OTIO::Thoughts V2 Summary

Motivations (Relevant Applications)

- History of physical editing & how that led to current paradigms
- Nomenclature of editing operations
- Survey of existing systems and designs
    - From paper 1 review: "In particular, the paper must clearly show what the problems are (how does "ambiguity" and "inconsistency" manifest itself?) and also demonstrate how the proposed formalism addresses those issues. While I am aware that there are many non-trivial details and corner cases that must be handled correctly to produce robust video editing software, I'm not convinced that there is a fundamental/pervasive set of deficiencies across the best existing video editing software."
    - Include a bridge between the nomenclature and current implementations (example: how much consistency is there in a "ripple" or "roll")
- Novelty
    - Foundational graphics, physics and mathematical papers express the basic relationships and structure of geometry and space, we want to apply that rigor to media that has both time and space
        - Novelty: proposing that editorial operations are not ad hoc but can be defined as mathematical operations and predications on sets, intervals, and points
        - Novelty: proposing that by using this consistent mathematical framework we can express temporal sampling and reconstruction formally rather than by the state of the art, which is to use ad hoc heuristics (frame snapping) to conform rates to one another
    - **Differentiable time algebra for editorial systems?**
- Foundational Applications:
    - Can show that by using the above novelties we can apply transformations consistently across domains, for example to 3d sampled data and media in a system that intermixes the two concepts
    - Editing AST -- show that theoretically an "editorial compiler" could be built to simplify timeline representations down to a common minimum representation
- Example applications (DCC, USD, etc.)
- ***Internal to Pixar: we need a temporal evaluation engine in order to evaluate the more advanced temporal transformations we are proposing in:***
    - ***pitch doctor***

    - ***presto***
    - ***For freestanding scripts***
    - ***...potentially in USD eventually***

Mathematical Model

- Primitives
    - Point
        - *Maybe a point is a coordinate and an +inf duration*
    - Interval
        - ContinuousInterval
            - Origin, duration, clusivity
        - MappedInterval
            - For any input time, there’s an output time
        - ClockedInterval
            - Origin, duty cycle { on frequency, on/off ratio }
    - Transformation
        - LinearTransformation (Affine, offset, translate)
- Temporal Coordinate systems
    - Application of hierarchical coordinate systems to temporal domain
    - Labelling specific coordinate systems for reference (wall clock time vs media time)
        - *Time warps are bad or broken*
        - *Sampling functions are unclear*
- Spatial Coordinate systems (pixels, mm, unit less)...
    - *Annotations*
    - *Image dimensions and transformations*
    - *Autodesk’s universal coordinate system thing*
- Mixed domain operator graphs
    - Notion of basis functions and transformations between coordinate systems in a given domain
    - *“What resolution is this image”*
    - *Sound vs picture transformations*
    - *Temporal vs spatial transformations*
    - Graphs with an actor in the loop, like a video game with a player. The player expresses inputs which the system evaluates in addition to its pre-programmed routines and data
- Sampling Mapping vs Operator Domains
    - Sample Mapping: provide information about how to sample a continuous space into a sampled space (IE: lanczos/132khz), or continuous from sampled. Or continuous to continuous, sampled to sampled.
        - One mapping provides only A to B, not B to A, nor does it provide A to C.. They aren’t invertible.

    - Operator Domains: describe the domain on which operators in the operator graph operate (IE: Picture, Audio, Color, Spatial, Lens metadata)
- Sampled vs. Continuous time
    - As values
- Coordinate systems (context for data)
    - *What to do about RationalTime*
    - *Frame sequences*
    - *Labels and descriptions of known coordinate systems*
- Time Algebra
    - *Implemented*
    - *Consistent, robust mathematical operations and definitions*
- Units/Metrics to relate sampled data to coordinate systems
    - RationalTime provides a valid metric.
        - Positivity: the distance between two points is always positive
        - Symmetry: the distance between a and b is the same as between b and a
        - Triangle Inequality: d(x,z) <= d(x,y) + d(y, z)

