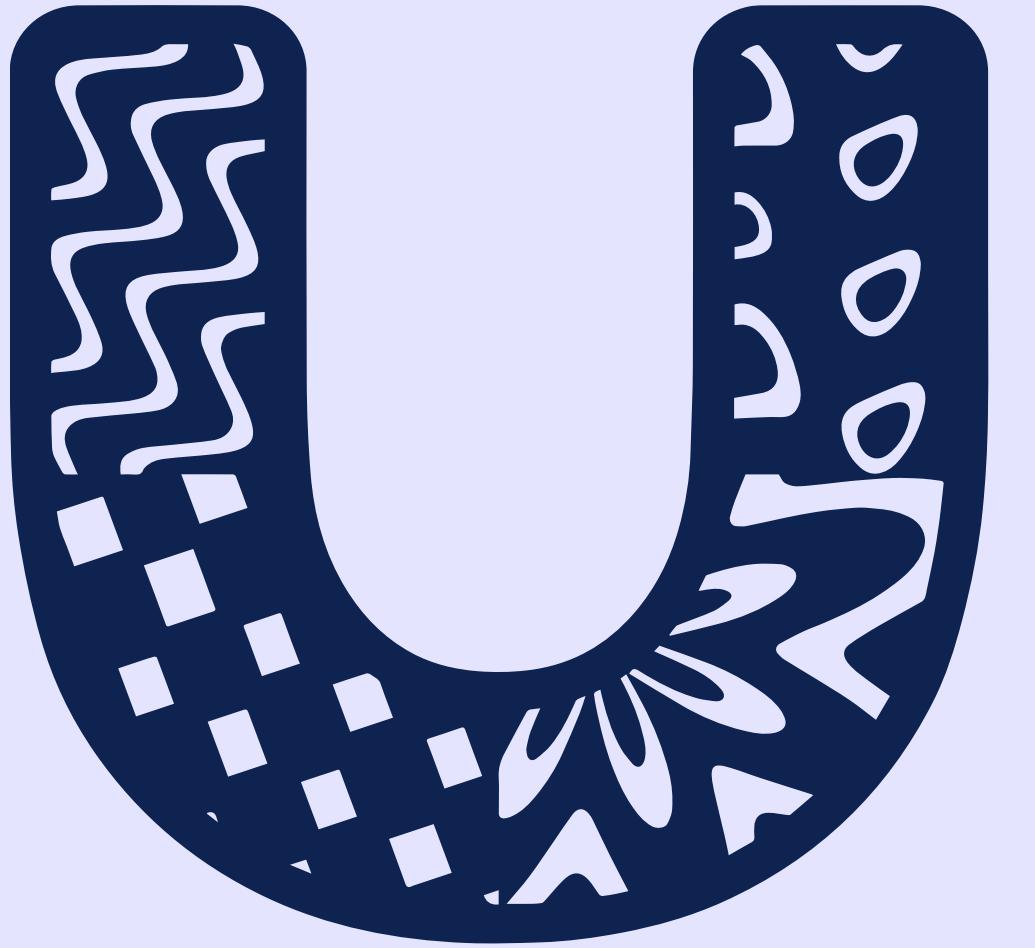


WebGL School 2023

Plus One Lesson

2023.08.26



unshift

unshift Inc.
Creative Developer

長谷川 巧

Website
<https://unshift.co.jp/>

X (Twitter)
[@_unshift](https://twitter.com/_unshift)

1人法人 / Webフロントエンド / WebGL雰囲気でやってる /
1y7mの娘 (かわいい) / クリエイティブコーディング /
バスケットボール

WebGLやるなら、たくさんのオブジェクトを動かしたいよね。

それ、ジオメトリインスタンシングでできます。

ジオメトリインスタンシングとは

ジオメトリインスタンシングとは 同じ3Dモデルを複数回描画する際の効率化手法

複数のオブジェクトを1つの頂点バッファにまとめ、
1つのドローコールで同じモデルを複数回描画できる。

各インスタンスの位置、回転、スケールなどの情報は、
別のバッファ（インスタンスバッファ）に格納される。

WebGL 1.0 では拡張機能
WebGL 2.0 では標準機能

組み込み変数などの違いもある。

以下の記事に詳しく書かれている。

<https://qiita.com/emadurandal/items/1812198d562bd1216c70>

すみません、
いつもthree.jsでやってるんで、
three.jsでやらせてください。

作例

<https://unshift.co.jp/labs/thankyoutwitter/>

Twitterのファンアート

<https://generativemasks.unshift.jp/>

Generativemasks の二次創作コンテンツのエントリー作品

<https://neort.io/art/cdsr8b4n70rqdtr2k4dg>

Neort++で行われた一般参加型の展示の出品作品

<https://fireworks.unshift.jp/>

自主制作の花火。画面をクリック / タップ

Step 1

three.js Appの雛形

three.js使いは散々やってるとは思います。

Step 2

InstancedBufferGeometryとRawShaderMaterial

この時点でのthree.jsのジオメトリインスタンシングはできる。

Step 3

インスタンスを個別に回転・変形させる

まずはどうやって個別に動かすか確認してみる。

Step 4

頂点シェーダでいい感じに並べる

少しだけ数学を頑張る。

Step 5

uniformの値をtweenさせる

ここからいい感じにしていく。

Step 6

1つのuniformの値の変化を、インスタンスごとの値に変換する

この講義でいちばん重要なところ。

Step 7

いい感じのビジュアルにする

遊び倒せ。

Step 1

three.js Appの雛形

three.js使いは散々やってるとは思います。

Step 2

InstancedBufferGeometryとRawShaderMaterial

この時点でのthree.jsのジオメトリインスタンシングはできる。

Step 3

インスタンスを個別に回転・変形させる

まずはどうやって個別に動かすか確認してみる。

Step 4

頂点シェーダでいい感じに並べる

少しだけ数学を頑張る。

Step 5

uniformの値をtweenさせる

ここからいい感じにしていく。

Step 6

1つのuniformの値の変化を、インスタンスごとの値に変換する

この講義でいちばん重要なところ。

Step 7

いい感じのビジュアルにする

遊び倒せ。

Step1

three.js Appの雛形

three.js使いは散々やってるとは思いますが。

Step1: three.js Appの雛形

Tweakpaneで再生コントロール

<https://cocopon.github.io/tweakpane/>

再生 (play)・停止 (stop)・再生速度 (timeScale)の調整
ができるようにしておく。

THREE.Clockを使って経過時間を計算

<https://threejs.org/docs/#api/en/core/Clock>

THREE.ClockのelapsedTimeで経過時間が取得できるが、
startメソッドでリセットされてしまうので、再生・停止機能に対応できない。

かわりにgetDeltaメソッドでフレームごとに変数に加算して、経過時間を取っておく。その際に加算する値にtimeScaleをかければ、再生速度をコントロールできる。

