Editing Movies in AV Foundation

Session 506

Tim Monroe  AV Foundation Engineering
What You Will Learn
Editing movies in AV Foundation
What You Will Learn

Editing movies in AV Foundation

AV Foundation provides new classes for editing **QuickTime movie files**:
What You Will Learn

Editing movies in AV Foundation

AV Foundation provides new classes for editing QuickTime movie files:

• Perform range-based editing on movies and tracks
What You Will Learn

Editing movies in AV Foundation

AV Foundation provides new classes for editing QuickTime movie files:

• Perform range-based editing on movies and tracks
• Add and remove tracks
What You Will Learn

Editing movies in AV Foundation

AV Foundation provides new classes for editing QuickTime movie files:

- Perform range-based editing on movies and tracks
- Add and remove tracks
- Set track associations between tracks
What You Will Learn

Editing movies in AV Foundation

AV Foundation provides new classes for editing QuickTime movie files:

- Perform range-based editing on movies and tracks
- Add and remove tracks
- Set track associations between tracks
- Add or modify movie and track metadata
What You Will Learn

Editing movies in AV Foundation

AV Foundation provides new classes for editing *QuickTime movie files*:

- Perform range-based editing on movies and tracks
- Add and remove tracks
- Set track associations between tracks
- Add or modify movie and track metadata
- Create movie files and URL sample reference movie files
Session Outline
Editing movies in AV Foundation
Session Outline

Editing movies in AV Foundation

Introduce new classes for creating and editing QuickTime movie files
Session Outline

Editing movies in AV Foundation

Introduce new classes for creating and editing QuickTime movie files
Survey the new movie- and track-editing methods
Session Outline

Editing movies in AV Foundation

Introduce new classes for creating and editing QuickTime movie files
Survey the new movie- and track-editing methods
Describe a personal project that will benefit from these new capabilities
AV Foundation Editing Classes
AVAsset and Its Editing Subclasses

AVAsset

AVComposition

AVMutableComposition
AVAsset and Its Subclasses

AVAsset

AVComposition

AVMutableComposition

AVMovie

AVMutableMovie
AVAssetTrack and Its Subclasses

AVAssetTrack

AVCompositionTrack

AVMutableCompositionTrack
AVAssetTrack and Its Subclasses

AVAssetTrack

AVCompositionTrack

AVMutableCompositionTrack

AVMovieTrack

AVMutableMovieTrack
New Movie Editing Classes
New Movie Editing Classes

AVMovie and AVMutableMovie
New Movie Editing Classes

AVMovie and AVMutableMovie
AVMovieTrack and AVMutableMovieTrack
New Movie Editing Classes

AVMovie and AVMutableMovie
AVMovieTrack and AVMutableMovieTrack
AVMediaDataStorage
QuickTime Movie Files
QuickTime Movie Files

AVMovie represents the data in a file that conforms to the QuickTime movie file format or to one of the related ISO base media file formats (such as MPEG-4)
QuickTime Movie Files

AVMovie represents the data in a file that conforms to the QuickTime movie file format or to one of the related ISO base media file formats (such as MPEG-4).

These formats impose a strict separation between the sample data and the information that organizes that sample data into tracks and movies.
QuickTime Movie Files
A sequence of boxes

<table>
<thead>
<tr>
<th>File Type</th>
<th>Movie</th>
<th>Sample Data</th>
</tr>
</thead>
</table>
QuickTime Movie Files

A sequence of boxes

| File Type | Sample Data | Movie |
QuickTime Movie Files

A sequence of boxes

<table>
<thead>
<tr>
<th>File Type</th>
<th>Sample Data</th>
<th>Movie</th>
<th>Sample Data</th>
</tr>
</thead>
</table>

Movie Box
Stores global settings, metadata, and track information

Global Settings  Movie Metadata  Track Boxes
Movie Box

Stores global settings, metadata, and track information

<table>
<thead>
<tr>
<th>Global Settings</th>
<th>Movie Metadata</th>
<th>Track Boxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Track count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Duration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Creation date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Preferred rate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Movie Box
Stores global settings, metadata, and track information