Data Model

- “Compiler” or isomorphic/homophonic different source same product, “AST” for editorial
    - *Adapting, translating structures between different configurations*
- Representation of Coordinate Systems and their relations
- Representation of Sampled Data
- Sampling and Reconstruction Kernel (translation between continuous and discrete domains)
    - Cyclical sampling functions
- Data containers that can specify what metric, unit or coordinate system they are seated in
- Container, Encoding, etc. for representing domains and associating data with those domains
- Temporal transformations can be specified and evaluated within a single document, but they cannot incorporate externally referenced data. So OTIO can evaluate through an embedded curve, but cannot evaluate through a referenced curve to a deeper point in a temporal transform hierarchy.
- At the point of evaluation, if the data is embedded in the file, we can apply a reconstruction filter on embedded data, eg., 128 samples can be convolved into 1 to correspond to an “average” value for a frame. We can’t reach through to a referenced file though, and can only tell “use this filter on this range of samples in your external file”

Algorithms

- Editing Operations
    - OTIO doesn’t implement editing operations

        - Operations such as a split are represented as a hierarchy of coordinate transformations and windows
        - Operations such as a split are performed trivially by flattening a subgraph
    - An editing library can be written by leveraging our composition operators and flattening
- Transformation to other formats is lossy, we do not guarantee that transforming an OTIO to AAF and back will result in the same object hierarchy, but will evaluate to the same composed result.

Document Model

- The hierarchy of the graph is the temporal transform hierarchy, because the operator graph is implicitly described by the hierarchy of objects
- Serialize and deserialize data containers
- “Per frame metadata” => sampled data linked or in reference to media -> in reference to a coordinate system that is also shared by another entity in OTIO
    - Sidecar vs internal data
    - *Lens metadata discussions happening inside the ASC, SMPTE, etc., Cooke, LMF...*

References

- AAF Edit Protocol: https://www.amwa.tv/aaf
- FCP7 XML Description: https://developer.apple.com/library/archive/documentation/AppleApplications/Reference/FinalCutPro_XML/Index/index_of_book.html
- Shake TimeX node https://www.manualsdir.com/manuals/547483/apple-shake-4.html?page=123
- Nuke TimeWarp node https://learn.foundry.com/nuke/content/reference_guide/time_nodes/timewarp.html

- Porter, Thomas; Duff, Tom (July 1984). “Compositing Digital Images”. *SIGGRAPH Computer Graphics*. New York City, New York: ACM Press. **18** (3): 253-259. Doi: 10.1145/800031.808606. ISBN 9780897911382.
- Roberts, Lawrence. Machine Perception of Three-Dimensional Solids, thesis, January 1963
- Allen, James F. (26 November 1983). "Maintaining knowledge about temporal intervals". Communications of the ACM. 26 (11): 832–843. CiteSeerX 10.1.1.472.5244. doi:10.1145/182.358434. ISSN 0001-0782
- Extending allen’s interval algebra to include points and branching time:
    (a) https://www.researchgate.net/profile/Alfred-Reich/publication/220810644_Intervals_Points_and_Branching_Time/links/0912f51299843e1109000000/Intervals-Points-and-Branching-Time.pdf

    (b) https://github.com/alreich/constraint-algebras
- Communicating sequential processes -- algebra to describe left branching/right branching processes + will be interesting when dealing with items of unknown duration until playback time (a game trigger that requires the player doing something, the amount of time it takes for a boat on a theme park ride to travel):
    (a) http://www.usingcsp.com
    (b) https://en.wikipedia.org/wiki/Discrete-event_simulation
    (c) https://en.wikipedia.org/wiki/SystemC
- http://opentimeline.io
- https://www.cs.cmu.edu/~./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf "A logical calculus…"
- Digital Nonlinear Editing, Ohanian

Existing Documents

- Our notes on the reviews from paper 1 https://docs.google.com/document/d/1rhiDWkh7P1H_6tkJT1A4VrPoaFEU-EHm3p8apT4hnso/edit#
    - In this document on the "While the topic is of interest for the siggraph audience and opens"
- "More Thoughts On Time" https://docs.google.com/document/d/1d1JQd2BDfTDe7sA2B4OuAXM90JT7UfXBLmV8P72RNHU/edit#heading=h.v68cyvtaet09
- Overleaf -> Paper v2 https://www.overleaf.com/project/5ff64a62db38230420acd4a8
- Keynote Document/Presentation
- V1 Paper https://docs.google.com/document/d/1uHdhFdiNhc5oB_6E8-RBEjA7M2uFMMTj32SX4jh9l6A/edit#heading=h.m75p8qik28zj
- This document

