

# 動画編集ソフトを作るための基礎知識

# お前誰？



ラグ (@\_ragg\_)

- ・フロントエンドエンジニア
- ・ TypeScript / React / Node.js / Electron / RoR / PHP
- ・Delir は趣味プロです
- ・IQ 3🌵

# 免責

- 基礎知識って言うけど、違う考え方の世界もあります。
- Delir(個人開発中の映像編集ソフト)はこう考えてるよって感じです
- 殴らないでください、、、

# 概要

- 基礎的な概念
- レンダリング処理の基礎
- 「動画」をする
  - キーフレームという概念
  - レンダリングでの処理



# 基礎的な概念

The background features a dark purple-to-black gradient with a light-colored diamond grid pattern. Scattered across the background are several triangles of varying sizes and colors, including shades of teal, purple, and brown. Some triangles are pointing up, while others are pointing down.

# 基礎的な概念

- **コンポジション**

ひとつの動画。「(自身以外の)コンポジション」「レイヤー」「オブジェクト」を内包できる。

- **レイヤー**

オブジェクトを内包し、オブジェクトの前後関係を表す。

- **オブジェクト**

出力される・出力に影響を与えるもの(画像・映像・音声・カメラなど)  
時間軸上に配置され、オブジェクトによって異なるパラメータを持つ。  
(Delirでは”クリップ”と呼んでいる)

# 基礎的な概念

- レンダラー

オブジェクトのパラメータを基に実際の画像データを生成するもの。

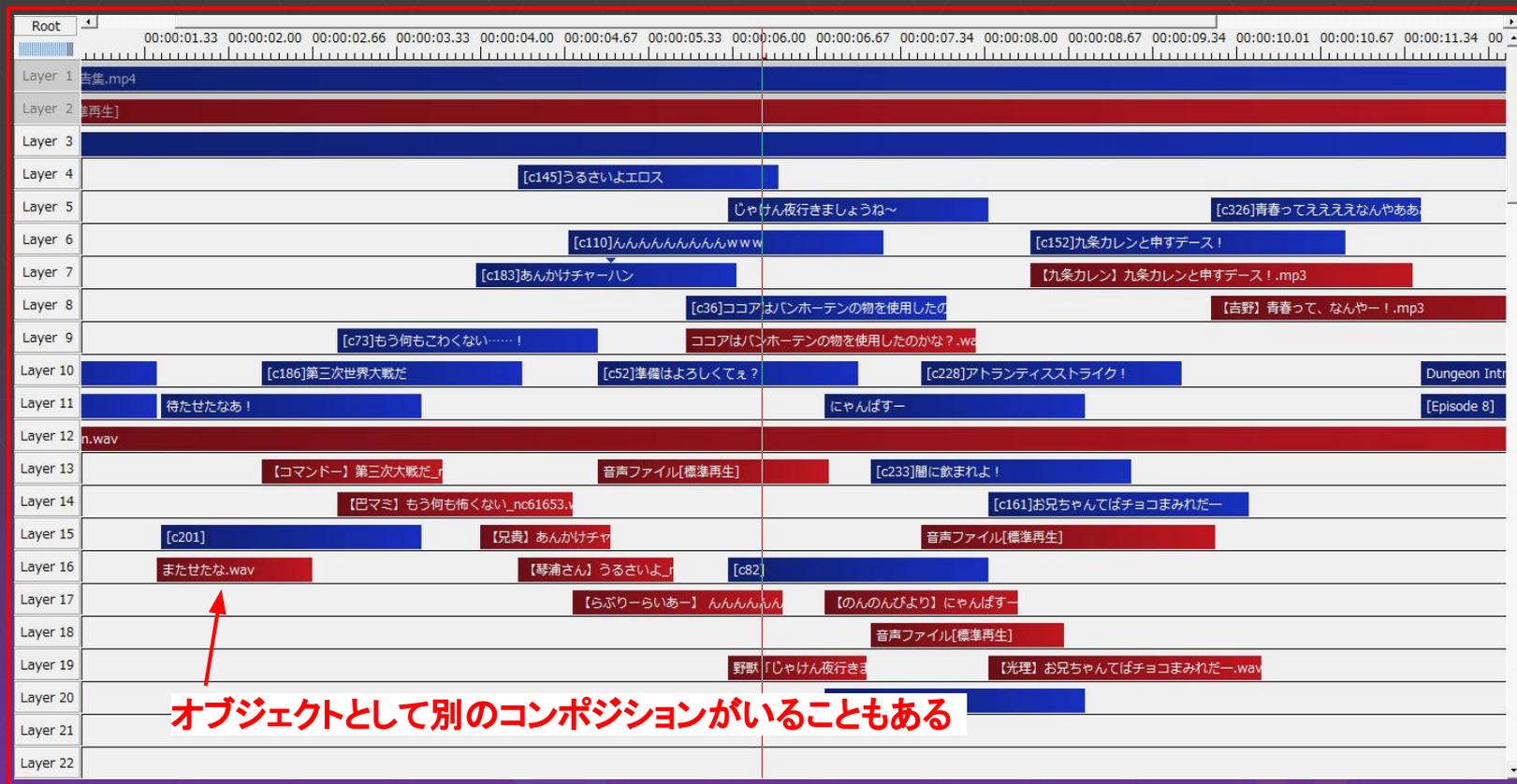
例えば図形レンダラは「位置・サイズ・形状」というパラメータを基に実際に画像を生成する。