<table>
<thead>
<tr>
<th>Global Settings</th>
<th>Movie Metadata</th>
<th>Track Boxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Track count</td>
<td>• Copyright statement</td>
<td></td>
</tr>
<tr>
<td>• Duration</td>
<td>• Author</td>
<td></td>
</tr>
<tr>
<td>• Creation date</td>
<td>• Title</td>
<td></td>
</tr>
<tr>
<td>• Preferred rate</td>
<td>• Custom metadata</td>
<td></td>
</tr>
</tbody>
</table>
# Movie Box

Stores global settings, metadata, and track information

<table>
<thead>
<tr>
<th>Global Settings</th>
<th>Movie Metadata</th>
<th>Track Boxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Track count</td>
<td>• Copyright statement</td>
<td>• Track type</td>
</tr>
<tr>
<td>• Duration</td>
<td>• Author</td>
<td>• Sample data location</td>
</tr>
<tr>
<td>• Creation date</td>
<td>• Title</td>
<td>• Track metadata</td>
</tr>
<tr>
<td>• Preferred rate</td>
<td>• Custom metadata</td>
<td>• Track associations</td>
</tr>
</tbody>
</table>
Track Box
Organizes the sample data into tracks

Sample References

Track Box → Sample Data

Movie Box
QuickTime Movie Files

Track box can point to external sample data

Movie Box

Track Box

Sample References

Sample Data

Sample References

Sample Data
QuickTime Movie Files

Track box can point to only external sample data

Movie Box

Sample References

Sample Data
QuickTime Movie Files

Track box can point to only external sample data

Movie Box

Sample Data

Sample References

Sample Reference Movie File

Track Box
Sample Reference Movie Files
Sample Reference Movie Files

Provide a powerful workflow tool
Sample Reference Movie Files

Provide a powerful workflow tool
But sample reference movies are inherently fragile
Sample Reference Movie Files

Provide a powerful workflow tool
But sample reference movies are inherently fragile
To help reduce that fragility, use relative URLs
Sample Reference Movie Files

Provide a powerful workflow tool

But sample reference movies are inherently fragile

To help reduce that fragility, use relative URLs

When it’s time to deliver content, export it using AVAssetExportSession
Movie Editing API
AVMovie
Inspecting QuickTime movie files
AVMovie

Inspecting QuickTime movie files

AVMovie provides inspection and header-writing methods:
AVMovie
Inspecting QuickTime movie files

AVMovie provides inspection and header-writing methods:
• Get a list of tracks in the movie
AVMovie

Inspecting QuickTime movie files

AVMovie provides inspection and header-writing methods:

• Get a list of tracks in the movie
• Retrieve the movie header from an existing file
AVMovie provides inspection and header-writing methods:
- Get a list of tracks in the movie
- Retrieve the movie header from an existing file
- Write a movie header into a new file
AVMovie

Initializing AVMovie objects
let movie = AVMovie(URL: inputURL, options: nil)
AVMovie

Initializing AVMovie objects

let movie = AVMovie(URL: inputURL, options: nil)

let movie = AVMovie(data: inputData, options: nil)
Creating a sample reference movie file

```swift
let movie = AVMovie(URL: inputURL, options: nil)
let options = AVMovieWritingOptions.TruncateDestinationToMovieHeaderOnly

try movie.writeMovieHeaderToURL(outputURL,
    fileType: AVFileTypeQuickTimeMovie,
    options: options)
```
AVMovie
Movie header writing options

let movie = AVMovie(URL: inputURL, options: nil)
let options = AVMovieWritingOptions.TruncateDestinationToMovieHeaderOnly

try movie.writeMovieHeaderToURL(outputURL,
    fileType: AVFileTypeQuickTimeMovie,
    options: options)
let movie = AVMovie(URL: inputURL, options: nil)
let options = AVMovieWritingOptions.AddMovieHeaderToDestination

try movie.writeMovieHeaderToURL(outputURL,
    fileType: AVFileTypeQuickTimeMovie,
    options: options)
AVMutableMovie
Modifying QuickTime movie files
AVMutableMovie
Modifying QuickTime movie files

