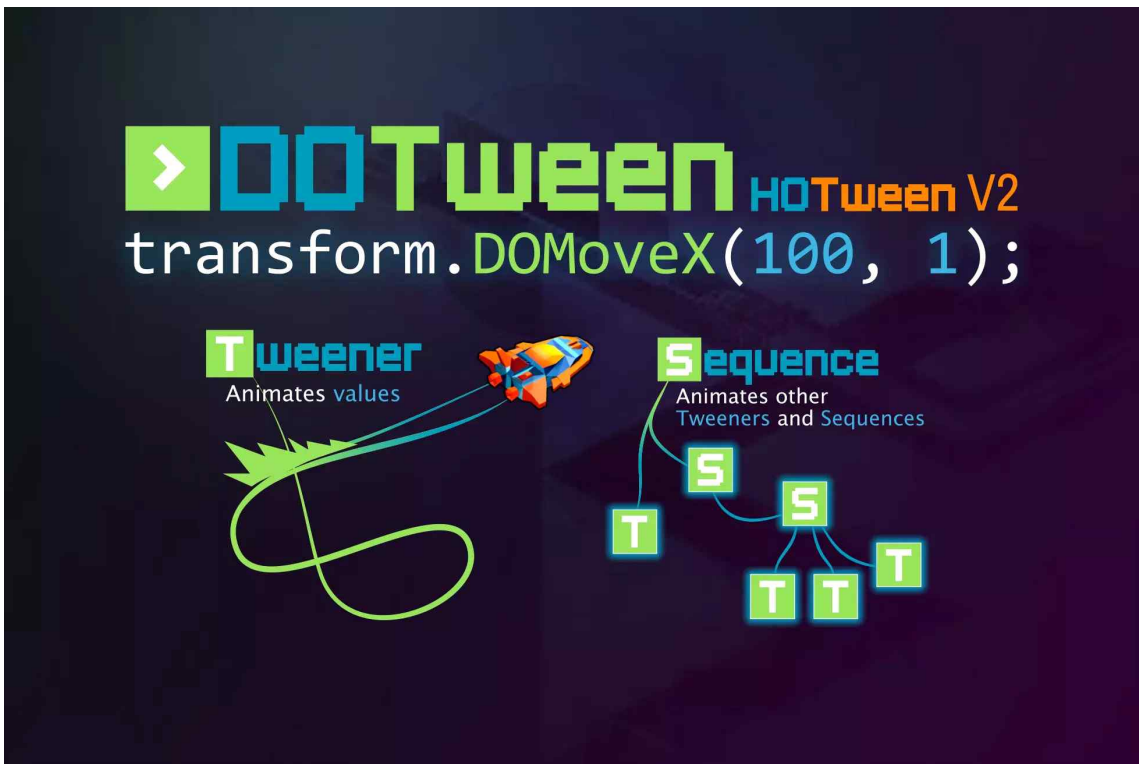
〈DOTween〉

1. 개요

DOTween은 Unity에서 매우 많이 쓰이는 트위닝 라이브러리로, 오브젝트의 위치, 회전, 크기, 색상, UI 요소 등의 애니메이션을 간단하고 효율적으로 제어할 수 있게 해줍니다. 코드가 간단하고, 다양한 옵션/이징(Ease) 및 콜백 기능이 있어서 게임 UI, 오브젝트 애니메이션, 효과 연출 등에 적합합니다.