Step1: three.js Appの雛形

TrackballControlsでグリグリ

<https://threejs.org/docs/#examples/en/controls/TrackballControls>

OrbitControlsよりもzoomがなめらかなので好き。

Step 1

three.js Appの雛形

three.js使いは散々やってるとは思います。

Step 2

InstancedBufferGeometryとRawShaderMaterial

この時点でthree.jsのジオメトリインスタンシングはできる。

Step 3

インスタンスを個別に回転・変形させる

まずはどうやって個別に動かすか確認してみる。

Step 4

頂点シェーダでいい感じに並べる

少しだけ数学を頑張る。

Step 5

uniformの値をtweenさせる

ここからいい感じにしていく。

Step 6

1つのuniformの値の変化を、インスタンスごとの値に変換する

この講義でいちばん重要なところ。

Step 7

いい感じのビジュアルにする

遊び倒せ。

Step2

InstancedBufferGeometryと RawShaderMaterial

この時点でthree.jsのジオメトリインスタンシングはできる。

InstancedBufferGeometry

<https://threejs.org/docs/#api/en/core/InstancedBufferGeometry>

ジオメトリインスタンシングは、WebGL 1.0と2.0で使い方
が異なるが、three.jsのInstancedBufferGeometryは内
部的にその違いを吸収してくれている。便利。

RawShaderMaterial

<https://threejs.org/docs/#api/en/materials/RawShaderMaterial>

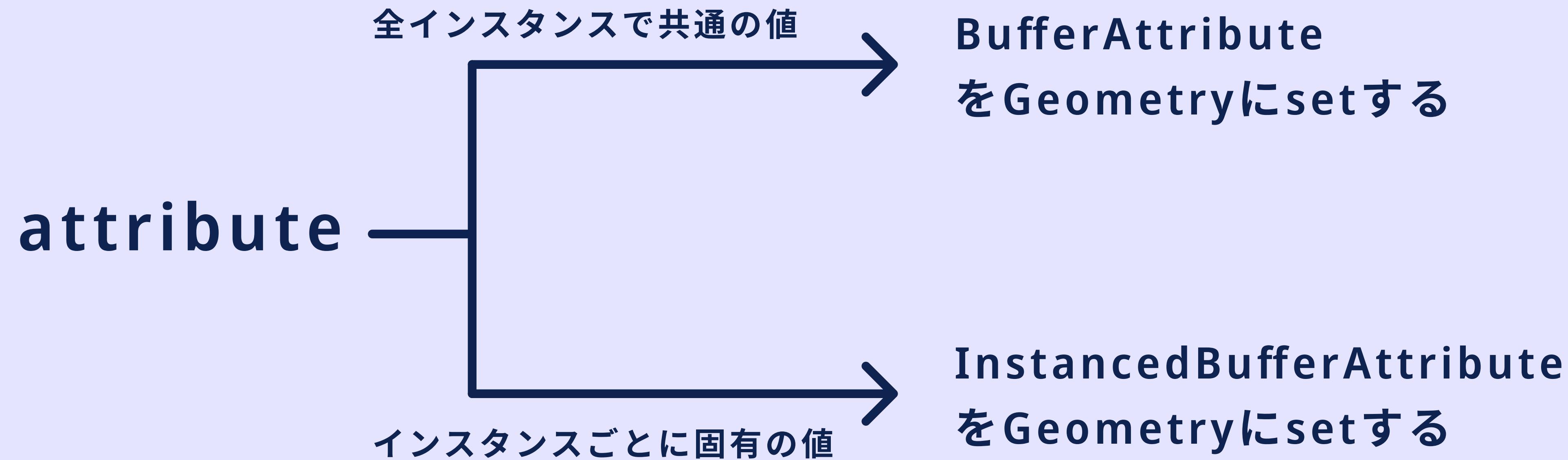
独自のシェーダを使う際に用いるMaterial。

似たものとしてShaderMaterialがあるが、そちらは組み込みのuniformやattributeがすでに宣言されているので、シェーダにそれらを書く必要がない。