これ！レイヤー！



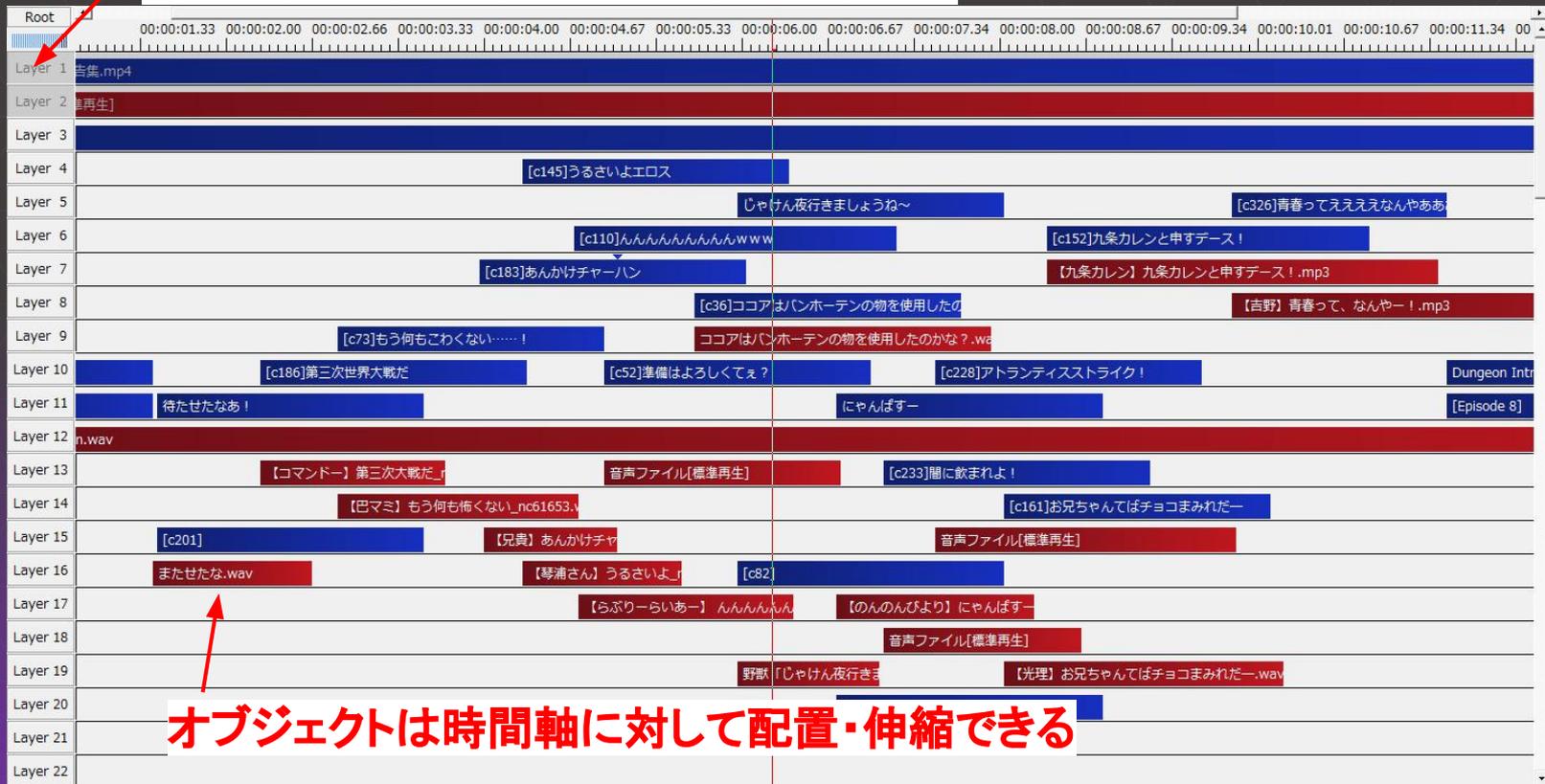
これ！オブジェクト！

# これ全部で1コンポジション！



オブジェクトとして別のコンポジションがあることもある

レイヤーが上にくるほどカメラの手前にくる



オブジェクトは時間軸に対して配置・伸縮できる

この4つを組み合わせると  
動画ができる。

- 完 -

The background features a dark purple-to-black gradient with a light-colored grid pattern. Several hollow triangles of various colors (blue, purple, brown) and sizes are scattered across the page, some pointing up and some pointing down.

# レンダリング処理の基礎

The background features a dark purple-to-black gradient with a light-colored grid pattern. Several large, semi-transparent triangles in various colors (blue, purple, teal) are scattered across the scene, some pointing up and some pointing down.

# レンダリング処理の基礎

これらの3つの要素(コンポジション・レイヤー・オブジェクト)はツリー状に展開される。

このツリーを現在の再生時間に合わせて順番にレンダリング・合成することで1フレームの映像がレンダリングされる。

# レンダリング処理の基礎

- コンポジション
  - レイヤー1
    - 0:00～ オーディオ オブジェクト
  - レイヤー2
    - 0:00～0:05 図形 オブジェクト
    - 0:10～0:15 コンポジション
  - レイヤー3
