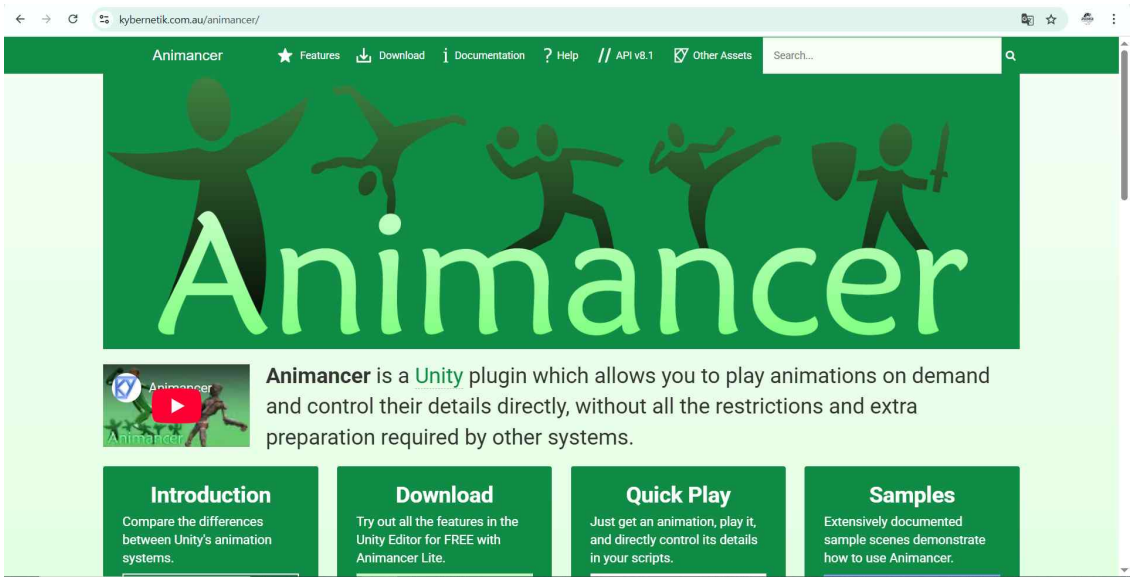
〈Animancer〉

1. 한 줄 요약 / 핵심 장점

Animancer는 Unity의 Mecanim Animator Controller보다 더 직관적이고 유연하게 애니메이션을 제어할 수 있도록 해 주는 플러그인입니다. 스크립트 중심 애니메이션 제어, 즉시 재생 및 페이드 전환, 계층/레이어/믹서 지원 등이 강점이며, Animator Controller의 복잡성/불필요한 제약을 피할 수 있습니다. ([링크][1])



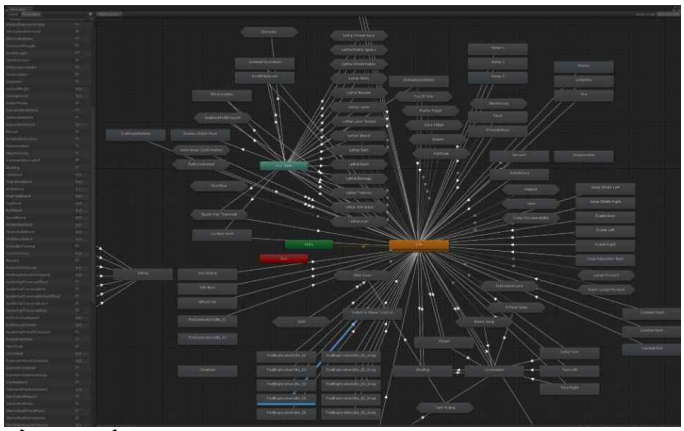
2. 적용 대상

– 애니메이션 상태 전환이 많고 복잡한 캐릭터 (예: 액션게임, RPG, 플랫포머)