RawShaderMaterialは全て自分で書く必要があるが、シェーダの全貌が見えるので、こちらがおすすめ。

シェーダで必要な値たちはどこに指定する？

uniform → RawShaderMaterialに指定する



Step 1

three.js Appの雛形

three.js使いは散々やってるとは思います。

Step 2

InstancedBufferGeometryとRawShaderMaterial

この時点でのthree.jsのジオメトリインスタンシングはできる。

Step 3

インスタンスを個別に回転・変形させる

まずはどうやって個別に動かすか確認してみる。

Step 4

頂点シェーダでいい感じに並べる

少しだけ数学を頑張る。

Step 5

uniformの値をtweenさせる

ここからいい感じにしていく。

Step 6

1つのuniformの値の変化を、インスタンスごとの値に変換する

この講義でいちばん重要なところ。

Step 7

いい感じのビジュアルにする

遊び倒せ。

Step3

インスタンスを個別に回転・変形させる

まずはどうやって個別に動かすか確認してみる。

インスタンスを個別で動かすための準備

インスタンスを個別で動かすために、

インスタンスの通し番号 (`float instanceIndex`) (※) と、

0~1のランダムな値4つ (`vec4 randomValues`) を

`InstancedBufferAttribute` で `Geometry` に set

あと経過時間 (`time`) を `uniform` として `Material` に set

※ WebGL2.0では、インスタンスの通し番号にアクセスできる

`gl_InstanceID` という値があるが、今回は WebGL1.0 でも OK な実装にするため使わない

Step3: インスタンスを個別に回転・変形させる

個別で動かすには、例えば…

- x座標にrandomValue.x * 10.0を指定する
- y軸に回転させるときの位相差として
`instanceIndex * PI`を指定する

など

Step 1

three.js Appの雛形

three.js使いは散々やってるとは思います。

Step 2

InstancedBufferGeometryとRawShaderMaterial

この時点でのthree.jsのジオメトリインスタンシングはできる。

Step 3

インスタンスを個別に回転・変形させる

まずはどうやって個別に動かすか確認してみる。

Step 4

頂点シェーダでいい感じに並べる

少しだけ数学を頑張る。

Step 5

uniformの値をtweenさせる

ここからいい感じにしていく。

Step 6

1つのuniformの値の変化を、インスタンスごとの値に変換する

この講義でいちばん重要なところ。

Step 7

いい感じのビジュアルにする

遊び倒せ。

Step4

頂点シェーダでいい感じに並べる

少しだけ数学を頑張る。

Step4: 頂点シェーダでいい感じに並べる

オフセット座標をattributeではなく
頂点シェーダ内で計算してみる

オフセット位置を時間経過やパラメータの変化で複雑に変
化させるため、attributeで渡すのではなく、頂点シェー
ダ内で計算してみる。

Step5: uniformの値をtweenさせる

剰余算(割り算の余りを求める計算) を使用してインスタンスを並べる

instanceIndexをnumPolygonsの値で割った余りの数値
をpolygonIndexとし、その値を使用してインスタンスを
グループ分けして並べる。



```
1 float polygonIndex = mod(instanceIndex, numPolygons);
```

Step4: 頂点シェーダでいい感じに並べる

媒介変数表示を使った 正多角形の方程式

<https://slpr.sakura.ne.jp/qp/polygon-spirograph/#poly2d>

Step 1

three.js Appの雛形

three.js使いは散々やってるとは思います。

Step 2

InstancedBufferGeometryとRawShaderMaterial

この時点でのthree.jsのジオメトリインスタンシングはできる。

Step 3

インスタンスを個別に回転・変形させる

まずはどうやって個別に動かすか確認してみる。

Step 4

頂点シェーダでいい感じに並べる

少しだけ数学を頑張る。

Step 5

uniformの値をtweenさせる

ここからいい感じにしていく。

Step 6

1つのuniformの値の変化を、インスタンスごとの値に変換する

この講義でいちばん重要なところ。

Step 7

いい感じのビジュアルにする

遊び倒せ。

Step5

uniformの値をtweenさせる

ここからいい感じにしていく。

Step5: uniformの値をtweenさせる

GSAPを使用してuniformの値を変化させる

<https://greensock.com/gsap/>

GSAPはJavaScriptのアニメーションライブラリ。数値をさまざまな条件で変化させることができる。今回はuniformの値を変化させ、WebGLのオブジェクトをアニメーションさせるために使用する。

※ 使用時はライセンスに注意。たとえば有料会員登録があるようなサイトには使用できない。

Step5: uniformの値をtweenさせる

Easing (イージング)

