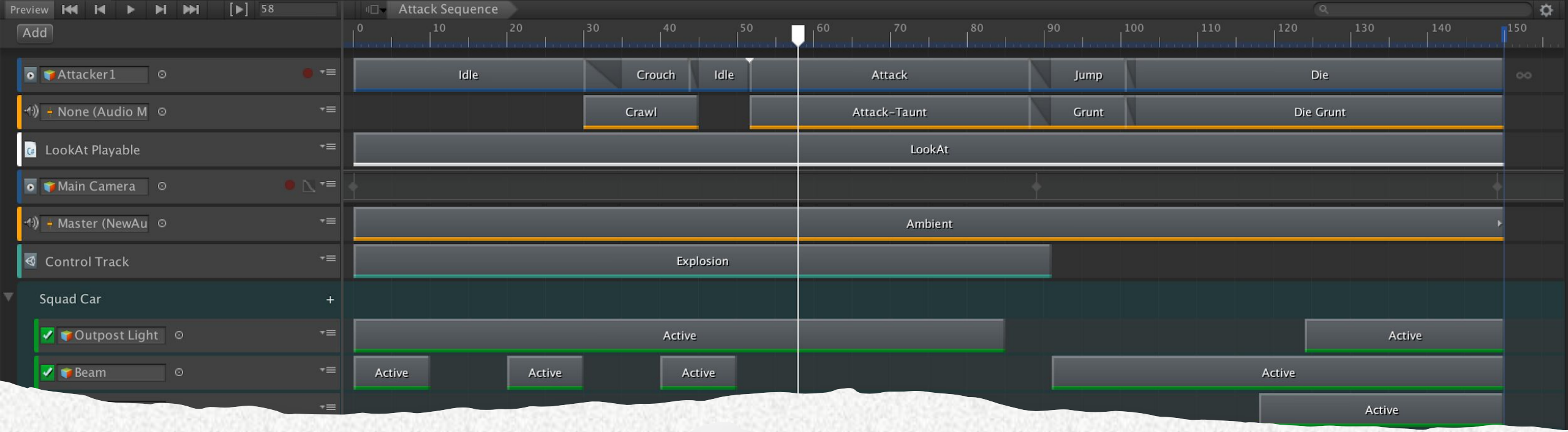
UNITY TIMELINE


Preview
58
Attack Sequence
Add
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
Attacker1
Idle
Crouch
Idle
Attack
Jump
Die
None (Audio M
Crawl
Attack-Taunt
Grunt
Die Grunt
LookAt Playable
LookAt
Main Camera
Master (NewAu
Ambient
Control Track
Explosion
Squad Car
Outpost Light
Active
Active
Beam
Active
Active
Active
Active
Active

Зачем оно надо и что это дает ?

- Создание интерактивных катсцен
- Возможность редактирования без участия программиста
- Простота изменения готовых катсцен
- Возможность переиспользования
- Результат виден сразу без необходимости запуска приложения
- Быстрая итерация и более короткий цикл