2. 준비사항 & 패키지 설치

2.1 DOTween 패키지 가져오기

– Unity Asset Store 또는 Demigiant 공식 사이트에서 DOTween 다운로드/구입{맨아래 링크 있음}

– 또는 Unity 패키지 매니저(UAS: Unity Asset Store)에서 설치

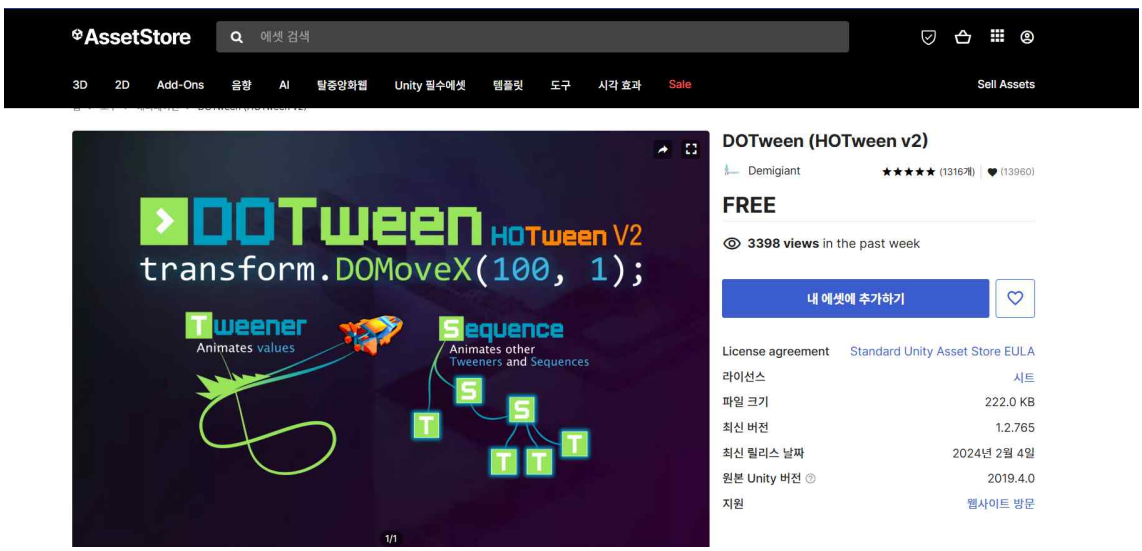
2.2 DOTween 설정 초기화

– 설치 후 Unity 상단 메뉴에서 'Tools'〉 'Demigiant'〉 'DOTween Utility Panel' 열기

'Setup DOTween…' 실행 (필요한 모듈 체크, 에디터 통합 여부, 에디터 타임라인 등 옵션 설정) ([링크][1])

2.3 필요한 네임스페이스 사용 선언

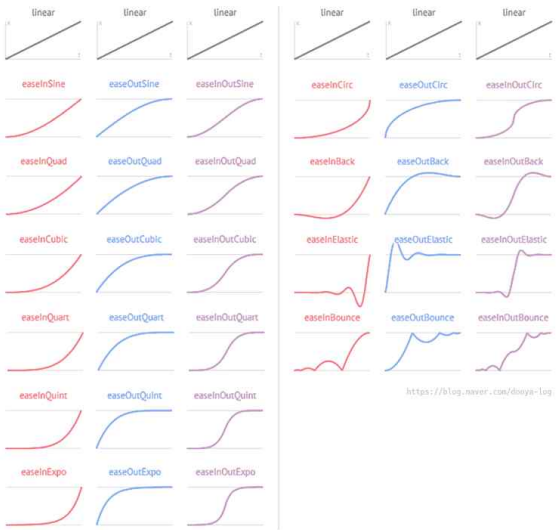
```
using DG.Tweening;
```



3. 기본 사용법 (Shortcuts + Tweener)

다음은 주로 쓰이는 DOTween의 기본 기능들입니다. 코드 + 키 입력 또는 트리거 이벤트 중심 실습 가능하게 구성한 표입니다.

기능	코드 예제	설명
이동 (Move)	csharp transform.DOMove(new Vector3(0,0,10), 1f);	월드 좌표에서 지정 위치로 부드럽게 이동
축별 이동	csharp transform.DOMoveX(5f, 1f); transform.DOMoveY(3f, 1f); transform.DOMoveZ(-2f, 1f);	특정 축(X,Y,Z)만 이동
로컬 이동	csharp transform.DOLocalMove(new Vector3(2f,1f,0), 1f);	부모 기준 등의 로컬 공간에서 이동
점프 (Jump)	csharp transform.DOJump(targetPos, jumpPower:2f, numJumps:1, duration:1.5f);	포물선 형태 점프 + 착지
회전 (Rotate)	csharp transform.DORotate(new Vector3(0,180,0), 1f, RotateMode.FastBeyond360);	회전 모드 옵션 포함
스케일 (Scale)	csharp transform.DOScale(new Vector3(2f,2f,2f), 1f);	크기 변경
펀치 스케일 (Punch Scale)	csharp transform.DOPunchScale(new Vector3(1.2f,1.2f,1.2f), duration:0.5f, vibrato:10, elasticity:1f);	흔들리는 효과 포함 크기 변화



4. 고급 옵션 & 애니메이션 조합

기본 기능 외에, 효과를 더 풍부하게 만들기 위해 유용한 기능들입니다.