AVMutableMovie adds editing methods:
AVMutableMovie

Modifying QuickTime movie files

AVMutableMovie adds editing methods:
• Perform range-based movie editing
AVMutableMovie

Modifying QuickTime movie files

AVMutableMovie adds editing methods:

- Perform range-based movie editing
- Add and remove tracks
AVMutableMovie

Modifying QuickTime movie files

AVMutableMovie adds editing methods:

• Perform range-based movie editing
• Add and remove tracks
• Add or modify movie metadata
AVMutableMovie

Initializing AVMutableMovie objects
AVMutableMovie

Initializing AVMutableMovie objects

let movie = try AVMutableMovie(URL: inputURL, options: nil)
AVMutableMovie

Initializing AVMutableMovie objects

let movie = try AVMutableMovie(URL: inputURL, options: nil)

let movie = AVMutableMovie()
AVMutableMovie

Segment editing

func removeTimeRange(timeRange: CMTimeRange)

func insertEmptyTimeRange(timeRange: CMTimeRange)

func scaleTimeRange(timeRange: CMTimeRange, toDuration: CMTime)

func insertTimeRange(timeRange: CMTimeRange, 
ofAsset: AVAsset, 
atTime: CMTime, 
copySampleData: Bool) throws
AVMutableMovie

Segment editing

func removeTimeRange(timeRange: CMTimeRange)

func insertEmptyTimeRange(timeRange: CMTimeRange)

func scaleTimeRange(timeRange: CMTimeRange, toDuration: CMTime)

func insertTimeRange(timeRange: CMTimeRange, ofAsset: AVAsset, atTime: CMTime, copySampleData: Bool) throws
AVMediaDataStorage

Setting the container for movie’s new sample data
AVMediaDataStorage

Setting the container for movie’s new sample data

```
movie.defaultMediaDataStorage = AVMediaDataStorage(URL: movURL, options: nil)
```
AVMutableMovie

Creating and removing tracks

```swift
func addMutableTrackWithMediaType(
    mediaType: String,
    copySettingsFromTrack: AVAssetTrack?,
    options: [String : AnyObject]?) -> AVMutableMovieTrack

func removeTrack(track: AVMovieTrack)
```
AVMutableMovie

Use case: updating an existing movie file

let movie = try AVMutableMovie(URL: url, options: nil)

perform some edits on the movie

let options = AVMovieWritingOptions.AddMovieHeaderToDestination
try movie.writeMovieHeaderToURL(url,
    fileType: AVFileTypeQuickTimeMovie,
    options: options)
Track Editing API
AVMutableMovieTrack
Modifying tracks in QuickTime movie files
AVMutableMovieTrack

Modifying tracks in QuickTime movie files

AVMutableMovieTrack adds editing methods:
AVMutableMovieTrack

Modifying tracks in QuickTime movie files

AVMutableMovieTrack adds editing methods:

• Perform range-based track editing
AVMutableMovieTrack
Modifying tracks in QuickTime movie files

AVMutableMovieTrack adds editing methods:
• Perform range-based track editing
• Set track associations between tracks
AVMutableMovieTrack adds editing methods:

• Perform range-based track editing
• Set track associations between tracks
• Add or modify track metadata
AVMutableMovieTrack

Segment editing

func removeTimeRange(timeRange: CMTimeRange)

func insertEmptyTimeRange(timeRange: CMTimeRange)

func scaleTimeRange(timeRange: CMTimeRange, toDuration: CMTime)

func insertTimeRange(timeRange: CMTimeRange, 
ofTrack: AVAssetTrack, 
atTime: CMTime, 
copySampleData: Bool) throws
AVMutableMovieTrack

Segment editing

func removeTimeRange(timeRange: CMTimeRange)

func insertEmptyTimeRange(timeRange: CMTimeRange)

func scaleTimeRange(timeRange: CMTimeRange, toDuration: CMTime)

func insertTimeRange(timeRange: CMTimeRange, 
ofTrack: AVAssetTrack, 
atTime: CMTimes, 
copySampleData: Bool) throws
AVMediaDataStorage