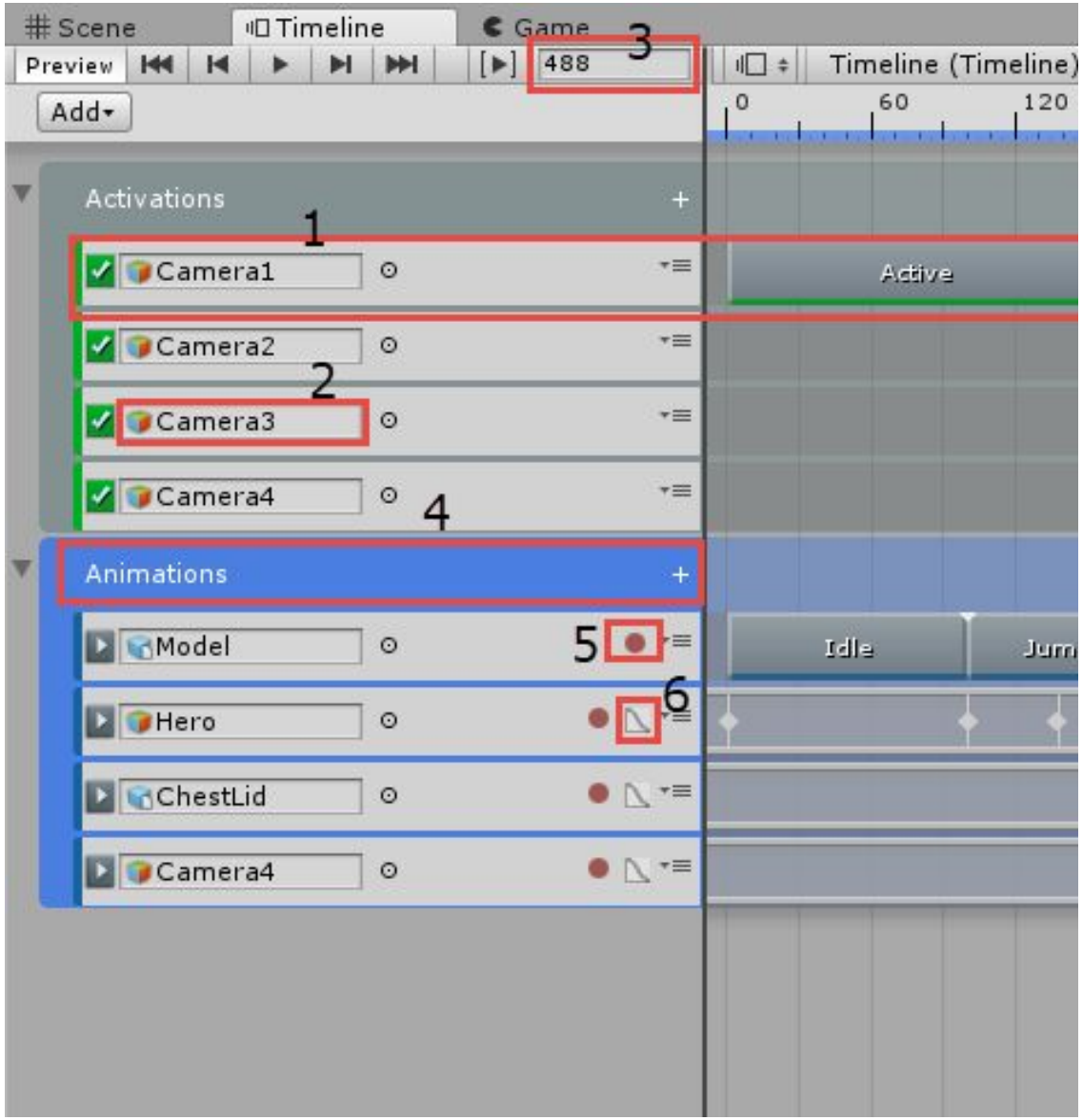
Альтернативы

- Tweens
- Coroutines
- Scripting

Полностью готовые катсцены созданные усилиями художников мультипликаторов

Из чего состоит визуальный редактор

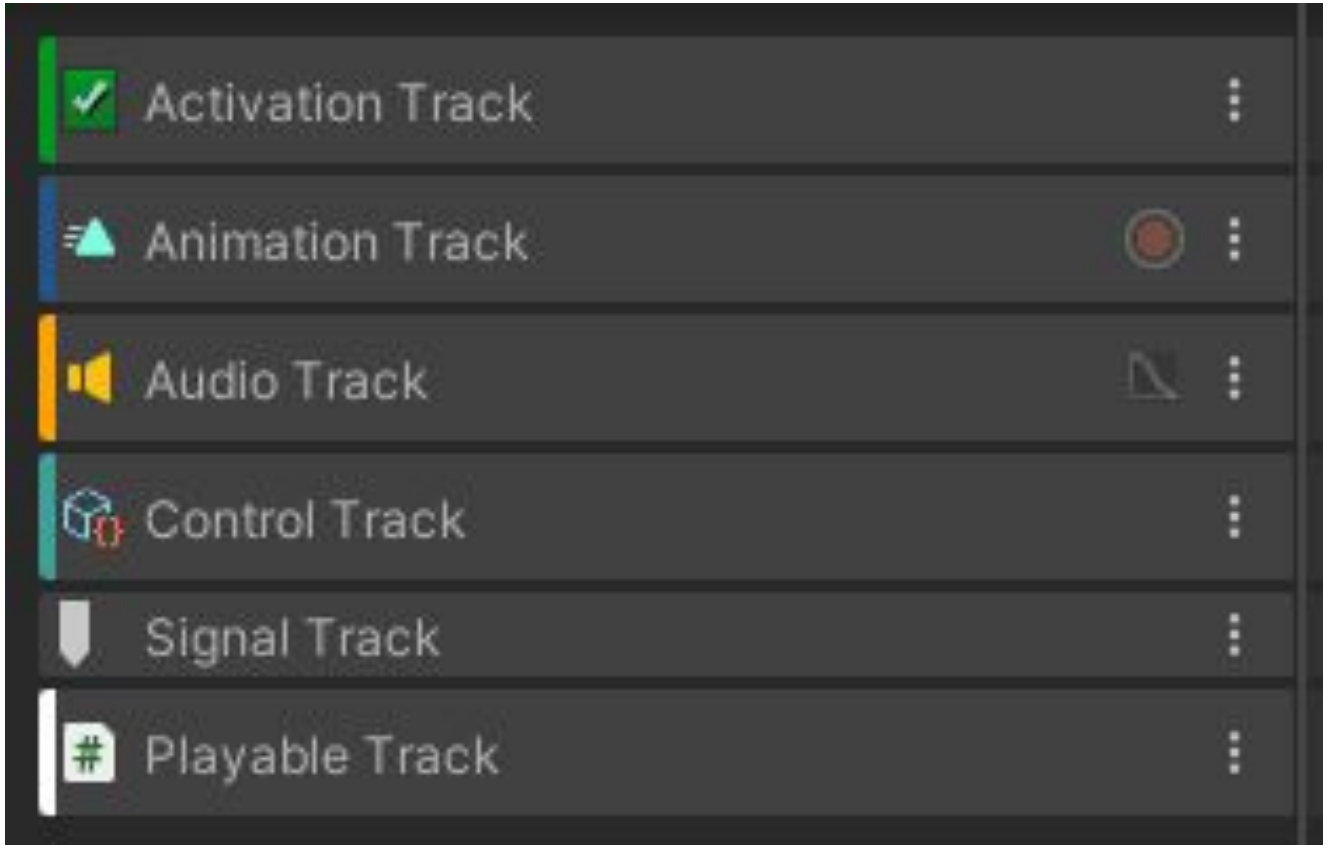
1. *Timeline Asset*
2. *Associated GameObject*
3. *Frame*
4. *Track Group*
5. *Record Button*
6. *Curves Icon*