    - 0:00～ 画像 オブジェクト

現在の再生時間 0:00

コンポジション→レイヤー→オブジェクトと潜って  
現在レンダリングすべきオブジェクトを判定する

# レンダリング処理の基礎

- コンポジション

- レイヤー1

- 0:00～ オーディオ オブジェクト

- レイヤー2

- 0:00～0:05 図形 オブジェクト

- 0:10～0:15 コンポジション

- レイヤー3

- 0:00～ 画像 オブジェクト

現在の再生時間 0:00

これらのオブジェクトをレンダラがレンダリングする



IQ3

# レンダリング処理の基礎

- コンポジション
  - レイヤー1
    - 0:00～ オーディオ オブジェクト
  - レイヤー2
    - 0:00～0:05 図形 オブジェクト
    - 0:10～0:15 コンポジション
  - レイヤー3
    - 0:00～ 画像 オブジェクト

現在の再生時間 0:00

下から順に焼き込む



# レンダリング処理の基礎

- コンポジション
  - レイヤー1
    - 0:00～ オーディオ オブジェクト
  - レイヤー2
    - 0:00～0:05 図形 オブジェクト
    - 0:10～0:15 コンポジション
  - レイヤー3
    - 0:00～ 画像 オブジェクト



これを毎フレーム繰り返す



こうして動画ができる！

- 完 -

The background features a dark purple-to-black gradient with a light-colored diamond-shaped grid pattern. Scattered across the grid are several hollow triangles of various sizes and colors, including shades of blue, purple, and brown. Some triangles are upright, while others are inverted.