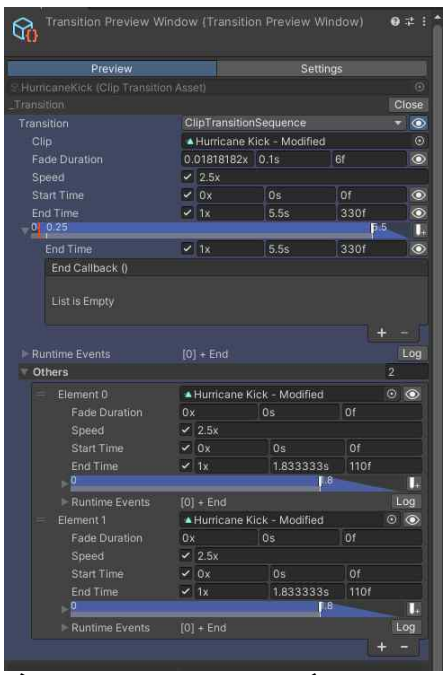
– 코드 기반으로 애니메이션을 제어하고 싶은 프로젝트

– 애니메이터 트리/전환 설정이 너무 무거워서 작업 속도가 느린 경우

– 애니메이션 이벤트, 레이어, 믹서, 방향별 애니메이션 등 세밀한 컨트롤 필요할 때



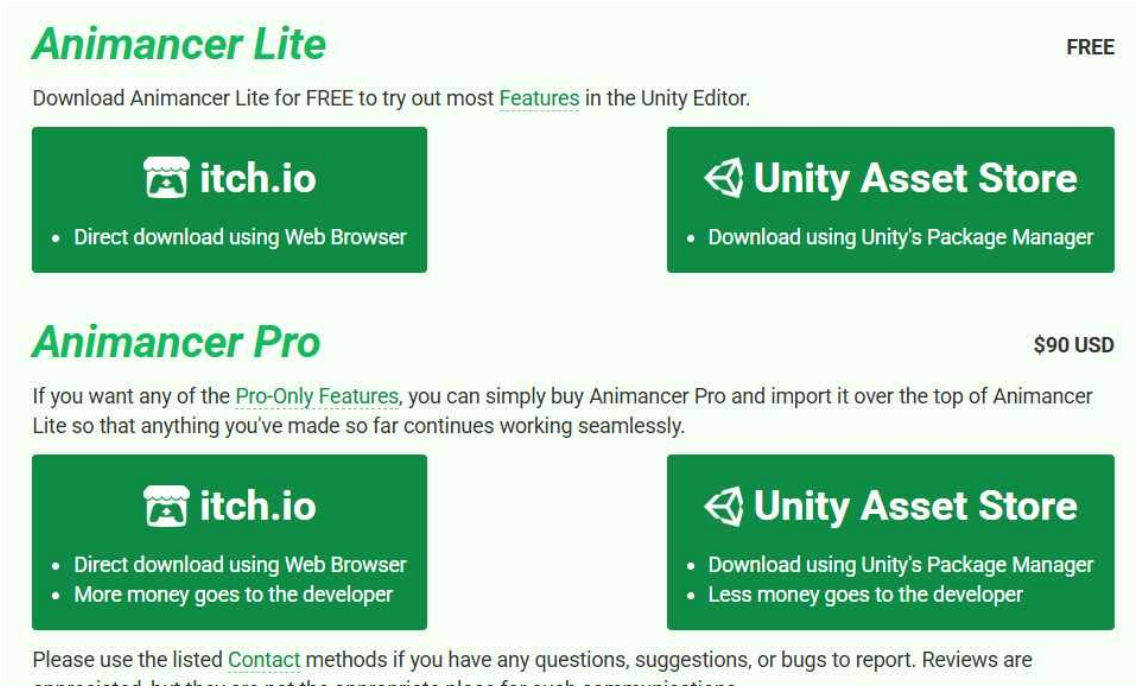
〈기존〉



〈Animancer〉

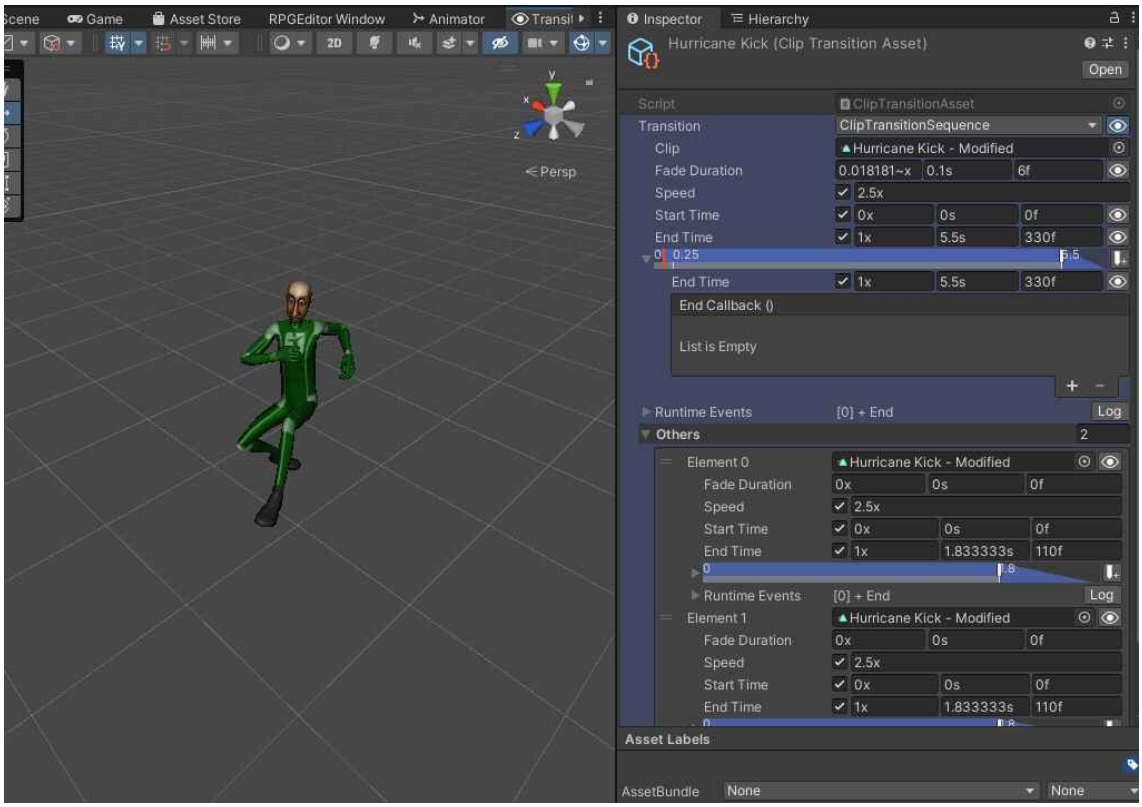
3. 버전 / 라이선스 개요

- Animancer Lite: 무료 버전으로, 일부 기능 제한 있음 비록 Editor 내에서는 Pro 기능 미리 써볼 수 있음. ([링크][1])
- Animancer Pro: 상업 프로젝트에서 전체 기능 및 런타임 사용 가능. 소스 코드 포함, 고급 기능, 보다 많은 툴 & 유틸리티 제공됨. ([itch.io][2])



4. 주요 기능 & 특징 정리

기능	설명
즉시 재생(Play on demand)	AnimancerComponent를 사용해 AnimationClip을 코드에서 바로 재생 가능. Animator Controller 사전 설정 불필요.[링크]
부드러운 전환/페이드(Fader/Blend)	애니메이션 간 전환을 시간 또는 커스터마이즈된 곡선(curve)으로 부드럽게 가능.[링크]