Playable director и Exposed Reference

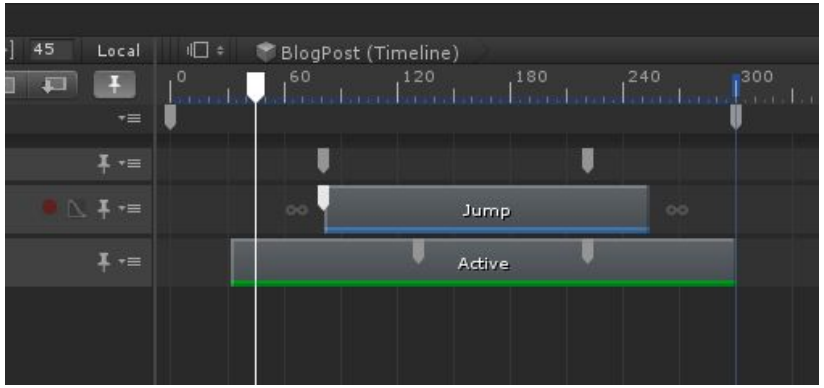
Timeline (Playable asset) это ассет и не может содержать прямые ссылки на объекты сцены. Поэтому в ассете создаются Exposed reference, в которые при исполнении с помощью Resolver (Playable director) будут проставлены необходимые объекты.

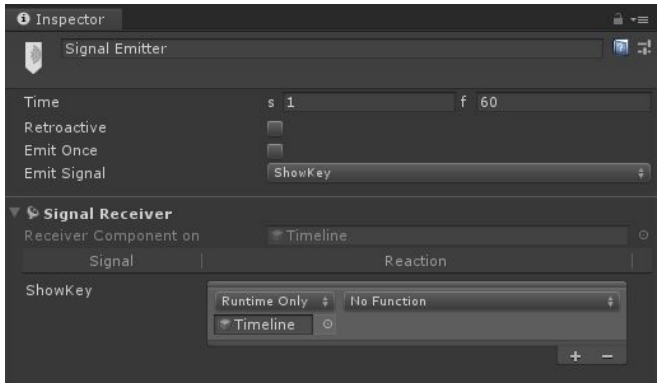
Стандартные треки Unity

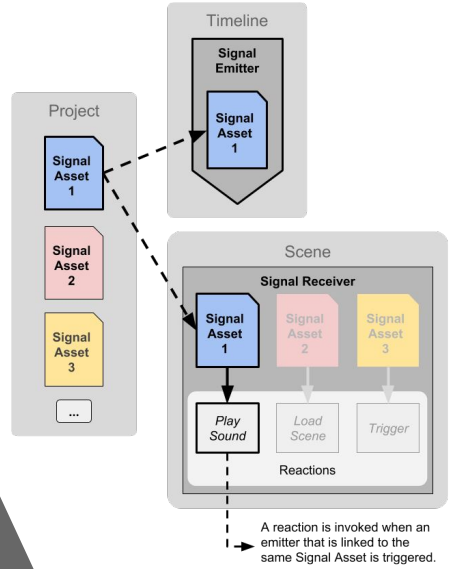


Signals

- Signal Asset - связующее звено между Emiter и Reciever. Может быть переиспользован в различных таймлайнах
- Signal Emiter - содержит ссылку на Signal Asset. На таймлайне визуализируется как маркер
- Signal Receiver - компонент со списком обработчик. В каждом обработчике есть ссылка на Signal Asset