Sampled Data in the Document Model:

1. Don't support it at all
2. Have a "sampled data reference" that points at an external file that contains the sampled data and an address inside that file for accessing it (analogous to a media reference)
3. Contain it in OTIO
    a. (alembic style constraints -- sampled data that is linearly interpolated between samples) <- linear tangent curve (TCurve terminology), only basic extrapolation support (ABC/USD have this constraint as well)
    b. Integrate AnimX support for dealing with sampled data and non trivial evaluation
    c. Come up with our own schema and structure, even for non-trivial evaluation

4. Integrate with a format that supports it in some form (IE embed in USD or ABC, etc)

- Introduce SampledData (available range, domain, inline data (3a))
- Introduce a list of SampleData as a valid value for a Media_Reference

---- start with an animated effect (e.g. a traveling crop region) with a trivial number of key frames to support inline SampleData

- subclass SampledDataReference from SampledData

---- mocap data obviously suggests reference data

- MediaReference becomes subclass of SampledData

Need to reevaluate and discuss?
TimeIndexedDataContainer (available range, [domains])
  +---- *Constant Data*
  +---- *SampledDataContainer [e.g. animated box for an effect]*
  +---- *EvaluatedDataContainer [maya style time curves]*
  +---- *SampledDataReference [e.g. camera temperature samples, USD file]*
  +---- *MediaReference [audio, video]*

ImageSequence
ChunkContenatonotron

available_range expresses what's in a the container, doesn't manipulate

Currently a clip is one to one with a reference

```
TCS {
        Origin
        Metricscale
        name
}

TCSHier {
        Parent
}

SamplingKernel {
        eval(in time) -> reference index
}

MediaReference {
        Url

        TCS available_range {
                Offset from zero 3000
                Extent of 100
                kernel(this) -> continuous implicit
                kernel(continuous implicit) -> this
        }

        Vector other coordinate systems <
                // “additional coordinate systems encoded in this media”
                TCS capture {
                        Offset from zero 16:14:0
                        Extent 16:14:05. 167
                        kernel(this) -> continuous implicit
                        kernel(continuous implicit) -> this
                }
        >
}

ReferenceSequence
        kernel(reference_stack) -> continuous implicit
        kernel(continuous implicit) -> reference_stack
        List of references

Clip
        .set_primary_reference()
        .primary_reference()
        .reference_count()
        .each_reference()[0]

        kernel(reference_stack) -> continuous implicit
        kernel(continuous implicit) -> reference_stack

        .references_stack : list of ReferenceSequence
                .references_tracks
                        references


MediaReference
```

MissingReference
GeneratorReference
ExternalReference
ChunkedMediaReference
ImageSequenceReference

```
Clip {

    TCS local {
        Offset from zero 0
        Extent of 100
        kernel
    }

    TCSMap { local to media, invertible yes/no }
    TCSMap { media to local }
    TCSMap { media to capture }
    TCSMap { capture to media }

    vector<MediaReference*> homogeneous_media_reference() const;
    vector<MediaReference*> heterogeneous_media_reference() const;

    { TCS, TimeRange } available_range(ErrorStatus* error_status) const;
}
```

Problem statement

1.
Compose in a timeline at 30fps
    Track
        Clip
            MediaReference: Video footage @30fps

Print to a reel @24fps with a continuous magnetic print on the side of the reel

Demonstrate going from reel feet back to indices in the original digitization
From reel feet back to media time in the footage

We need to a introduce Sampler, which is the thing that is responsible for going through the composition and resolving all the TCS sampling/reconstruction to answer the questions

Domains:
- Input Video footage 30fps
- Editorial space
- Output 24fps film reel
- Output magnetic analog print on side of the film reel

Sample mappings:
- Input 30 fps -> Editorial
    - Sample/Reconstruction (conform)
- Editorial -> 24fps film
    - Sample/Reconstruction (conform)

- Input n fps and record a mapping of n to continuously metriced editorial space
- A mapping of the continuous metric to output domains

Domain (discrete)
- Mapping from continuous time to the output domain
- Given an output frame index,
    - _-----___ filter window for time (box kernel)
    - Signal kernel = filterWindow(frame index)
- Reconstruction, given a continuous time, what’s the index

