iMerge: an App to merge two video files

By Shiyu Dong

1. Introduction

This iOS app, called iMerge, allows users to merge two videos that are in either .mp4 or .mov format into one video, and export it to another file (in .mov format). Users can also preview each file as well as the result file, before exporting it to photo library.

2. Requirements

The following requirements are from Marc.
Develop an original application with a user-interface to allow pasting together multiple source audiovisual media files (e.g., QuickTime movies, MPEG-4 files). Add functionality to export the resulting media to a new file.

Requirements of the solution:
(*) Implement in Swift, Objective-C, or C++
(*) Use an OS-native or open-source library to operate on media (e.g., DirectX, AVFoundation)
(*) The user interface should allow preview of each media file as well as the pasted result

Delivery artifacts:
(*) Source code and supporting project build files
(*) [Optional] Binaries that are known to run (specify platform compatibility)
(*) [Optional] Screen recording of the project running (preferably in a QuickTime movie or MPEG-4 file)

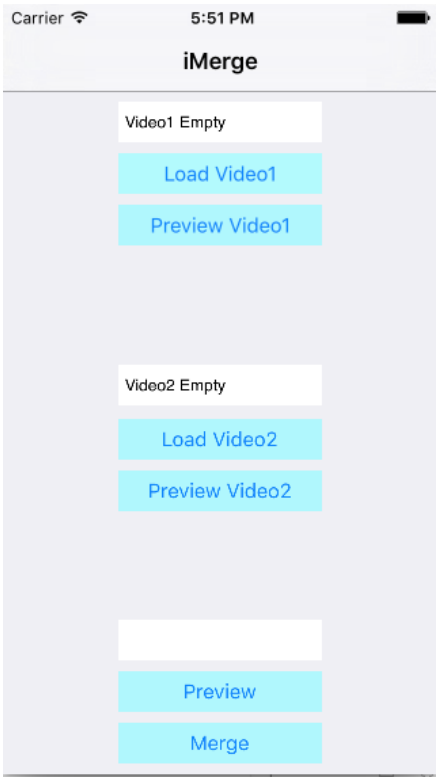
Clarifications: We should be able to select two media files (.mov or .mp4) and your application should accept them and be able to export them into one flattened media file.

3. Design

Since this is my first iOS App and I haven't used Swift before, I made this App very simple in design. I used Main.storyboard for all UI designs, and all the Swift implementations are under ViewController class.

3.1 UI Design

The UI is very simple at this stage. It just has basic functionalities including a title, 6 buttons and 3 text fields. A screen shot of the UI is attached below:



Buttons:

- "Load Video1" (@IBAction func loadVideo1) is used to load the first video. It will redirect user to photo library to select he video that they want to load.
- "Load Video2" (@IBAction func loadVideo2) has the same feature as "Load Video1" for video 2.
- "Preview video1" (@IBAction func previewVideo1) is used for previewing video1 if it's loaded, otherwise it will pop up an alert saying that video1 is not loaded
- "Preview video2" (@IBAction func previewVideo2) has the same feature as "Preview video1" for video 2.
- "Preview" (@IBAction func preview) is used for previewing merged video, before it's exported to photo library, if both videos are loaded. Otherwise it will pop up an alert saying that not both videos are loaded.
- "Merge" (@IBAction func merge) will export the merged photo to photo-library if both videos are loaded. Otherwise it will pop up an alert saying that not both videos are loaded.

Text fields

- The text field above "Load Video1" (@IBOutlet var firstVideoText) is used to show the status of video1. It will show "Video1 empty" before video1 is loaded, or after merge completes.
- The text field above "Load Video2" (@IBOutlet var secondVideoText) works the same as the previous text filed, except for it's used to show the status of video2.

- The text field above “preview” (@IBOutlet var mergeText) is to show merge status. If both videos are loaded it will show “Ready to merge”. If merging is in progress it will show “Merging…”. For all other cases it is empty.