레이어 & 믹서(Layers/Mixers)	서로 다른 애니메이션을 여러 레이어에서 함께 재생/오버라이드 가능. 예: 상체/하체 별 애니메이션 분리 처리 등.[링크]
FSM(Finite State Machine) 통합 또는 별도 사용	Animancer.FSM 기능이 있어서 애니메이션 상태 전환 로직과 애니메이션 시스템을 분리할 수 있음.[링크]
이벤트 관리	애니메이션 중 특정 타이밍에 코드 콜백 등록 가능 (Inspector 또는 코드에서).[링크]
Live Inspector & 디버깅	실행 시 인스펙터에서 현재 애니메이션 상태, 블렌딩 정보 등을 실시간으로 볼 수 있음.[링크]
지원 애니메이션 타입	Humanoid / Generic / Sprite 기반 애니메이션 / Root Motion / IK 등 다양한 유형 지원.[링크]



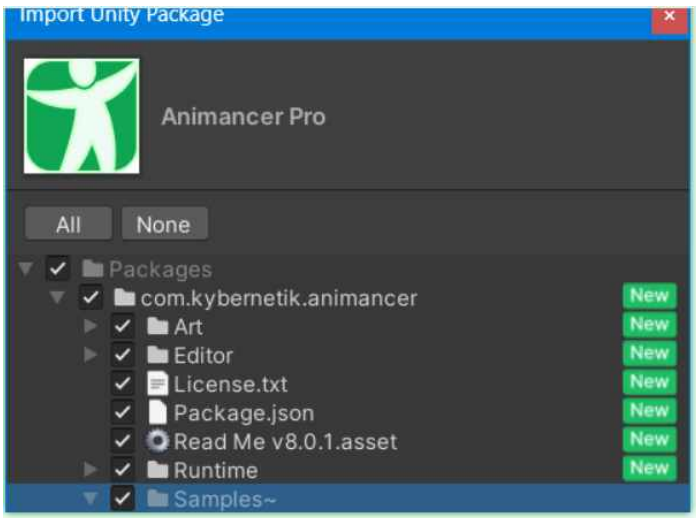
5. 기본 사용법 & 예제 코드

5.1 설치 및 기본 설정

5.1.1 Unity Asset Store 〉 Animancer Pro 구매 & 프로젝트에 Import. (또는 Lite 버전으로 시험) ([링크][5])

5.1.2 씬(Scene)에 애니메이션을 사용할 캐릭터 GameObject에 'AnimancerComponent' 추가.

