[ 유니티 중요 함수 ]

Awake

- 스크립트가 실행될 때 한 번만 호출죄는 함수다.

- 주로 게임의 상태 값 또는 변수의 초기화에 사용한다.

- Start 함수가 호출되기 전에 맨 먼저 호출된다.

- 스크립트가 비활성화돼 있어도 실행된다.

- 코루틴으로 실행 불가

Start

- Update 함수가 호출되기 전에 한 번만 호출된다.

- 스크립트가 활성화돼 있어야 실행된다.

- 다른 스크립트의 모든 Awake가 모두 다 실행된 이후에 실행된다.

- 코루틴으로 실행 가능

Update

- 프레임마다 호출되는 함수로 주로 게임의 핵심 로직을 작성한다.

- 스크립트가 활성화돼 있어야 실행된다.

LateUpdate

- 모든 Update 함수가 호출되고 나서 한 번씩 호출된다.

- Update 함수에서 전처리가 끝난 후 실행해야 하는 로직에 사용한다.

- 카메라 이동 로직에 주로 사용하는 함수다.

- 스크립트가 활성화돼 있어야 실행된다.

FixedUpdate

- 물리엔진의 시뮬레이션 계산주기로 기본값은 0.02초다.

- 발생하는 주기가 일정하다.

OnEnable

- 게임오브젝트 또는 스크립트가 활성화됐을 때 호출된다.

- 이벤트 연결 시 사용한다.

- 코루틴 사용 불가

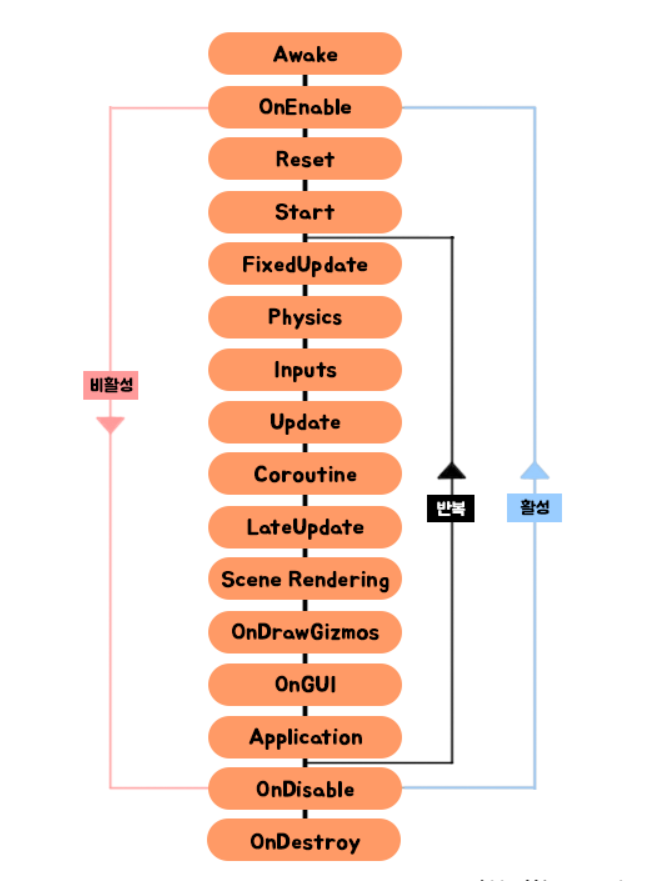
OnDisable

- 게임오브젝트 또는 스크립트가 비활성화됐을 때 호출된다.

- 이벤트 연결을 종료할 때 사용한다.

- 코루틴 사용 불가

OnGUI : 레거시 GUI 관련 함수를 사용할 때 사용한다.



**Get**은 키값에 넣어져있는 데이터를 불러온다.(그값을 스스로 가집니다.)

예) keydata =PlayerPrefs..GetInt("KEY",default); -default는 초기생성시에만 필요.

**Set**은 키값에 데이터를 넣어줍니다.

예) PlayerPrefs.SetInt("KEY",data);

**Save**는 이게임의 PlayerPrefs의 값들을 저장합니다.

예) PlayerPrefs.Save();

**DeleteKey**는 지정된 키값과 데이터를 지웁니다.

예) PlayerPrefs.DeleteKey("KEY");

**DeleteAll**은 이게임의 PlayerPrefs의 모든 키값과 데이터를 지웁니다.

예) PlayerPrefs.DeleteAll();

**HasKey**는 키값에 값이 있는지 체크합니다.

예) keybool = PlayerPrefs.HasKey("KEY"); -값이 있다면 이것은 true 없다면 false의 값을 가집니다.

**버튼(Button)**

버튼은 클릭했을 때 수행할 동작을 정의하기 위한 **OnClick** UnityEvent를 가지고 있습니다.

## 토글(Toggle)

토글은 현재 토글이 켜짐/꺼짐 중 어느 상태인지를 결정하는 **Is On** 체크박스를 가지고 있습니다. 이 값은 사용자가 해당 토글을 클릭할 때마다 변하며 그에 따라 체크 마크가 표시되거나 사라집니다. 또한 이 값이 변했을 때 수행할 동작을 정의하기 위한 **OnValueCHanged** UnityEvent를 가지고 있습니다.

## 슬라이더(Slider)

슬라이더는 사용자가 최소값에서 최대값까지 드래그할 수 있는 10진수 숫자 **값**을 가집니다. 슬라이더는 수평 또는 수직 중 하나의 형태입니다. 또한 이 값이 변했을 때 수행할 동작을 정의하기 위한 **OnValueChanged** UnityEvent를 가지고 있습니다.