<https://easings.net/ja>

アニメーションや遷移の速度や変化を制御するための概念。アニメーションがどのように始まり、どのように進行し、どのように終わるかを調整するために使用される。

GSAPではイージングを `ease` オプションに指定して、値の変化のさせ方を制御できる。

Step 1

three.js Appの雛形

three.js使いは散々やってるとは思います。

Step 2

InstancedBufferGeometryとRawShaderMaterial

この時点でのthree.jsのジオメトリインスタンシングはできる。

Step 3

インスタンスを個別に回転・変形させる

まずはどうやって個別に動かすか確認してみる。

Step 4

頂点シェーダでいい感じに並べる

少しだけ数学を頑張る。

Step 5

uniformの値をtweenさせる

ここからいい感じにしていく。

Step 6

1つのuniformの値の変化を、インスタンスごとの値に変換する

この講義でいちばん重要なところ。

Step 7

いい感じのビジュアルにする

遊び倒せ。

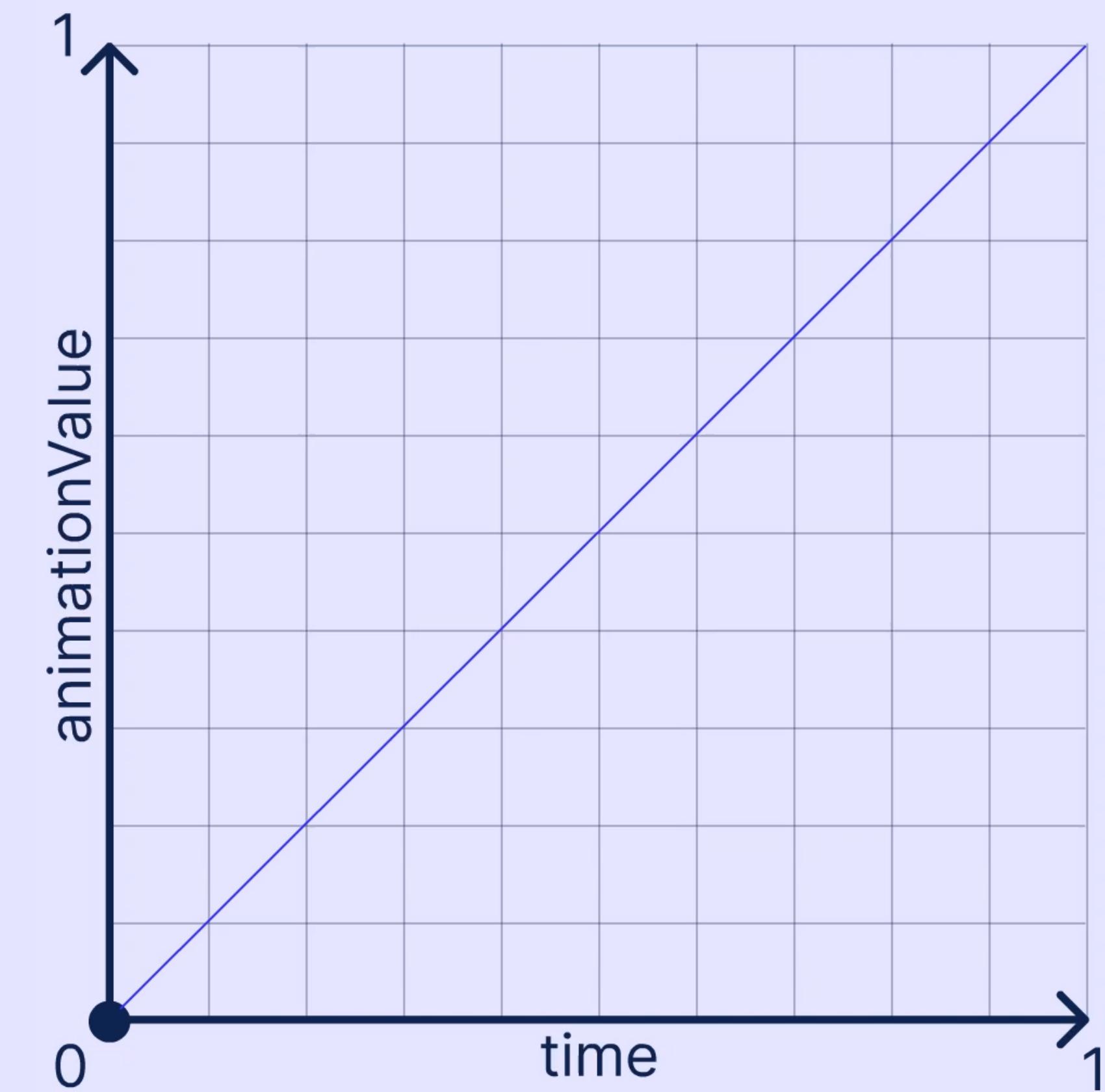
Step6

1つのuniformの値の変化を
インスタンスごとの値に変換する

この講義でいちばん重要なところ。

Step6: 1つのuniformの値の変化を、インスタンスごとの値に変換する

JavaScript側では、Linearに変化する値(animationValue)を
1つだけ用意し、uniformに渡す



[グラフの動画へのリンク](#)