3.2 Components

3.2.1 Interface builders

All interface builders are linked to text fields in UI and will update the live status of loading/merging status

1) firstVideoText: status of first video, will be either “Empty” or “Loaded”
2) secondVideoText: same as above, for second video
3) mergeText: status of merge, will be either “”, “Ready to merge” or “Merging…”

3.2.2 Global variables

I use following global variables to track the status of the App:

1) firstAsset, secondAsset: stores video information in AVAsset?
2) loadingVideo1: true if we are currently loading video1, false otherwise
3) pre_merged: if video has been composed before. If there’s new loading after composition, pre_merged will be set to false
4) exporter: an exporter in AVAssetExportSession that stores the video information and can be used to exporting
5) is_exporting: if we are currently in the middle of exporting (merging)

3.2.3 System API

In system API viewDidLoad we will reset all interface builders to their default values

3.2.4 loadVideo API (loadVideo1, loadVideo2)

This API and will first check whether photo library is available. It will then set global loadingVideo1, and then start media controller if photo library exists, and hence allow user to open a video using presentViewController. Otherwise it will pop up an error message.

3.2.5 Delegate API

Above presetViewController need a delegate API to notify our App which (if any) video has been selected or not. It will do the following things:

1) Update firstAsset/secondAsset based on loadingVideo1 and the actual video asset loaded
2) Set pre_merged to false since the video has been updated
3) Update firstVideoText/secondVideoText to update UI.
4) If both assets are loaded, it will also update mergeText indicating that we are ready to merge.

3.2.6 previewVideo API (previewVideo1, previewVideo2)

If firstAsset/secondAsset exists, it will play the video inside that asset using AVPlayer. Otherwise it will pop up an error message.

3.2.7 preview API

This API will call helper API pre_merge(is_preview: true). True means we are in preview mode and we need to play the video after it's been composed.
The pre_merge() API is the key API for this App. It includes the following main operations:

1) Check if both assets are loaded, if not, pop up error message and exit.
2) Create an AVMutableComposition object, and initialize videoTrack and audioTrack to handle video and audio separately.
3) Merge video and audio tracks for each video, using the time range information of each video.
4) Set the merge properties using AVMutableVideoCompositionInstruction . More specifically, modify current composition by correcting the orientations and resizing if necessary.
5) Calculate the file path and file name for the exported file
6) Updating global variable exporter with information from current composition.
7) If we are in preview mode, play the video using previously generated video composition.
8) Set pre_merged to true, since our exporter has latest video to be exported already

3.2.8 merge API

Given that pre_merge() API defined previously, the behavior of merge is simple, since pre_merge() does all the job of combining the two assets and generating exporter. It will do the following:

1) Check if (pre_merged == false), and call pre_merge(false) if the video hasn't been previewed after last video load (false for not in preview mode, so that we don't need to play the video).
2) If there is a valid exporter and we are not in the middle of merging (is_exporting == false), export the current exporter to file using an asynchronized export completion handler
3) Once export is finished, the completion handler will pop up a message saying either export succeeded or not, and reset all global variables and interface builders to their initial values.

4. Future works

Since this is my first iOS App and my first time writing Swift code, I'm sure many of the code that I've written or design that I made are very naïve. In general this App is just a working solution that satisfies all of the basic requirements. Both the UI and implementation can be greatly improved if I have more knowledge about XCode, Swift and AVFoundation.

5. References

Some great tutorials for learning Swift and iOS programming:
http://www.raywenderlich.com/115253/swift-2-tutorial-a-quick-start
http://www.raywenderlich.com/115279/swift-2-tutorial-part-2-a-simple-ios-app

http://www.raywenderlich.com/115300/swift-2-tutorial-part-3-tuples-protocols-delegates-and-table-views
A great tutorial about video editing using Swift
http://www.raywenderlich.com/94404/play-record-merge-videos-ios-swift
Apple's official documentation about video editing using AVFoundation
https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/03_Editing.html