Domain (continuous)
    - _-----___ filter window for time (sinc kernel)
- Reconstruction is a convolution

OTIO Doc:
- Sample Mappings:
    - Analog magnetic audio (out)
        - Named Kernel “sinc for lanczos”
        - Metric is seconds
        - rate 132 khz (relationship of seconds to frame number)
    - Film strip (out):
        - Named Kernel “copy image”
        - rate 24 hz
    - M4V Video (in)
        - Named kernel “copy image”

        - rate 30hz
    - M4V Audio (in)
        - Named kernel “copy frames”
        - Rate 48 khz
- Coordinate Systems:
    - TCS Editorial
        - Metric: seconds/continuous
    - TCS Media Intrinsic
        - Metric: SMPTE/discrete
    - TCS Base
        - Metric: seconds/continuous
- Timeline:
    - Operator Domains: Implicit
    - Stack:
        - Operator Domains: Implicit
        - TCS Base
            - Origin: 0
        - Tracks:
            - Operator Domains: Implicit
            - TCS Editorial
                - Origin: (1:00:00:00.0)
            - Clip:
                - Media Reference mymov.mov
                    - Operator domains: Audio
                    - TCS Media Intrinsic
                    - Input Domains
                        - M4V Audio
                - Media Reference mymov.mov
                    - Operator domains: Picture
                    - TCS Media Intrinsic
                    - Input Domains
                        - M4V Picture
            - Output domains:
                - Analog magnetic audio
                    - TCS Media Intrinsic
                - Film Strip
                    - TCS Media Intrinsic

Hey timeline, I want to sample at 24fps, how many frames are you, what range do you span?

For each frame

- **How can I reconstruct the audio corresponding to the duration of frame?**
    - (media reference, sample range, reconstruction filter/kernel(?)) : one per

intersecting reference

Math Primitives

- TimePoint
- TimeInterval
    - ContinuousTimeInterval
    - DiscreteTimeInterval
    - DiscontinuousTimeMapping
    - Offset
    - Scale
    - Union, Intersect, etc

transform(timeline.origin_and_duration(), timeline_space, 24fps_output_space) => target frame indices, and their span

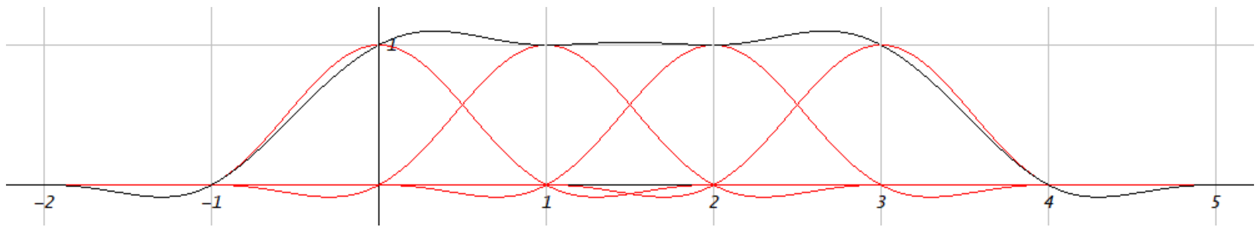
What media given interval in output coordinate system
my_media_reference_reader(temporal_raycast(my_timeline, film_strip_sample_mapping) . for_domain(picture)) ⇒ operator to evaluate

How can I reconstruct the audio to print beside film frame 97? (I will handle compositing of audio outside of OpenTimelineIO of course)

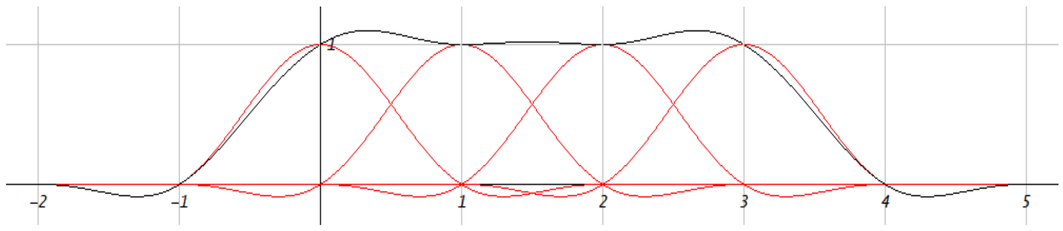
1. Audio Refs = Timeline..tracks.clips.with(audio, 97)
2. For each audio ref
    a. give me an operator that reconstructs an audio signal at any possible time in the interval [97, 98)
        i. This operator chains temporal transformations through the TCS hierarchy to generate an index into the audio clip, and has a reconstruction kernel at the end so that the appropriate set of input frames can be convolved to reconstruct the signal in the output frames