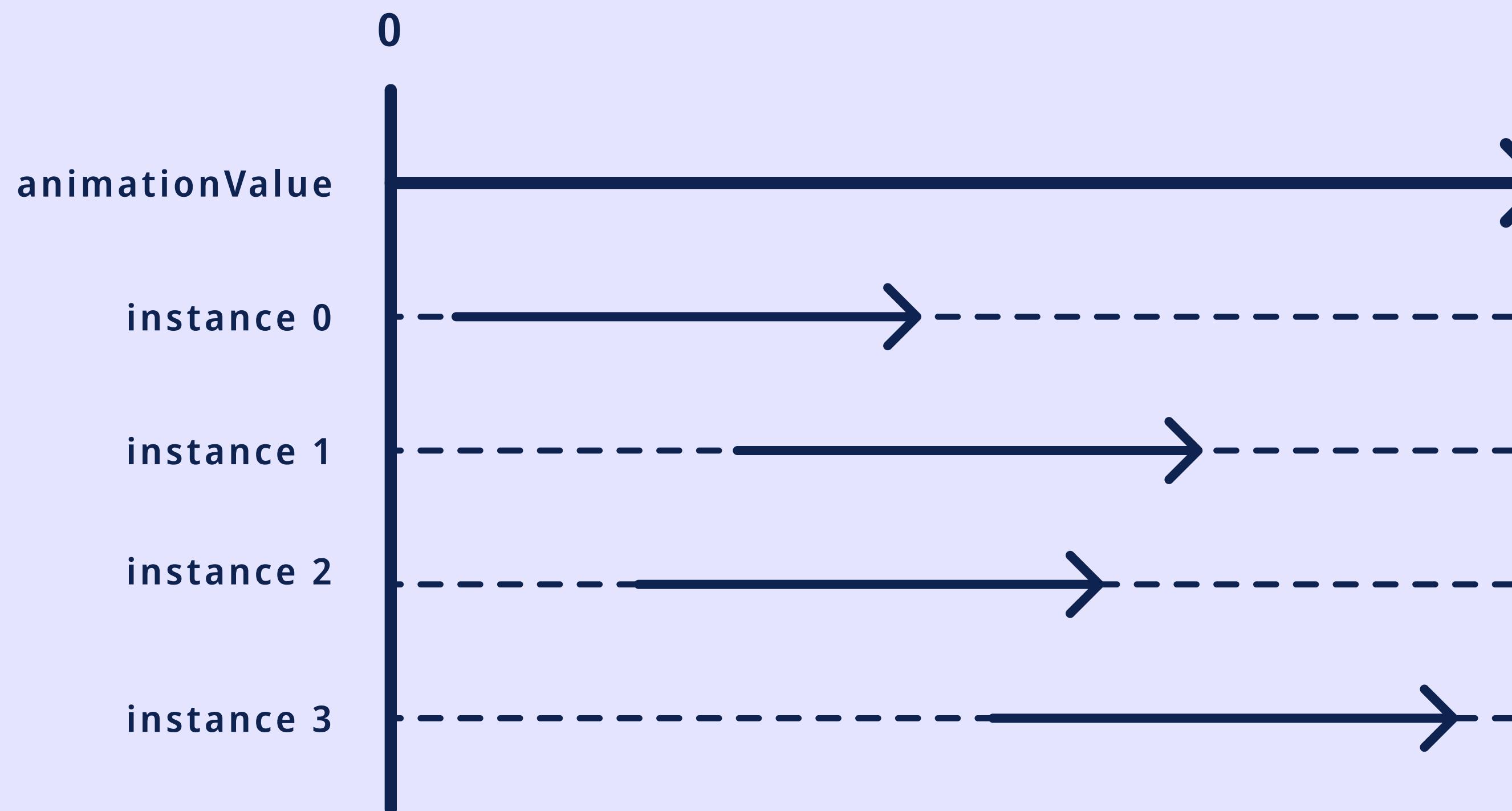
Step6: 1つのuniformの値の変化を、インスタンスごとの値に変換する

GLSLでEasingを使う

<https://github.com/glslify/glsl-easings>

Step6: 1つのuniformの値の変化を、インスタンスごとの値に変換する

animationValueが $0 \rightarrow 1$ に変化する間に、
インスタンスごとに異なる時間差で $0 \rightarrow 1$ になる値が欲しい



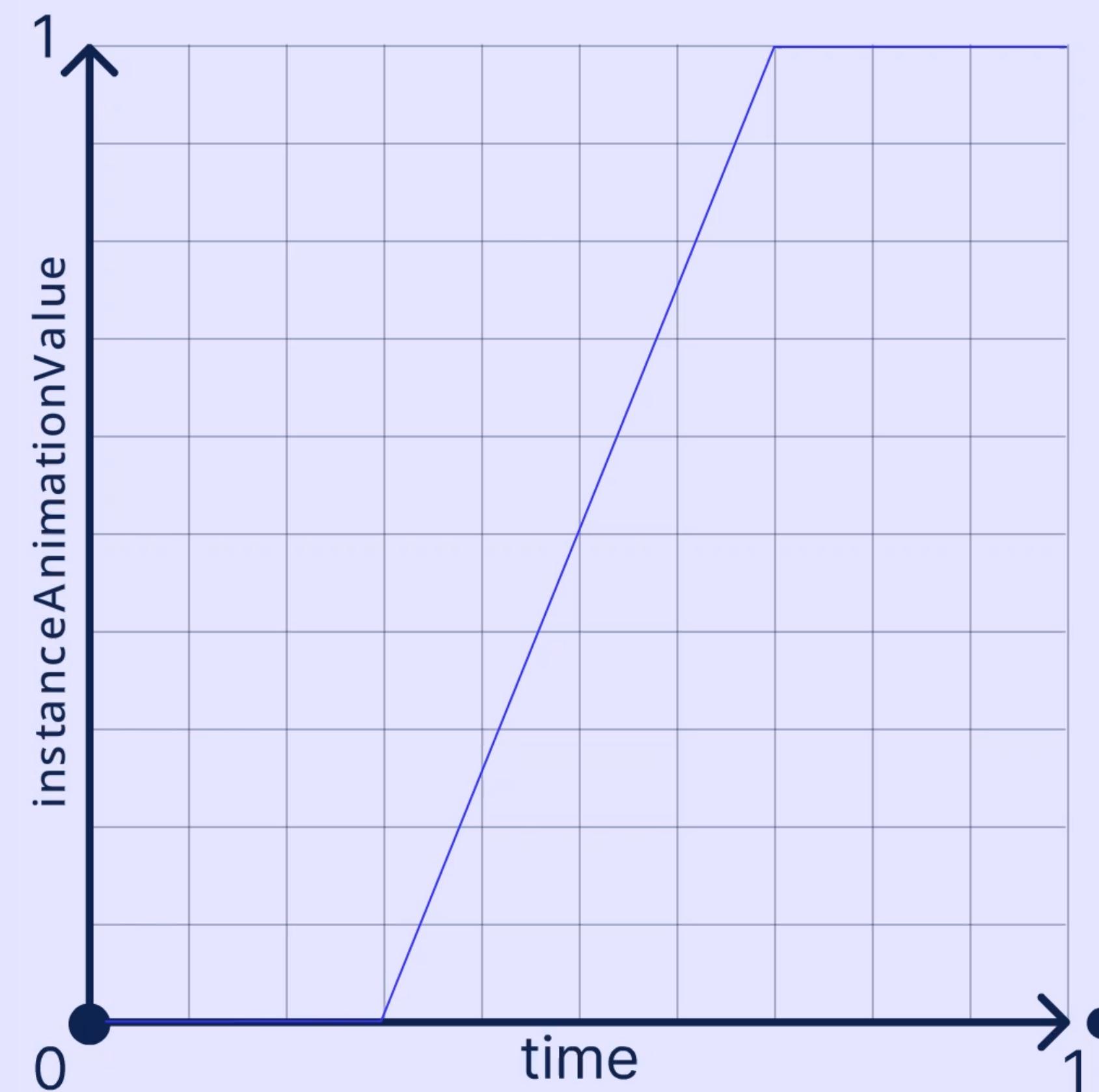
Step6: 1つのuniformの値の変化を、インスタンスごとの値に変換する

getIndivisualAnimationValue

```
● ● ●  
1 float getIndivisualAnimationValue(  
2     float animationValue,  
3     float randomValue,  
4     float minOffsetRatio,  
5     float maxOffsetRatio  
6 ) {  
7     // アニメーション開始のオフセット時間の割合  
8     float offsetRatio = minOffsetRatio + (maxOffsetRatio - minOffsetRatio) * randomValue;  
9  
10    // アニメーション時間の割合  
11    float durationRatio = 1.0 - maxOffsetRatio;  
12  
13    // animationValueが  
14    // offsetRatio から offsetRatio + durationRatioに変化する間,  
15    // 最終的に得られる値 (indivisualAnimationValue) を0 ~ 1として返したい  
16    float indivisualAnimationValue = max(0.0, animationValue - offsetRatio);  
17    indivisualAnimationValue = min(indivisualAnimationValue, durationRatio);  
18    indivisualAnimationValue = indivisualAnimationValue / durationRatio;  
19    return indivisualAnimationValue;  
20 }
```

