직렬화된 사전

직렬화된 사전은 Unity 에디터에서 자연스럽게 사용할 수 있도록 설계되었으며, 자주 사용하는 워크플로의 속도를 높일 수 있는 몇 가지 추가 기능을 제공합니다.

빠른 시작

데이터를 직렬화하려면 **Dictionary<,>** 클래스 대신 *AYellowpaper.SerializedCollections* 네임스페이스에서 **SerializedDictionary<,>** 클래스를 사용하세요. 추가 커스터마이징을 위해 **SerializedDictionary** 어트리뷰트를 사용합니다. 다른 유니티 유형과 동일한 유니티 직렬화 규칙을 따릅니다.

사용 예는 아래 이미지를 참조하세요:

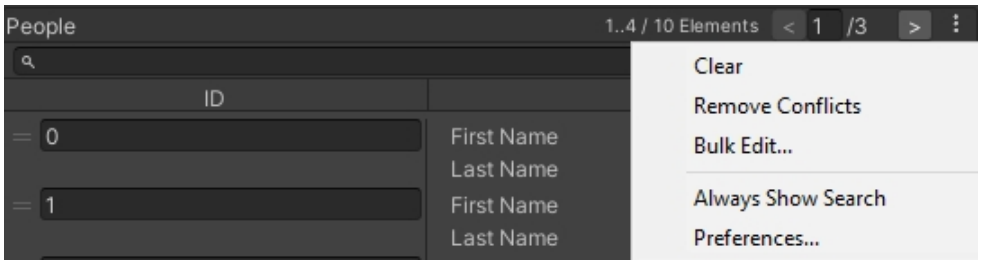
```
[SerializedDictionary("Damage Type", "Description")]
public SerializedDictionary<DamageType, string> ElementDescriptions;
```

사용자 가이드

Serialized Dictionary는 트랜스폼 및 ScriptableObject와 같은 Unity 오브젝트를 포함하여 모든 Unity 직렬화 가능한 유형을 직렬화합니다. 또한 중복 키와 널 값도 직렬화할 수 있습니다. 주요 목적은 코드를 변경하거나 오브젝트를 제거할 때 실수로 데이터가 손실되는 것을 방지하는 것입니다. 다음과 같은 색상 코딩이 있습니다:

- 빨간색: 키가 유효하지 않습니다. 즉, 중복되거나 널입니다.
- 노란색: 중복된 키가 있지만 이 키가 사용되는 키입니다(다른 키보다 먼저 나옴).
- 파란색: 검색에서 키를 찾았습니다.

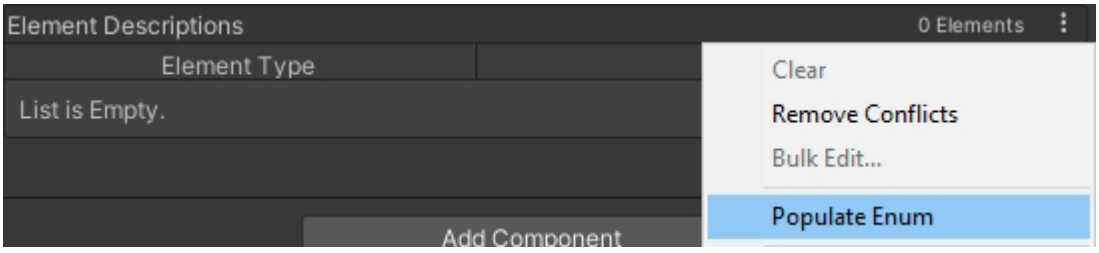
오른쪽 상단의 버거 메뉴는 매우 중요합니다. 여기에는 워크플로 속도를 높 여 주 는 중요한 옵션이 포함되어 있습니다. 대부분은 설명이 필요 없을 것입니다.


People
1..4 / 10 Elements
1
/3
ID
0
First Name
Last Name
1
First Name
Last Name
Clear
Remove Conflicts
Bulk Edit...
Always Show Search
Preferences...