https://en.wikipedia.org/wiki/Lanczos_resampling

24hz



30 hz



Projection:

Film strip:

```
--------+---------------------------------+-------------------
Shutter open                             shutter close

        |                                |   24fps
            Resample from  source domain
      |                                |                             | 30fps
```

Frame index in output domain -> Frame index in editorial domain
  ==== 2:3 pull down/up

Currently: you configure a project in the output format
  Input data has to be conformed on ingest
    Source media is at arbitrary framerate, but by the time it makes it to the bin
    It's already in the output format

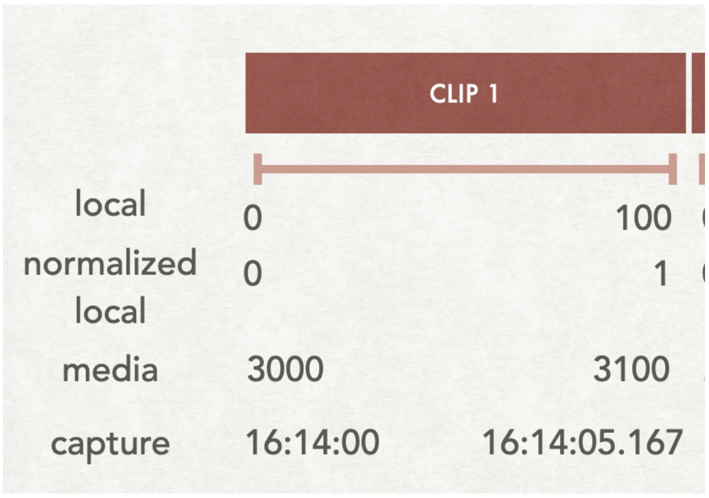
2.
Compose in a timeline at 30fps

```
        Track 1
                Clip
                        MediaReferenceStack
                                Video footage @30fps
                                Digitized to mp4 files
                                External data reference: focal length @100hz
                                Diagetic audio at 48khz

        Track 2
                Clip
                        Media ReferenceStack
                                MediaRefenceSequence
                                        Video footage @30fps
                                Voiceover at 44.1khz
```

Print to a reel @24fps with a continuous magnetic print on the side of the reel

Demonstrate going from reel feet back to indices in the original digitization
From reel feet back to media time in the footage and audio



Questions for the ASC/Autodesk proposal:

- Per frame metadata: “is this out of scope for your application”
    - Our original vision was that per-frame metadata be out of scope for OTIO, and that such data be represented in a USD, ABC, etc. sidecar file.
    - However, as we develop the temporal coordinate system support in OTIO, it is clear that this kind of metadata needs to be evaluated within the context of the temporal coordinate systems that OTIO is encoding, (because the metadata may take one time as an input, and generate another time as an output, which is then used to further evaluate a time to sample at.) That means that in order to answer some questions that OTIO already answers, such as “what clips intersect this time?”, the time warp must be evaluated, which means that the data must be interpretable within OTIO without resort to a black box evaluator, such as a program that can evaluate a USD file, and perform USD queries on behalf of OTIO
    - To put it another way, we’re building a consistent mathematical model for OTIO that could support per time sample metadata, but our original position was to specifically omit that support from the data model and therefore the document model.
    - As we see users want to associate this kind of data with OTIO objects, it is clear that the data model needs to include support for this. For example, if an OTIO clip has a time warp on it, that time warp also needs to manipulate the time coordinates of any time varying data the user intends to associate with it.