- Ease 함수: '.SetEase(Ease.OutQuad)'등으로 애니메이션의 속도 곡선 조정
- 루프 (Loops):
  '.SetLoops(횟수, LoopType.Yoyo 또는 LoopType.Restart)'
- 시퀀스 (Sequence): 여러 Tween을 순차 혹은 병렬로 조합

```
Sequence seq = DOTween.Sequence();
seq.Append(transform.DOMoveX(3f,1f));
seq.Append(transform.DORotate(new Vector3(0,90,0),1f));
seq.Join(transform.DOScale(new Vector3(2,2,2),1f));
```

- 콜백: 트위닝 완료 시 또는 특정 시점에서 함수 실행

```
transform.DOMove(new Vector3(5,0,0),1f)
        .OnComplete(() => Debug.Log("이동 완료"))
        .OnStart(() => Debug.Log("이동 시작"));
```

- DOTween.To (Generic Tween): 단순 값 + 커스텀 Getter/Setter로 사용 가능

```
 float myValue = 0f;
DOTween.To(() => myValue, x => myValue = x, 10f, 2f)
      .OnUpdate(() => Debug.Log($"값 변화: {myValue}"));
```

5. 예제 코드 (MonoBehaviour 스크립트)

실습용 예제로 붙일 수 있는 코드입니다. 키 입력으로 여러 애니메이션 실행해보는 형태로 효과 비교 가능하게 했습니다.

```
using UnityEngine;
using DG.Tweening;

public class DOTweenDemo : MonoBehaviour
{
    public Vector3 destination = new Vector3(3f, 1f, 0f);
    public float duration = 1f;
    public float jumpPower = 2f;
    public int numJumps = 1;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            // 이동
            transform.DOMove(destination, duration);
        }
        if (Input.GetKeyDown(KeyCode.X))
        {
            transform.DOMoveX(destination.x, duration);
        }
        if (Input.GetKeyDown(KeyCode.Y))
        {
            transform.DOMoveY(destination.y, duration);
        }
        if (Input.GetKeyDown(KeyCode.Z))
        {
            transform.DOMoveZ(destination.z, duration);
        }
        if (Input.GetKeyDown(KeyCode.J))
        {
            transform.DOJump(destination, jumpPower, numJumps, duration);
        }
        if (Input.GetKeyDown(KeyCode.R))
        {
            transform.DORotate(new Vector3(0f, 180f, 0f), duration,
RotateMode.FastBeyond360);
        }
        if (Input.GetKeyDown(KeyCode.S))
        {
            transform.DOScale(new Vector3(2f,2f,2f), duration);
        }
    }
}
```

6. UI / 색상 / 투명도 애니메이션

DOTween은 UI 요소나 색상 변화, 알파 투명도 변화도 쉽게 할 수 있어요.

```
using UnityEngine;
using UnityEngine.UI;
using DG.Tweening;

public class DOTweenUIDemo : MonoBehaviour
{
    public Image uiImage;
    public Color targetColor = Color.red;
    public float colorDuration = 1f;
    public float fadeTo = 0.5f;
    public float fadeDuration = 1f;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.C))
        {
            uiImage.DOColor(targetColor, colorDuration);
        }
        if (Input.GetKeyDown(KeyCode.F))
        {
            uiImage.DOFade(fadeTo, fadeDuration);
        }
    }
}
```

7. 적용 시 고려사항

- 과도한 Tween이 씬에 많으면 성능 영향 있을 수 있음- 적절한 종료(완료) 및 메모리 관리 필요
- ase, Loop, Delay 등의 조합이 많아지면 애니메이션 간 타이밍 관리 중요
- I 애니메이션과 게임 오브젝트 애니메이션 간 우선순위 및 타이밍 조정 신경 써야 함
- OTween 버전 / Unity 버전 호환 체크
- OTween이 활성화된 플랫폼(iOS, Android, WebGL 등)에서의 동작 테스트 필요

[다운]

에셋스토어 링크

[공식 사이트]

공식 페이지

[문서작성 참고자료]

1. 블로그
2. GPT