5.1.3 사용할 'AnimationClip' 준비 (Humanoid / Generic / Sprite 애니메이션).



You can then import the samples via the Package Manager:

5.2 간단 예제: 이동 & 대기 애니메이션 제어

```
using UnityEngine;
using Animancer;

public class AnimancerMovement : MonoBehaviour
{
    public AnimancerComponent animancer;
    public AnimationClip idleClip;
    public AnimationClip walkClip;

    void Update()
    {
        if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.D))
        {
            // 걷는 애니메이션 재생 (반복)
            animancer.Play(walkClip);
        }
        else
        {
            // 대기 애니메이션
            animancer.Play(idleClip);
        }
    }
}
```

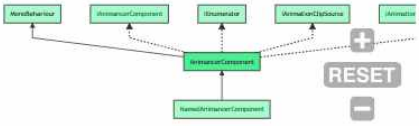
5.3 전환(Blending) 사용 예제

```
// 부드러운 페이드 전환
animancer.Play(walkClip, fadeDuration: 0.2f);
```

Summary

> The main component through which other scripts can interact with `Animancer`. It allows you to play animations on an `UnityEngine.Animator` without using a `UnityEngine.RuntimeAnimatorController`.

Assembly	Animancer.dll
Namespace	Animancer
Interfaces	IAnimancerComponent IEnumerator IAnimationClipSource IAnimationClip Collection
Base Types	MonoBehaviour
Derived Types	NamedAnimancer Component