    - ...We could still support this without extending the document model, for example, by providing some kind of temporal context for a third party to integrate, but at that point it seems like an ergonomic issue to not just support that kind of data in the OTIO document model itself.
    - Finally, the marker-based approach is not a great embodiment for this kind of data. It requires introspection of all markers to interpret a single marker (if you put time varying metadata on a marker, for example, that value requires context to be evaluated properly).
    - This same problem applies to the lens metadata, color timing, parameters to effects like wipes and transitions, annotations, etc.
    - Proposal - If the time-varying metadata is encapsulated within a core attribute data type, in addition to the data types we already have (bool, int64, double, string, and now, timecurve), then existing machinery transparently extends to embrace it.
- Autodesk Universal Coordinate System proposal
    - Similar to the previous question, it has become clear that there is spatial information that is relevant to users of OTIO that also needs to be tracked through the graph.
    - The way that autodesk does this is by transforming values from a concrete coordinate system (with a metric like `mm`) to one that in R2, and then theoretically don’t encode the metric
    - If we identify domains, metrics, and metric-transformers, then the Autodesk UCS becomes a metric space. The relationship between that metric space and any other could be at Autodesk’s discretion, trivial; namely that the UCS maps to any other metric via a scalar identity mapping, and the original metric could in that case be named as metadata, if not actually recorded using our new proposed schema extensions describing domains and metrics. Possibly, the UCS is the default unnamed metric domain, in the case that an OTIO file is encountered without such specified.

```
typedef {
   float t;    // seconds
} TimePoint;

typedef {
   TimePoint start;
   float d;    // seconds
} ContinuousTimeInterval;

typedef {
   TimePoint start;
   float rate; // cycles per second
   int repetitions;
} DiscreteTimeInterval;

typedef operator ()(TimePoint t) -> TimePoint {
} TimeMapping;

typedef {

   ContinuousTimeInterval range;

   typedef operator ()(TimePoint t) -> TimePoint {
   } TimeMapping;

} TcurveLikeMedia;

typedef {

   ContinuousTimeInterval range;
   DiscreteTimeInterval sampled_ranged;

} MovieLikeMedia;

typedef {

   ContinuousTimeInterval range;
   ordered_list<TimePoint, Value> samples;

} SensorLikeMedia<Value>;

typedef {

   TimePoint origin;
   scalar basis;

} TimeCoordinateSystem;

typedef {

   TimeCoordinateSystem tcs;
   ContinuousTimeInterval range; // bounding range relative to origin
   // label frames
   float first_frame;
   float rate;             // how is continuous represented?

} Domain;


/*
   given:
       my_timeline
           picture_track
               picture_media
           audio_track
               audio_media

       audio_output_domain

   we need to get a ContinuousTimeInterval
       in the TCS of my_timeline
           encompassing picture_media --> Itvl1


   given ContinuousTimeInterval in the TCS of my_timeline (Itvl1)
       return a corresponding DiscreteTimeInterval in the TCS of audio_media (Intvl2)

   given DiscreteTimeInterval in the TCS of audio_media (Intvl2)
   and ontinuousTimeInterval in the TCS of my_timeline (Itvl1)
       return a corresponding DiscreteTimeInterval in the TCS of audio_output_domain
       and a sampling kernel for audio_media func(Intvl1)
       and a reconstruction filter for audio_output_domain func(Intvl2) -> Filter




   we need to get a ContinuousTimeInterval
       in the output TCS of my_timeline
           encompassing picture_media --> Itvl1

   we need to get a DiscreteTimeInterval
       in the TCS of audio_output_domain --> AOD_Interval
   foreach(t : AOD_Interval)

       SampleIndex_Audio(t,
             tcs_AOD__to__tcs_timeline, tcs_timeline__to__tcs_audio_media,
             kernel size in seconds, kernel function)
                 ----> the discrete indices and weights for audio_media (i, w)

       Reconstruct_Audio(t, (audio_media, i, w)) -> value output for t

   foreach(t : AOD_Interval)

       SampleIndex_30(t)
                 ----> the discrete indices and weights for audio_media (i, w)

       Reconstruct_24(t) -> value output for t



   SampleIndex is in the realm of tcs and can therefore be provided by OTIO
       we can provide certain kernel functions / sinc, box, dirac
       user can also provide???

   Reconstruct is codec dependent and is provided by the application because it
requires
        knowledge of the sampling system, and the reconstruction system



*/
```