「動画」をする

# キーフレームという概念

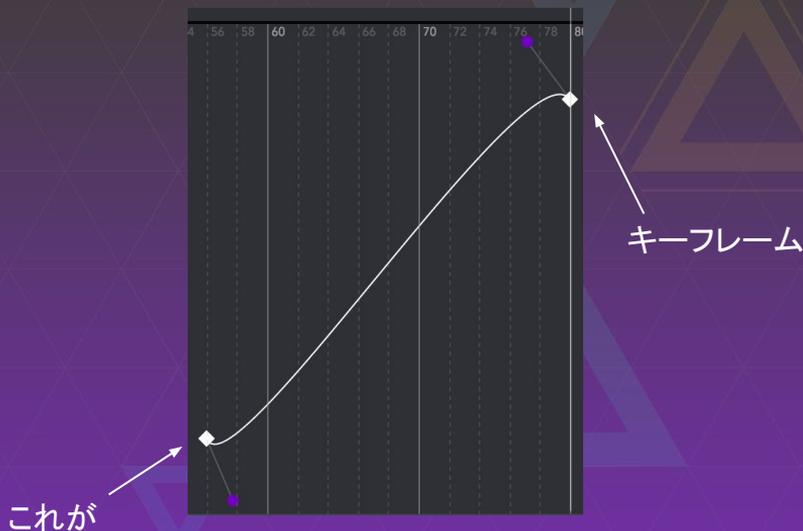
# キーフレームという概念

## - キーフレーム

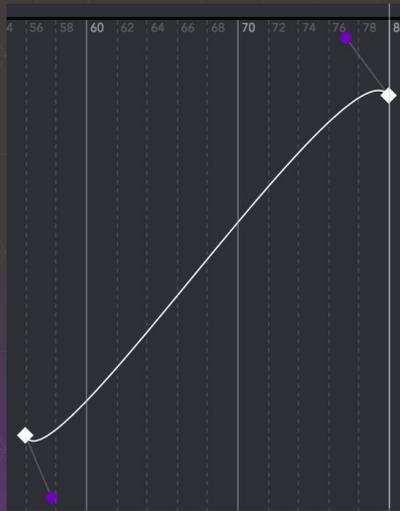
オブジェクト・エフェクトの1パラメータの変化点。

(「この時にこの値になる」という設定点。)

点から点の間は何かしらの補間関数によって補間される(三次ベジエとか)



# キーフレームという概念

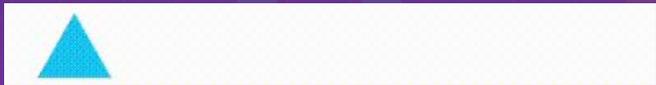


この“値”から”次の値”への推移がアニメーションになる。  
キーフレームは1プロパティの時間軸上に複数設定される。

たとえばオブジェクト 横位置に以下のようなキーフレームを打つと、「4秒目まで右に行って、5秒目には左に戻る」という動きをする。

0 ————— 0:04 ——— 0:05

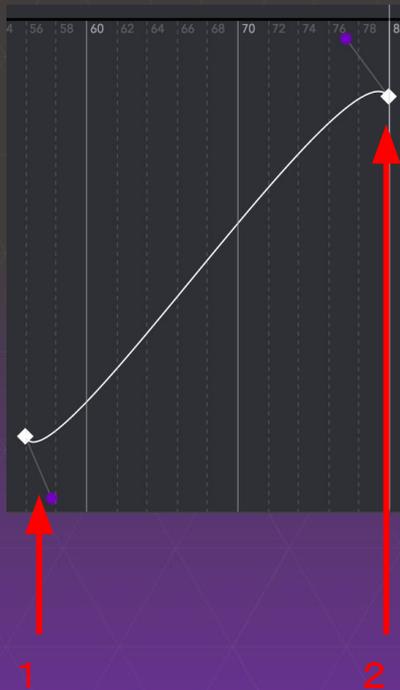
0 → 5 → 40 → -10



# レンダリングでの処理

The background features a dark purple-to-black gradient with a light-colored grid pattern. Several large, semi-transparent triangles in various colors (blue, purple, brown) are scattered across the scene, some pointing up and some pointing down.