## 스크롤바(Scrollbar)

스크롤바는 01 사이의 소수점 **값**을 가집니다. 사용자가 스크롤바를 드래그하면 그에 따라 이 값이 변합니다.

스크롤 뷰를 만들기 위해 종종 [스크롤 사각 영역](https://docs.unity3d.com/kr/2018.4/Manual/script-ScrollRect.html) 및 [마스크](https://docs.unity3d.com/kr/2018.4/Manual/script-Mask.html)를 스크롤바와 함께 사용합니다. 스크롤바는 전체 스크롤바 길이의 비율로써 핸들의 크기를 결정하는 01 사이의 **크기** 값을 가집니다. 이는 종종 스크롤 뷰에 있는 콘텐츠가 어느 비율만큼 보여야 하는지를 나타내기 위해 다른 컴포넌트에 의해 조절됩니다. 이를 Scroll Rect 컴포넌트가 자동으로 할 수 있습니다.

스크롤바는 수평 또는 수직 중 하나의 형태입니다. 또한 값이 변했을 때 수행할 동작을 정의하기 위한 **OnValueChanged** UnityEvent를 가지고 있습니다.

## 드롭다운(Dropdown)

드롭다운은 선택할 수 있는 옵션 리스트를 가지고 있습니다. 텍스트 문자열과 선택적인 이미지를 각 옵션에 지정할 수 있으며, 인스펙터 또는 코드에서 동적으로 설정할 수 있습니다. 현재 선택되어 있는 옵션이 변경되었을 때 수행할 동작을 정의하기 위한 **OnValueChanged** UnityEvent를 가지고 있습니다.

## 입력 필드(Input Field)

입력 필드는 사용자가 [텍스트 요소](https://docs.unity3d.com/kr/2018.4/Manual/script-Text.html)의 텍스트를 수정할 수 있도록 하기 위해 사용합니다. 텍스트 콘텐츠가 변경되었을 때 수행할 동작을 정의하기 위한 UnityEvent, 그리고 사용자가 편집을 마쳤을 때 수행할 동작을 정의하기 위한 또 다른 UnityEvent를 가지고 있습니다.

## 스크롤 사각 영역(Scroll Rect)(스크롤 뷰)

공간을 많이 차지하는 콘텐츠를 작은 영역에 표시해야 할 때 스크롤 사각 영역을 활용할 수 있습니다. 스크롤 사각 영역(Scroll Rect)은 콘텐츠를 스크롤하는 기능을 제공합니다.

일반적으로 스크롤 사각 영역은 [마스크](https://docs.unity3d.com/kr/2018.4/Manual/script-Mask.html)와 함께 사용하여 스크롤 사각 영역 중 유일하게 스크롤 가능한 콘텐츠가 보이는 스크롤 뷰를 생성합니다. 여기에 한두 개의 [스크롤바](https://docs.unity3d.com/kr/2018.4/Manual/script-Scrollbar.html)를 추가하여 수직 혹은 수평으로 드래그하게 할 수도 있습니다.

## 1. 코루틴 (Coroutine)

코루틴은 시간의 경과에 따른 절차적 단계를 수행하는 로직을 구현하는 데 사용되는 함수이다. 시간의 경과에 따른 절차적 단계에 대한 로직을 구현하는 것은 Update() 함수들에서도 가능하다. 매 프레임마다 호출되는 로직은 Update()에서 구현하면 된다. 하지만 초당 호출이나, 매 프레임마다 호출이 필요하지 않은 부분을 매 프레임마다 호출하는 것은 바람직한 로직이 아니다.

코루틴은 함수를 호출한다. 함수는 한 프레임에 호출되어 완료가 된다. 이에 IEnumerator 형식을 반환 값으로 가지는 함수를 사용한다. IEnumerator는 함수 내부에 실행을 중지하고, 다음 프레임에서 실행을 재개할 수 있는 yield return 구문을 사용한다.

## 2. yield return

yield return 구문 에는 다음과 같은 YieldInstruction 클래스를 사용한다.

yield return null : 다음 프레임에 실행을 재개한다.

yield return new WaitForSeconds : 지정된 시간 후에 재개한다.

yield return new WaitForSecondsRealtime :  Time.timescale 값에 영향을 받지 않고 지정된 시간 후에 재개한다.

yield return new WaitForFixedUpdate : 모든 스크립트에서 모든 FixedUpdate가 호출된 후에 재개한다.

yield return new WaitForEndOfFrame : 모든 카메라와 GUI가 렌더링을 완료하고, 스크린에 프레임을 표시하기 전에 호출된다.

yield return StartCoroutine() : 코루틴을 연결하고 코루틴이 완료된 후에 재개한다.

## 3. StartCoroutine()와 StopCoroutine()

Coroutine은 3개의 전달 인자를 사용한다.

3.1. StartCoroutine()

string MethodName을 사용하는 경우, 두 번째 매개변수로, 전달 인자를 사용한다.

Coroutine StartCoroutine(string MethodName, object = null)

Coroutine StartCoroutine(IEnumerator routine)

Coroutine StartCoroutine(Coroutine routine)

3.2. StopCoroutine()

StopCoroutine(string MethodName)

StopCoroutine(IEnumerator routine)

StopCoroutine(Coroutine routine)

유니티는 코루틴을 사용할 때, 전달 인자를 혼합하여 사용하지 말 것을 권장한다.

만약 StartCoroutine(string MethodName)을 사용했다면, StopCoroutine(string MethodName)을 사용하여 중지한다. 실제 사용해 보면, 대상 코루틴을 찾지 못한다.