대량 편집 작업

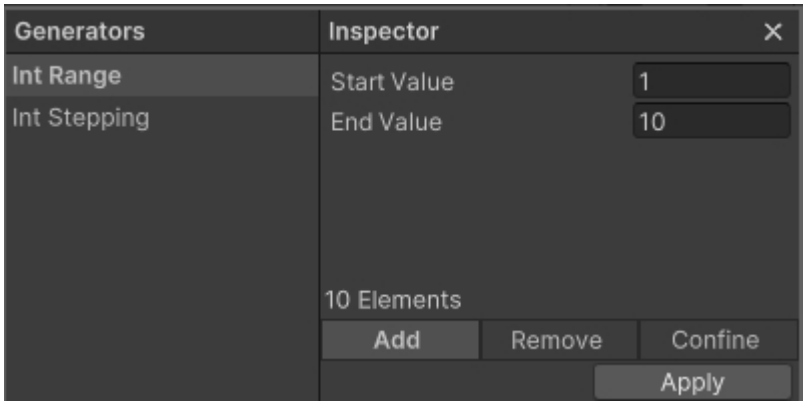
많은 기존 항목을 빠르게 수정하려면 사용자 지정 **KeyListGenerators**를 사용하거나 생성할 수도 있습니다.
예를 들어 열거형을 키로 포함하는 딕셔너리의 경우 **키 목록** 생성기를 통해
버튼을 한 번만 누르면 열거형에 있는 모든 값이 포함된 사전을 만들 수 있습니다.

1. 열거형 키가 있는 사전으로 '열거형 채우기'를 선택합니다.



2. 사전은 열거형의 모든 값으로 채워집니다.

또한 정수를 위한 포퓰레이터가 있어 사용자 지정 입력 필드를 통해 생성될 데이터를 수정할 수 있습니다.



이 경우 Int Range는 1에서 10 사이의 범위의 키를 생성합니다. 생성된 값을 적용하기 전에 추가, 제거 및 제한 중에서 선택할 수 있는 옵션이 있습니다. 각 옵션은 다음과 같은 작업을 수행합니다:

- *추가는* 값이 아직 키로 존재하지 않는 경우 값을 추가합니다.
- 제거는 지정된 값을 제거합니다.
- *Cofine은* 값이 아직 키로 존재하지 않는 경우 값을 추가하고, 생성된 값 목록에 포함되지 않은 모

든 키를 제거합니다.

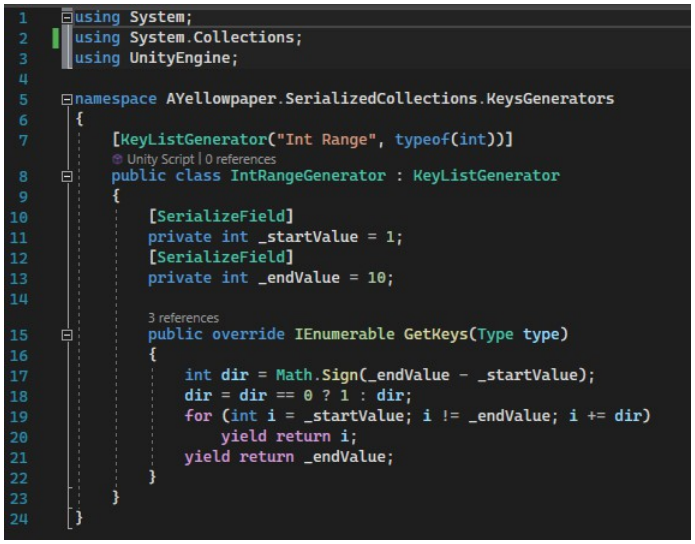
예를 들어, 사전에서 5번부터 15번까지의 키가 있고 생성기에서 1번부터 10번까지의 키를 선택했다고 가정해 보겠습니다. 다음 옵션이 주어지면 결과 키는 다음과 같습니다:

- 추가하면 1~4가 추가되므로 1에서 15까지의 키가 생성됩니다.

- 제거하면 5~9번 키가 제거되므로 11~15번 키가 생성됩니다.
- 1~4가 추가되고 11~15가 제거되므로 Confine은 1~10이 됩니다.

대량 편집 작업 만들기

열거형과 정수형에 대한 KeyListGenerator가 일부 존재합니다. 하지만 자신만의 커스텀 키 생성기를 추가하고 싶을 수도 있습니다. **KeyListGenerator를 상속하는** 새 클래스를 만들고 여기에 **KeyListGenerator** 어트리뷰트를 추가하면 쉽게 추가할 수 있습니다. 정수 생성기 예제는 이미지를 참조하세요:



```
using System;
using System.Collections;
using UnityEngine;

namespace AYellowpaper.SerializedCollections.KeysGenerators
{
    [KeyListGenerator("Int Range", typeof(int))]
    Unity Script | 0 references
    public class IntRangeGenerator : KeyListGenerator
    {
        [SerializeField]
        private int _startValue = 1;
        [SerializeField]
        private int _endValue = 10;

        3 references
        public override IEnumerable GetKeys(Type type)
        {
            int dir = Math.Sign(_endValue - _startValue);
            dir = dir == 0 ? 1 : dir;
            for (int i = _startValue; i != _endValue; i += dir)
                yield return i;
            yield return _endValue;
        }
    }
}
```