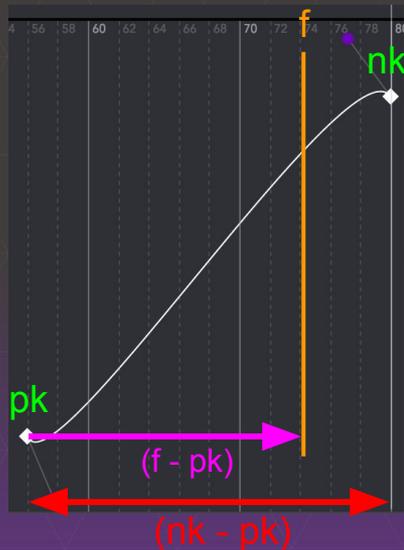
# レンダリングでの処理



キーフレームは基本、離散的に設定されるので  
キーフレームの間を何かしらの関数で補完する必要がある。  
(よく使われるのはリニアとか三次ベジェ)

キーフレーム間での現在のフレームの位置を0~1に正規化し、  
補間関数に与えて現在のフレームでどれくらいの変化量を適用する  
かを計算する。

# レンダリングでの処理 (経過量計算)



f = 現在の絶対フレーム番号

pk = 直前のキーフレームが置かれている絶対フレーム番号

nk = 次のキーフレームが置かれている絶対フレーム番号

$$\underline{\text{progress}} = (f - pk) / (nk - pk)$$

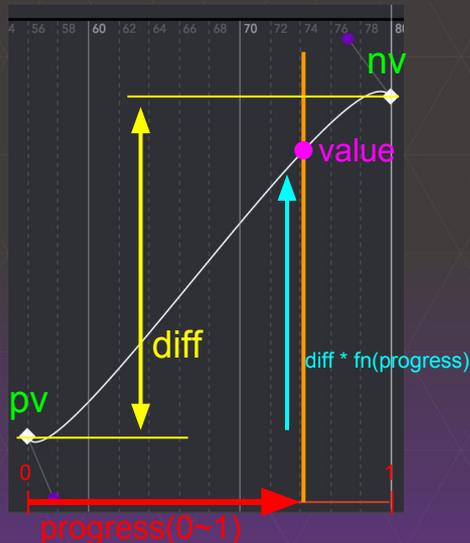
(f - pk) = 直前のキーフレームからの経過フレーム数

(nk - pk) = 次と直前のキーフレームの間のフレーム数

これでキーフレーム間の経過量を0 ~ 1に正規化出来る。

※ 間違っていたらごめん

# レンダリングでの処理 (キーフレーム値計算)



pv = 直前のキーフレームの値

nv = 次のキーフレームの値

fn = `(progress: 0~1) => float` の補間関数

diff = nv - pv (キーフレーム間の変化量)

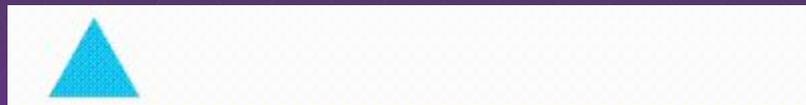
value = pv + ( diff \* fn(progress) )