Step6: 1つのuniformの値の変化を、インスタンスごとの値に変換する

グラフで表すとこうなる

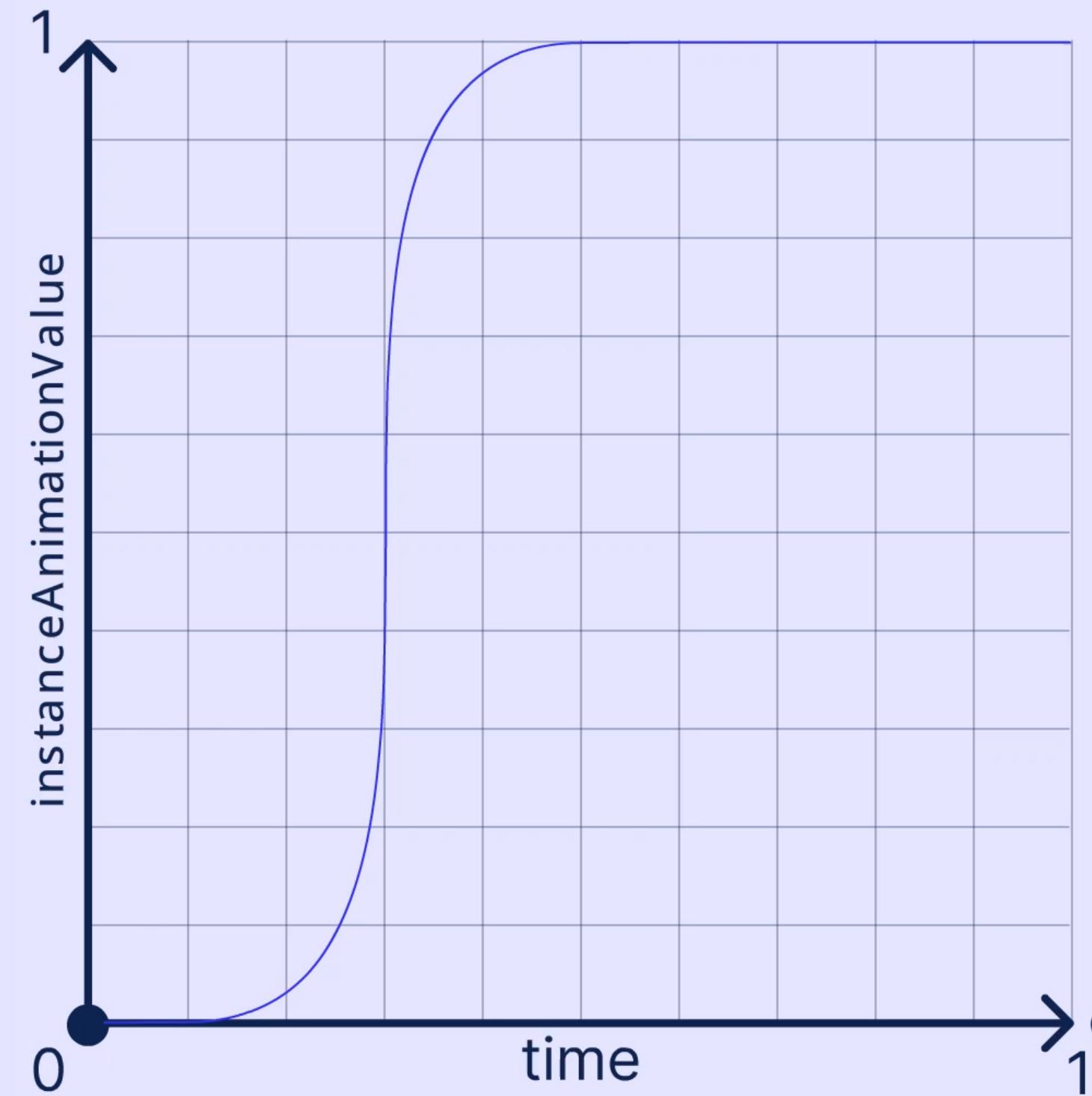


[グラフの動画へのリンク](#)