Setting the container for a track’s new sample data
AVMediaDataStorage

Setting the container for a track's new sample data

```
track.mediaDataStorage = AVMediaDataStorage(URL: movURL, options: nil)
```
AVMutableMovieTrack
Use case: silencing a track segment

```swift
if let track = movie.tracksWithMediaType(AVMediaTypeAudio).first {
    let range = CMTTimeRangeFromTimeToTime(start, end)
    track.removeTimeRange(range)
    track.insertEmptyTimeRange(range)
}
```
func addTrackAssociationToTrack(movieTrack: AVMovieTrack, type: String)

func removeTrackAssociationToTrack(movieTrack: AVMovieTrack, type: String)
AVMutableMovieTrack

Use case: using relative URLs to reference data
AVMutableMovieTrack

Use case: using relative URLs to reference data

```swift
let url = NSURL(fileURLWithPath: "/Users/monroe/tristo_boston/movies")

for track in movie.tracks {
    track.sampleReferenceBaseURL = url
}
```
A Study in Scarlet (and Gray)
TRISTO Oakland

Tim’s Radical Inline Skate Tour of Oakland

Continued from 1985 to 2005
TRISTO Oakland
Tim’s Radical Inline Skate Tour of Oakland

Continued from 1985 to 2005
800+ miles of roadways
TRISTO Oakland
Tim’s Radical Inline Skate Tour of Oakland

Continued from 1985 to 2005
800+ miles of roadways
No location data
TRISTO Oakland

Tim’s Radical Inline Skate Tour of Oakland

Continued from 1985 to 2005
800+ miles of roadways
No location data
Almost no video
TRISTO Boston
Tim’s Radical Inline Skate Tour of Boston
TRISTO Boston
Tim’s Radical Inline Skate Tour of Boston

Began in 2011
TRISTO Boston

Tim’s Radical Inline Skate Tour of Boston

Began in 2011

Estimated 5 year project
TRISTO Boston
Tim’s Radical Inline Skate Tour of Boston

Began in 2011
Estimated 5 year project
800+ miles of roadways
TRISTO Boston
Tim’s Radical Inline Skate Tour of Boston

Began in 2011
Estimated 5 year project
800+ miles of roadways
Completed May 2013
TRISTO Boston
Tim’s Radical Inline Skate Tour of Boston

Began in 2011
Estimated 5 year project
800+ miles of roadways
Completed May 2013
Audio-video data: 490 MPEG-4 files covering about 200 sorties (1.5 terabytes)
TRISTO Boston

Tim’s Radical Inline Skate Tour of Boston

Began in 2011
Estimated 5 year project
800+ miles of roadways
Completed May 2013
Audio-video data: 490 MPEG-4 files covering about 200 sorties (1.5 terabytes)
Location data: GPS data as .gpx files (150 megabytes)
Some Movies
The Task
Manage 1.5 terabytes of data
The Task

Manage 1.5 terabytes of data

Step 1: Combine each sortie’s MPEG-4 files into one sample reference movie file
The Task

Manage 1.5 terabytes of data

Step 1: Combine each sortie’s MPEG-4 files into one sample reference movie file
Step 2: Add indexing metadata as movie metadata
The Task

Manage 1.5 terabytes of data

Step 1: Combine each sortie's MPEG-4 files into one sample reference movie file
Step 2: Add indexing metadata as movie metadata
Step 3: Add GPS data as a timed metadata track
The Task

Manage 1.5 terabytes of data

Step 1: Combine each sortie’s MPEG-4 files into one sample reference movie file
Step 2: Add indexing metadata as movie metadata
Step 3: Add GPS data as a timed metadata track

Do this all without modifying the original files and minimizing data copying
The Solution: Step 1
Combine camera files into a sample reference movie file

2013_11_28_1.MP4
2013_11_28_2.MP4
2013_11_28.MOV
The Solution: Step 2
Add custom metadata

- 2013_11_28_1.MP4
- 2013_11_28_2.MP4
- 2013_11_28.MOV
The Solution: Step 3
Add a timed metadata track for location data