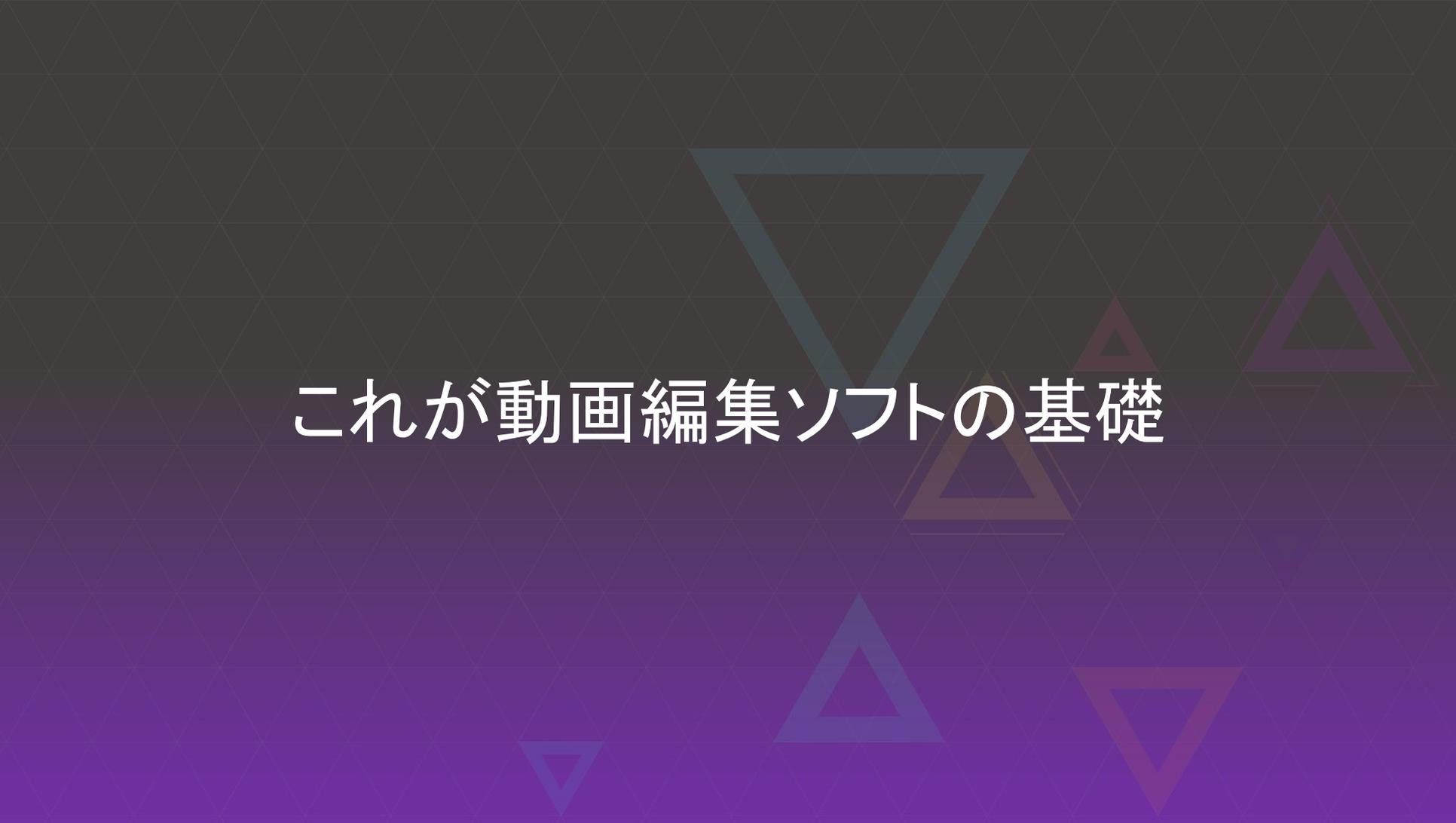
これで現在のフレームでのプロパティの値が求められる。

この計算によって得られたプロパティ値を毎フレームレンダラに渡すことでアニメーションを行える。

※ 間違っていたらごめん

これらを毎フレーム繰り返すことで  
“動画”が私たちの元へ届けられる





これが動画編集ソフトの基礎

- ・ポストエフェクト
- ・カメラと音声の先行処理
  - ・パフォーマンス
- ・実装(クラス設計とか)



地獄はまだいっぱいあるけど



これで君も動画編集ソフトを作れる

やろう！ (with TypeScript)

<https://github.com/Ragg-/delir>

ぽしまい