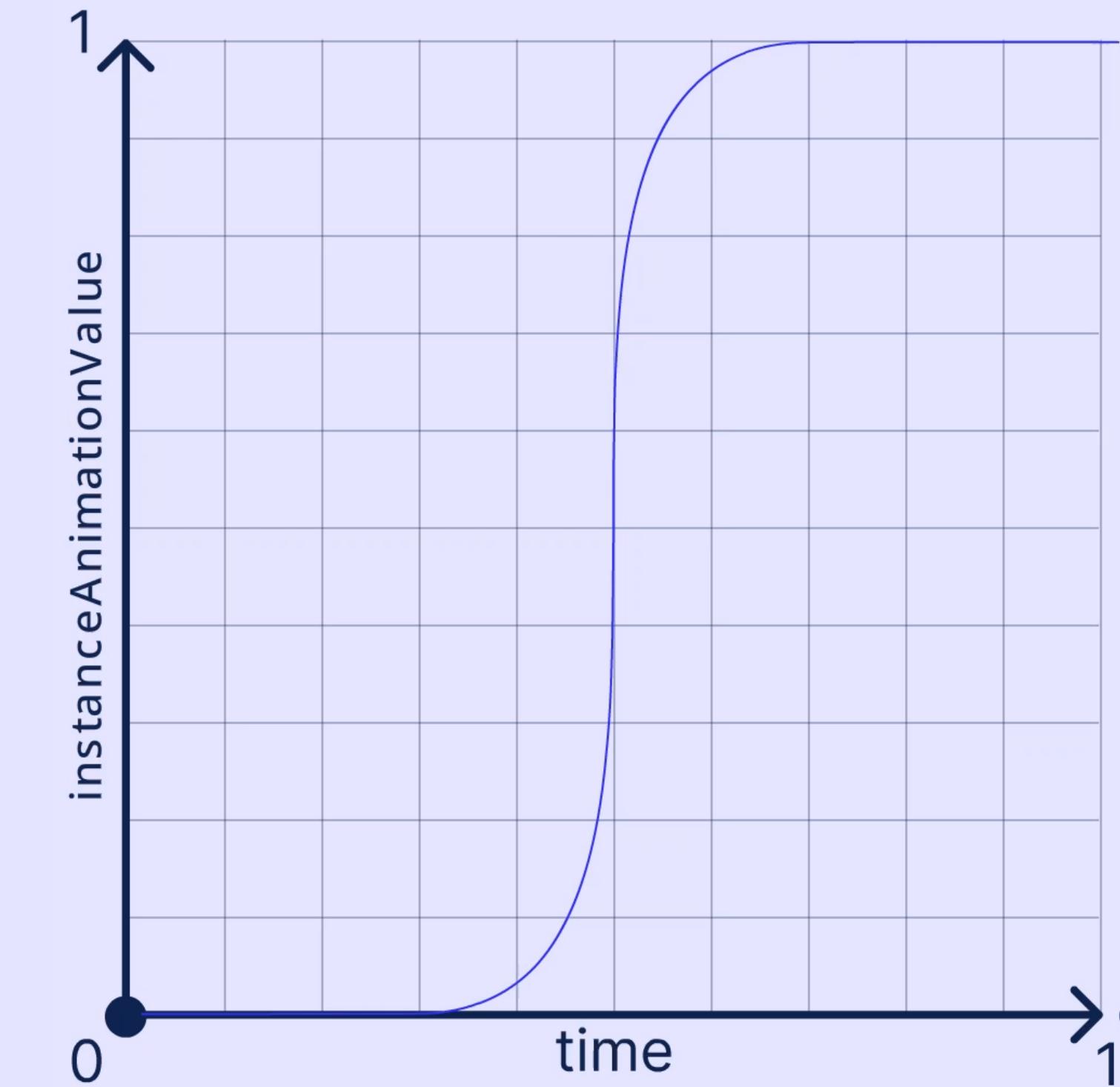
Step6: 1つのuniformの値の変化を、インスタンスごとの値に変換する

最後にイージングをかけると、

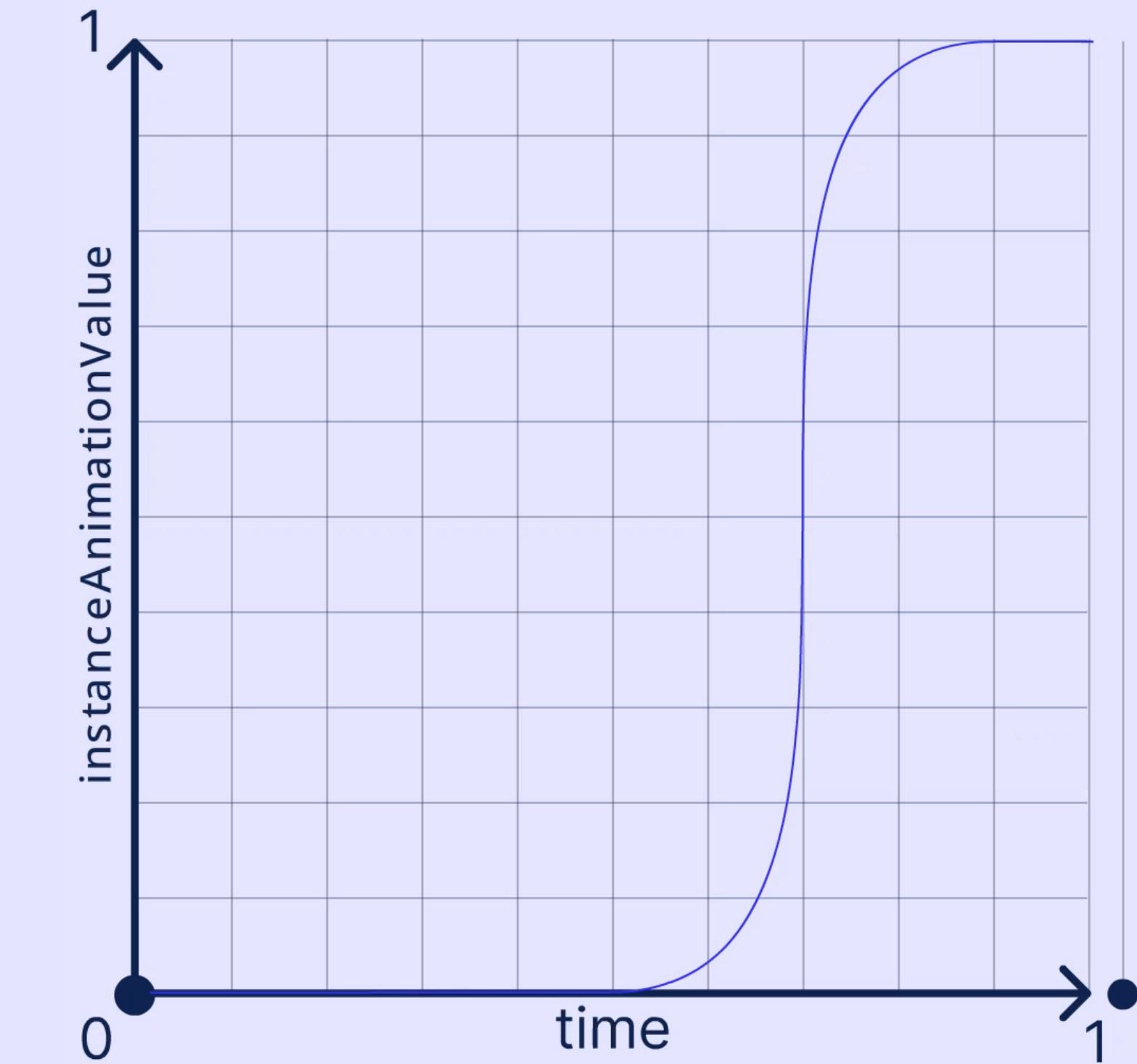
時間差でイージングがかかったアニメーションを実現できる！



[グラフの動画へのリンク](#)