Custom Tracks

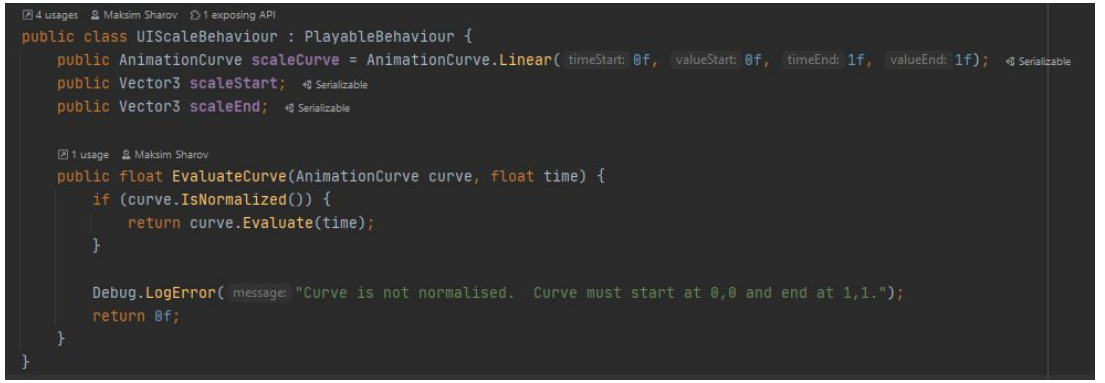
- Behaviour
- Clip
- Track
- MixerBehaviour

Behaviour

Кастомная логика. Вообще должен должен реализовывать

public override void ProcessFrame(Playable playable, FrameData info, object playerData)

Но на скрине его нет, так как ее реализует mixer



Clip

Содержит данные и сериализуется в Timeline Asset

Должна реализовывать метод

```
public override Playable CreatePlayable(PlayableGraph graph, GameObject owner)
```

Track

Может хранить в себе ссылку на GameObject, Component или Asset.

- TrackClipType обозначает с каким типов PlayableAsset будет работать данный трек.
- TrackBindingType указывает на то, с чем можно будет в дальнейшем связать данный трек.

```
[TrackColor( r: 0.855f, g: 0.8623f, b: 0.870f)]
[TrackClipType(typeof(UIScaleClip))]
[TrackBindingType(typeof(RectTransform))]
No asset usages  Maksim Sharov
public class UIScaleTrack : TrackAsset {
    Maksim Sharov
    public override Playable CreateTrackMixer(PlayableGraph graph, GameObject go, int inputCount) {
        return ScriptPlayable<UIScaleMixerBehaviour>.Create(graph, inputCount);
    }

    Maksim Sharov
    public override void GatherProperties(PlayableDirector director, IPropertyCollector driver) {
#if UNITY_EDITOR
        var comp = director.GetGenericBinding( key: this) as RectTransform;
        if (comp == null) {
            return;
        }
        var so = new SerializedObject(comp);
        var iter :SerializedProperty = so.GetIterator();
        while (iter.NextVisible( enterChildren: true)) {
            if (iter.hasVisibleChildren) {
                continue;
            }
            driver.AddFromName<RectTransform>(comp.gameObject, iter.propertyPath);
        }
#endif
        base.GatherProperties(director, driver);
    }
```

MixerBehaviour

Позволяет делать перекрытие треков на timeline. Переопределяет поведение Behaviour.

```
public class UIScaleMixerBehaviour : PlayableBehaviour {
    Maksim Sharov
    public override void ProcessFrame(Playable playable, FrameData info, object playerData) {
        var trackBinding = playerData as Transform;

        if (trackBinding == null) {
            return;
        }

        ProcessScale(trackBinding, playable);
    }

    1 usage  Maksim Sharov
    private void ProcessScale(Transform transform, Playable playable) {
        var inputCount :int  = playable.GetInputCount();
        var totalWeight = 0f;

        var blendedScale :Vector3  = Vector3.zero;
        var defaultScale :Vector3  = transform.localScale;

        for (var i = 0; i < inputCount; i++) {
            var playableInput = (ScriptPlayable<UIScaleBehaviour>)playable.GetInput(i);
            var input :UIScaleBehaviour  = playableInput.GetBehaviour();

            var inputWeight :float  = playable.GetInputWeight(i);
            totalWeight += inputWeight;
            var normalisedTime = (float)(playableInput.GetTime() / playableInput.GetDuration());
            var value :float  = input.EvaluateCurve(input.scaleCurve, normalisedTime);
            blendedScale += (input.scaleStart + (input.scaleEnd - input.scaleStart) * value) * inputWeight;
        }

        blendedScale += defaultScale * (1f - totalWeight);
        transform.localScale = blendedScale;
    }
}
```