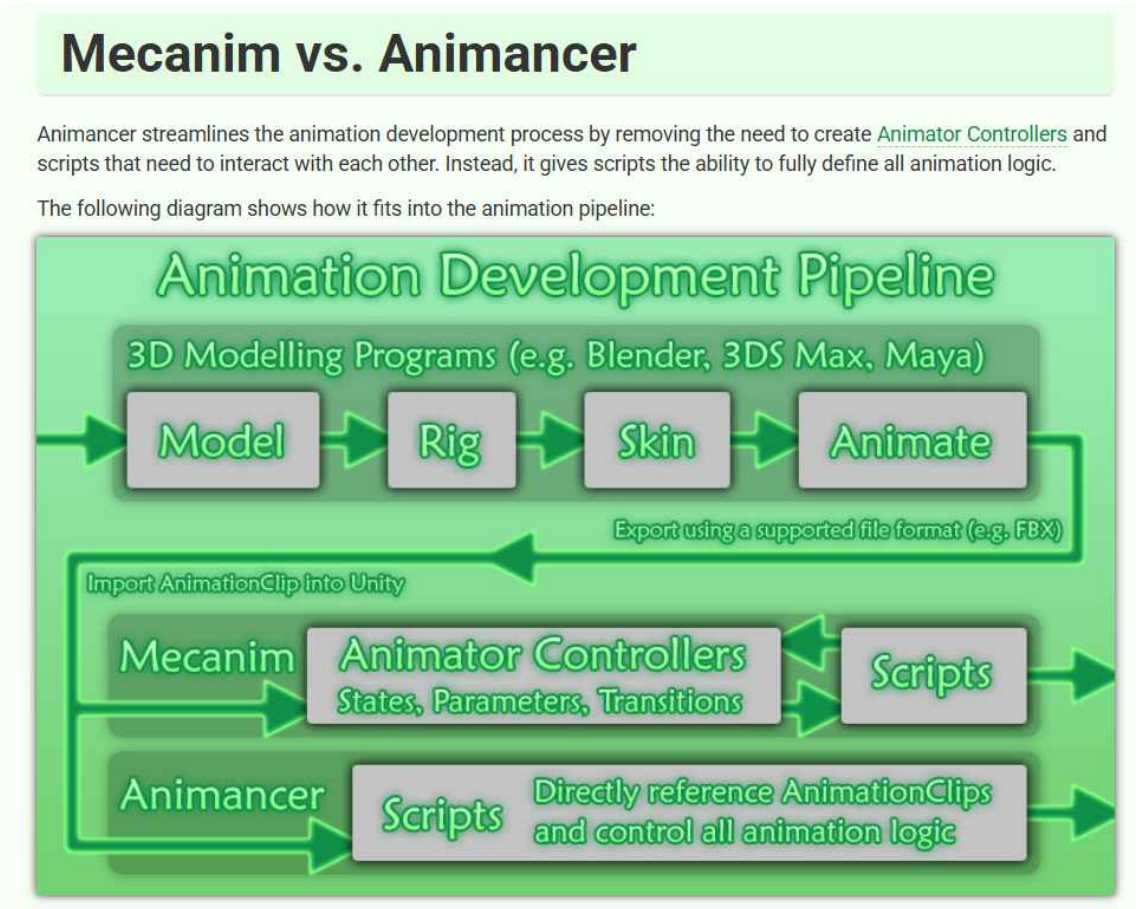


5.4 이벤트 등록 예제

```
using Animancer;

public class AnimancerWithEvents : MonoBehaviour
{
    public AnimancerComponent animancer;
    public AnimationClip attackClip;

    void PerformAttack()
    {
        var state = animancer.Play(attackClip);
        if (state.Events(this, out AnimancerEvent.Sequence events))
        {
            events.Add(0.3f, OnAttackHit); // 애니메이션의 30% 지점에서 공격 히트 판정
            events.OnEnd = OnAttackEnd;
        }
    }

    void OnAttackHit()
    {
        Debug.Log("Attack Hit!");
    }

    void OnAttackEnd()
    {
        Debug.Log("Attack Animation Finished");
    }
}
```

6. 비교: Mecanim Animator Controller vs Animancer

항목	Mecanim(Animator Controller)	Animancer
상태/전환 정의 방식	비주얼 에디터에서 상태/전이(transition)를 미리 정의	코드 중심, 필요할 때 애니메이션 재생 + 전환 가능
애니메이션 리소스 관리	복잡한 Controller, 큰 그래프, 많은 파라미터 필요	단순히 AnimationClip 참조하거나 배열/SO 형태로 관리 가능
런타임 변경성	Controller 수정이 어렵고 동적 전환 제약 많음	런타임에 애니메이션 클립 변경, 이벤트 추가 등 가능
디버깅/가시성	상태 머신 시각화는 좋지만 전환 조건 추적이 복잡	Live Inspector 등을 통한 실행중 애니메이션 상태 추적 용이
복합 애니메이션(레이어,믹서,블렌딩)	가능하나 Controller 설정 복잡	레이어/블렌더/믹서 기능 내장, 코드/Inspector에서 비교적 간단하게 설정 가능



7. 실무 팁 & Best Practices

- AnimationClip 리소스 정리: 자주 쓰는 애니메이션 클립들을 ScriptableObject나 배열로 묶어 관리하면 코드가 깔끔해짐.
- 이벤트 타이밍 조정: 애니메이션 이벤트를 Inspector에서 조정하고, 코드에서는 callback만 연결하는 "하이브리드 방식" 권장. ([링크][4])
- Blending 시간과 그래프 성능 튜닝: 과도한 블렌딩이나 많은 동시 재생 상태는 퍼포먼스 이슈 발생 가능 〉 필요한 것만 켜기.
- Animator Controller와 혼용 가능: 기존 프로젝트에서 Animator Controller 많이 쓰고 있다면, Animancer를 부분 도입해서 덜 복잡한 동작부터 옮겨보는 단계적 마이그레이션 가능. ([링크][1])
- 디버깅 시 Live Inspector 활용: 애니메이션이 바뀌는 시점, 블렌드 가중치(weight) 등을 실시간으로 확인하며 문제 파악 〉 애니메이션 끊김/이상 동작 감소.

8. 확장성 & 주의할 점

- 스크립트 기반 상태 머신 (Animancer.FSM) 과 연결하면 애니메이션 로직 + 게임 로직 분리 가능
- Humanoid Rig / IK / Root Motion 같이 복잡한 리깅/움직임은 실험 필요 (특히 IK & Root Motion이 많을 경우 예상치 못한 동작 가능성 있음)
- 대규모 캐릭터 + 많은 애니메이션 클립을 메모리/리소스 측면에서 미리 프로파일링 해볼 것
- 패키지 버전 업데이트 시 기존 Animator Controller 설정과 호환성 또는 마이그레이션 지침 확인

9. 요약 & 추천 사용 시나리오

Animancer는 Animator Controller의 제약 때문에 애니메이션 전환이 복잡하거나 빈번한 프로젝트, 게임 플레이 중 동적으로 애니메이션을 바꿔야 하는 캐릭터가 많은 경우에 특히 유용해요. 예를 들어, 액션 게임에서 캐릭터가 점프 중에 공격 – 회피 – 착지 동작이 자주 바뀌는 경우, Animancer의 즉시 재생 + 이벤트 + 믹서/레이어 기능이 큰 도움이 됩니다.