[グラフの動画へのリンク](#)



[グラフの動画へのリンク](#)

Step 1

three.js Appの雛形

three.js使いは散々やってるとは思います。

Step 2

InstancedBufferGeometryとRawShaderMaterial

この時点でのthree.jsのジオメトリインスタンシングはできる。

Step 3

インスタンスを個別に回転・変形させる

まずはどうやって個別に動かすか確認してみる。

Step 4

頂点シェーダでいい感じに並べる

少しだけ数学を頑張る。

Step 5

uniformの値をtweenさせる

ここからいい感じにしていく。

Step 6

1つのuniformの値の変化を、インスタンスごとの値に変換する

この講義でいちばん重要なところ。

Step 7

いい感じのビジュアルにする

遊び倒せ。

Step7

いい感じのビジュアルにする

遊び倒せ。

このビジュアルを得るためにやったこと（抜粋）

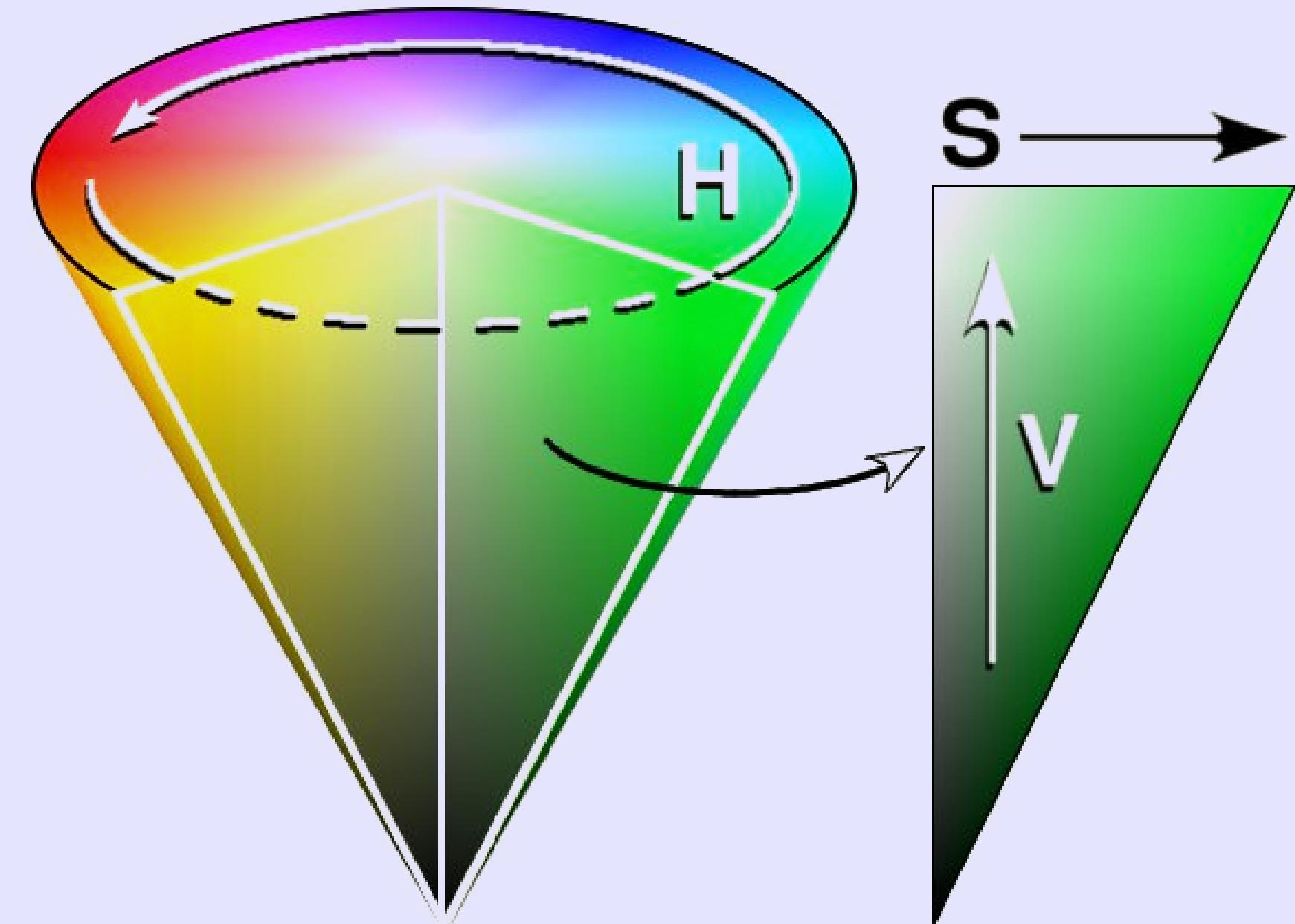
- MaterialのblendingにTHREE.AdditiveBlending (加算合成)
- 背景を黒に (加算合成を活かすため)
- インスタンスの色を時間経過で変える + インスタンスごとに少し色のばらつき
(HSV色空間を使用して着色)
- カメラを遠ざける + 図形の描画半径を大きく (より迫力がある画)
- インスタンス数を増やす (数が多いは正義)
- インスタンスの個別の回転の周期をバラバラに (より浮遊感を出す)
- インスタンスの配置にはらつきをもたせる (引きで見るとザラッとした質感になる)
- アニメーション変形後の図形を立体図形 (八面体) に
- ポストエフェクトでBloomエフェクトを追加 (ちょっとずるい)

Step7: いい感じのビジュアルにする

HSV色空間

<https://www.peko-step.com/html/hsv.html>

HSV色空間は色相 (Hue)、彩度 (Saturation, Chroma)、明度 (Value, Brightness) の三つの成分からなる色空間。HSBモデル (Hue, Saturation, Brightness) とも言われる。
似たモデルにHSL色空間がある。



まとめ

WebGLのジオメトリインスタンシングは
軽快に動作しつつ、かつインパクトのあるビジュアルを
作ることができる！

あと、three.jsだと超お手軽。