2013_11_28_1.MP4
2013_11_28_2.MP4
2013_11_28.MOV
Step 1
Combine camera files into a sample reference movie file

let movie = AVMutableMovie(URL: url1, options: nil)  // *_1.MP4
let asset = AVURLAsset(URL: url2, options: nil)      // *_2.MP4

let range = CMTimeTypeRangeMake(kCMTimeTypeZero, asset.duration)

try movie.insertTimeRange(range,  
ofAsset: asset,  
atTime: movie.duration,  
copySampleData: false)

try movie.writeMovieHeaderToURL(dstURL,  
fileType: AVFileTypeQuickTimeMovie,  
options: AVMovieWritingOptions.AddMovieHeaderToDestination)
Step 1
Combine camera files into a sample reference movie file

```
let movie = AVMutableMovie(URL: url1, options: nil)  // *_1.MP4
let asset = AVURLAsset(URL: url2, options: nil)      // *_2.MP4

let range = CMTimeRangeMake(kCMTimeZero, asset.duration)

try movie.insertTimeRange(range,
    ofAsset: asset,
    atTime: movie.duration,
    copySampleData: false)

try movie.writeMovieHeaderToURL(dstURL,
    fileType: AVFileTypeQuickTimeMovie,
    options: AVMovieWritingOptions.AddMovieHeaderToDestination)
```
Step 1

Combine camera files into a sample reference movie file

```swift
let movie = AVMutableMovie(URL: url1, options: nil) // *_1.MP4
let asset = AVURLAsset(URL: url2, options: nil) // *_2.MP4

let range = CMTimeRangeMake(kCMTimeZero, asset.duration)

try movie.insertTimeRange(range,
ofAsset: asset,
atTime: movie.duration,
copySampleData: false)

try movie.writeMovieHeaderToURL(dstURL,
fileType: AVFileTypeQuickTimeMovie,
options: AVMovieWritingOptions.AddMovieHeaderToDestination)
```
Step 1

Combine camera files into a sample reference movie file

```swift
let movie = AVMutableMovie(URL: url1, options: nil)  // *_1.MP4
let asset = AVURLAsset(URL: url2, options: nil)      // *_2.MP4

let range = CMTimeRangeMake(kCMTimeZero, asset.duration)

try movie.insertTimeRange(range, ofAsset: asset, atTime: movie.duration, copySampleData: false)

try movie.writeMovieHeaderToURL(dstURL, fileType: AVFileTypeQuickTimeMovie, options: AVMovieWritingOptions.AddMovieHeaderToDestination)
```
Some Demos

Alvin: an AV Foundation-Based Linear Indexer
The Solution: Step 2
Add custom metadata

- 2013_11_28_1.MP4
- 2013_11_28_2.MP4
- 2013_11_28.MOV
Step 2
Add custom metadata

```swift
var metadataArray = movie.metadata

var newItem = AVMutableMetadataItem()
newItem.identifier = "mdta/com.example.weather.wind"
newItem.locale = NSLocale.currentLocale()
newItem.value = averageWindSpeedValue
newItem.extraAttributes = nil

metadataArray.append(newItem)
movie.metadata = metadataArray
```
The Solution: Step 3
Add a timed metadata track for location data

2013_11_28_1.MP4

2013_11_28_2.MP4

2013_11_28.MOV
Step 3, Part 1

Create a movie file containing location timed metadata

See “Harnessing Metadata in Audiovisual Media,” WWDC 2014

Sample code: AVCaptureLocation and AVTimedAnnotationWriter
Step 3, Part 2
Add a timed metadata track for location data

```swift
let gpsAsset = AVURLAsset(URL: gpsURL, options: nil)

if let gpsTrack = gpsAsset.tracks.first {
    let newTrack = movie.addMutableTrackWithMedia_type(gpsTrack.mediaType,
                                                        copySettingsFromTrack: gpsTrack,
                                                        options: nil)

    let range = CMTimeRangeMake(kCMTimeZero, gpsAsset.duration)
    try newTrack.insertTimeRange(range,
                                ofTrack: gpsTrack,
                                atTime: kCMTimeZero,
                                copySampleData: true)
}
```
Step 3, Part 2
Add a timed metadata track for location data

```
let gpsAsset = AVURLAsset(URL: gpsURL, options: nil)

if let gpsTrack = gpsAsset.tracks.first {
    let newTrack = movie.addMutableTrackWithMediaType(gpsTrack.mediaType, 
        copySettingsFromTrack: gpsTrack, 
        options: nil)

    let range = CMTimeRangeMake(kCMTimeZero, gpsAsset.duration)
    try newTrack.insertTimeRange(range, 
        ofTrack: gpsTrack, 
        atTime: kCMTimeZero, 
        copySampleData: true)
}
```
Step 3, Part 2

