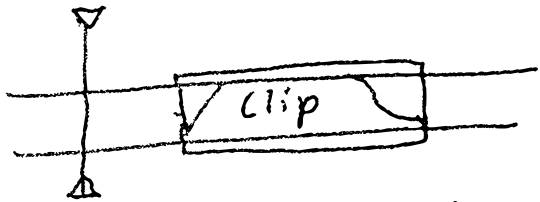
KAIROS BLENDING

We want to have tracks that are transparent:



Outside a clip the track shall have no effect on the final result.

Inside a clip we want to be able to fade into the clip.

For that to work we need to "blend into the background":

sample: (Time, Background) → Value

Typically we can implement this function with a lerp:

sample $(t, bg) = v \cdot \alpha + bg \cdot (1 - \alpha)$

with v being the clip value at t

and α the alpha value at t

So when calling sample from downstream we need to hand over a background on which the track can blend on top.

Performance

$$v \cdot \alpha + bg \cdot (1 - \alpha)$$

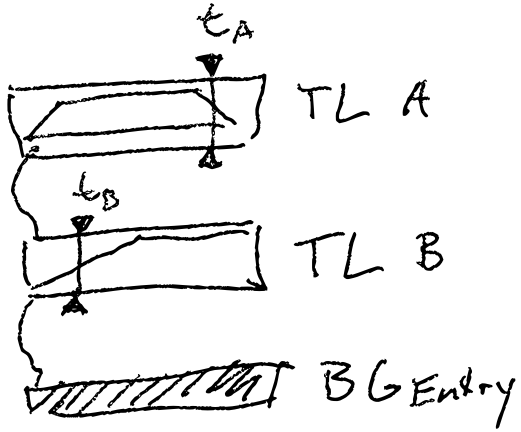
returns v for fully opaque clips

In these scenarios it would be nice to not evaluate the background value. Note that in big compositions with many containers (timelines, composers..) evaluating bg might be more expensive than sampling the local clip..

Idea: hand over a function to compute the background that we only call if necessary.

In complex setups this leads to:

- building delegates depending on each other. ↑
- then we sample from top to bottom. ↓



TL A samples at t_A. It needs a background BGA. As it is not opaq. it calls the delegate BGFuncA.

This one just samples TL B wich samples BGFuncB at t_B = BGEntry - as it also isn't fully opaque at t_B.