 0.3f;

private Color _originalColor;

Methods:

private void Awake()

public void OnPointerEnter(PointerEventData eventData)

public void OnPointerExit(PointerEventData eventData)

TextDarkenOnPointer_TMP.cs

Info:

Exactly the same, except using TextMeshPro.

FPSCounter_TMP.cs

[SerializeField] private TextMeshProUGUI _fpsText;

[SerializeField] private float _updateInterval = 0.5f;

private float _accum;

private int _frames;

private float _timeLeft;

private float _fps;

private void Update()

7. Starts (4)

StartThreadSleep.cs

Info:

Use only to achieve glitchy 2000’s game feel.
But be cautious. Do not put in Awake() or an enabled == false bool of script will not work (If this is not what you want).

Variable:

public int msToSleep = 1000;

Method:

private void Start()

Explosion.cs

Info:

Put this on your explosion prefab, just a classic addforce boom with a layerMask.

Variables:

```
    [SerializeField] private float _radius = 5f;
    [SerializeField] private float _force = 700f;
    [SerializeField] private float _upwardsModifier = 3f;
    [SerializeField] private LayerMask _affectedLayers;
```

Methods:

Start()

LockCursor.cs

Info:

It just locks your cursor and hides the mouse pointer.

Method:

```
void Start()
{
    Cursor.lockState = CursorLockMode.Locked;
}
```

TargetFramerate.cs

Info:

This is the most important thing, it prevents you from flying away with your coolers and paying a huge electricity bill. Especially when the scene is empty.

A must have in a build version, that gives a large amount of frames.

This version works even if the Nvidia panel is set to Fast mode Sync (The QualitySettings.vSyncCount = 0 part is important in this case).

```
private void Start()
{
QualitySettings.vSyncCount = 0;
Application.targetFrameRate =
(int)Screen.currentResolution.refreshRateRatio.value;
}
```

8. Extra (1)

PublicFloat.cs

Info:

(Potentially useful (like you can modify all the timers to use a timer float from this instead). This one requires your code in other scripts to work, but you can set it with UnityEvent thanks to setter)

Namespace: Basic.PublicVariables

Other:

```
public class PublicFloat : MonoBehaviour
[SerializeField] private float _float;
public float FloatGetSet { get => _float; set => _float =
value; }

public void PrintFloat()
```

FAQ:

- Q: What is that number next to the main headers?
- A: It is the total number of scripts, easier to track whether it was added to docs or not.

- Q: I don’t understand your documentation and code.
- A: Scripts are mostly near 50 lines of code, just finish https://learn.unity.com/pathway/junior-programmer
- (No, this actually will never make you job ready.)
- and you probably will understand most of it (except the async UEventTimer). Or practice with what you need with AI Chats, they are very helpful for starters, you can even put existing code there to improve, but better do not do it with a confidential one.
- The more detailed you give the prompt the better, use permanent context prompts like “Use linearVelocity for Rb’s” to fix problems with latest Unity versions or its hallucinations.

- Q: What is a method?
- A: In this case everything that is not a variable, class, you can call it a procedure, this one is shorter.
- Q: Why is your text so huge and white?
- A: It is easier to look in a browser without pressing F11, or a phone. And at night.