Add a timed metadata track for location data

```
let gpsAsset = AVURLAsset(URL: gpsURL, options: nil)

if let gpsTrack = gpsAsset.tracks.first {
    let newTrack = movie.addMutableTrackWithMediaType(gpsTrack.mediaType,
        copySettingsFromTrack: gpsTrack,
        options: nil)

    let range = CMTimeRangeMake(kCMTimeZero, gpsAsset.duration)
    try newTrack.insertTimeRange(range,
        ofTrack: gpsTrack,
        atTime: kCMTimeZero,
        copySampleData: true)
}
```
Step 3, Part 3
Add a track association

```swift
let vidTrack = movie.tracksWithMediaType(AVMediaTypeVideo).first
let type = AVTrackAssociationTypeMetadataReferent

newTrack.addTrackAssociationToTrack(vidTrack, type)
```
AVMutableMovie

Best practices
AVMutableMovie

Best practices

An AVMovie or AVMutableMovie is an AVAsset, so you can
AVMutableMovie

Best practices

An AVMovie or AVMutableMovie is an AVAsset, so you can

• Play it using an AVPlayerItem
AVMutableMovie

Best practices

An AVMovie or AVMutableMovie is an AVAsset, so you can

• Play it using an AVPlayerItem
• Grab an image using AVAssetImageGenerator
AVMutableMovie

Best practices

An AVMovie or AVMutableMovie is an AVAsset, so you can

• Play it using an AVPlayerItem
• Grab an image using AVAssetImageGenerator
• Export it using AVAssetExportSession
AVMutableMovie

Best practices

An AVMovie or AVMutableMovie is an AVAsset, so you can

• Play it using an AVPlayerItem
• Grab an image using AVAssetImageGenerator
• Export it using AVAssetExportSession

However, to do these operations on a changing AVMutableMovie, make a copy of it:
An AVMovie or AVMutableMovie is an AVAsset, so you can
• Play it using an AVPlayerItem
• Grab an image using AVAssetImageGenerator
• Export it using AVAssetExportSession

However, to do these operations on a changing AVMutableMovie, make a copy of it:

```swift
let playerItem = AVPlayerItem(withAsset: mutableMovie.copy)
```
AVMutableMovie

Best practices
AVMutableMovie

Best practices

When opening assets to insert into an AVMutableMovie,
AVMutableMovie

Best practices

When opening assets to insert into an AVMutableMovie,

• set `AVURLAssetPreferPreciseDurationAndTimingKey` to true
Summary

New editing features provide access to QuickTime movie file format
Allows simplified editing workflows, especially when handling large amounts of data
Sample code: AVMovieEditor
More Information

Documentation and Videos
Documentation
http://developer.apple.com/

Technical Support
Apple Developer Forums
http://developer.apple.com/forums

Developer Technical Support
http://developer.apple.com/support/technical
<table>
<thead>
<tr>
<th>Session Title</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVKit and AV Foundation Lab</td>
<td>Graphics, Games, and Media Lab B</td>
<td>Thursday 11:00AM</td>
</tr>
<tr>
<td>AVKit and AV Foundation Lab</td>
<td>Graphics, Games, and Media Lab B</td>
<td>Friday 1:30PM</td>
</tr>
<tr>
<td>Editing Media with AV Foundation</td>
<td>Session 407</td>
<td>WWDC10</td>
</tr>
<tr>
<td>Harnessing Metadata in Audiovisual Media</td>
<td>Session 505</td>
<td>WWDC14</td>
</tr>
</tbody>
</table>