VK Клипы

Показываем красивое с помощью видеоредактора в VK Клипах на iOS

Дементьев Михаил, VK Клипы

Я IOS-разработчик в команде VK Клипов

Работаю с видео уже больше трех лет

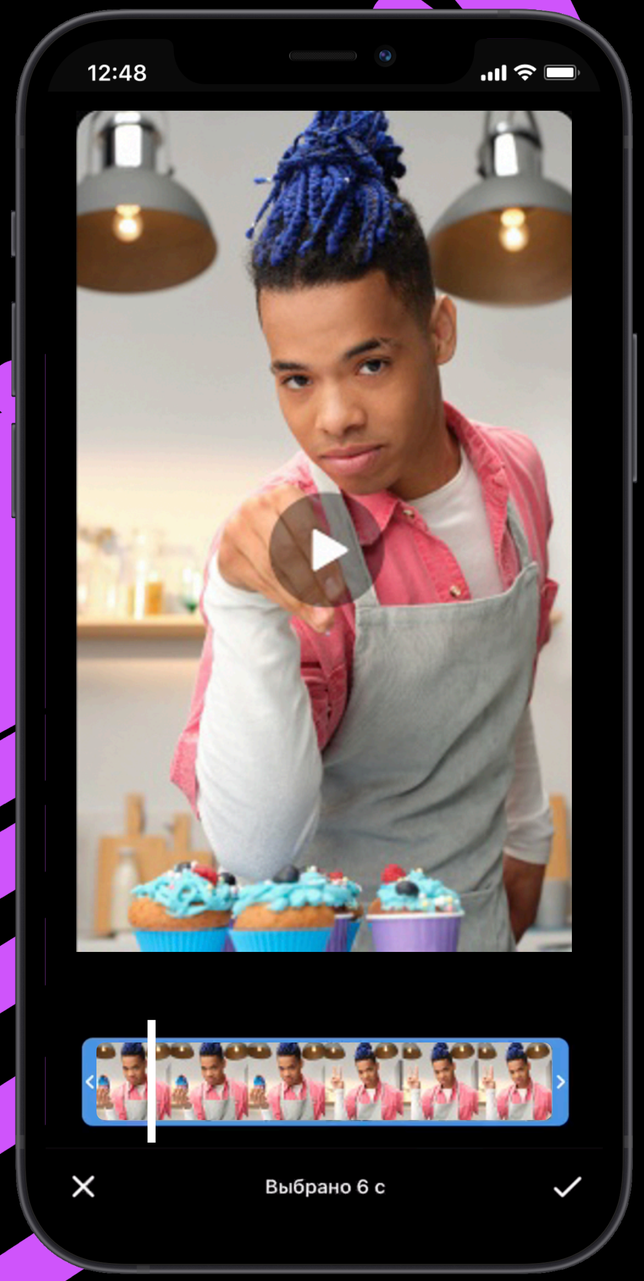
VK Клипы

VK Клипы — номер один среди локальных сервисов коротких видео

836 млн
просмотров в сутки

25 млн
зрителей каждый день

1,1 млрд рекорд
просмотров в сутки

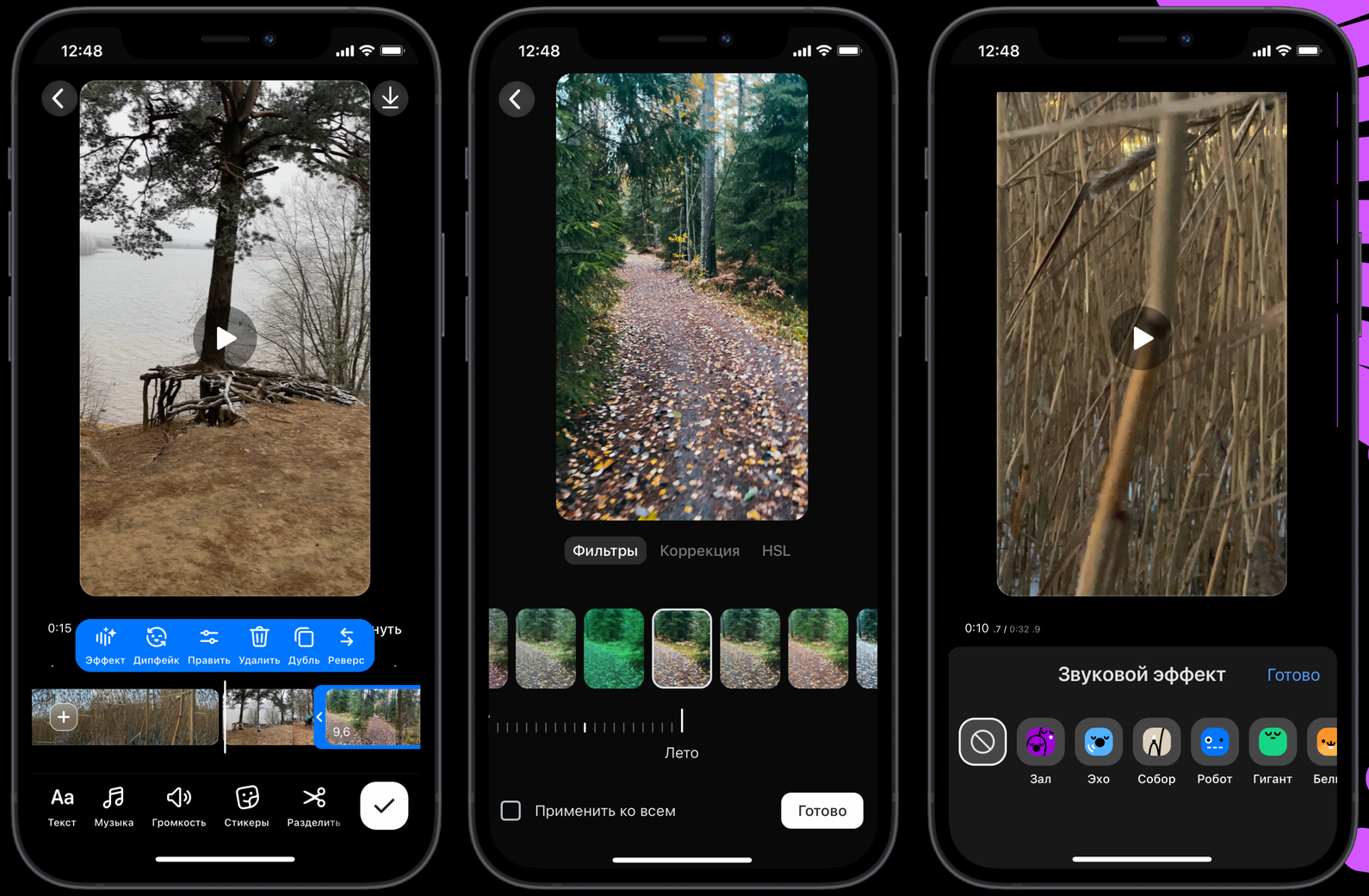


Самый первый редактор

Обрезка длительности видео при первом открытии

Вторая итерация редактора

Финальный расширенный редактор



AVPlayer

В режиме реального времени отредактировать видео перед проигрываем.

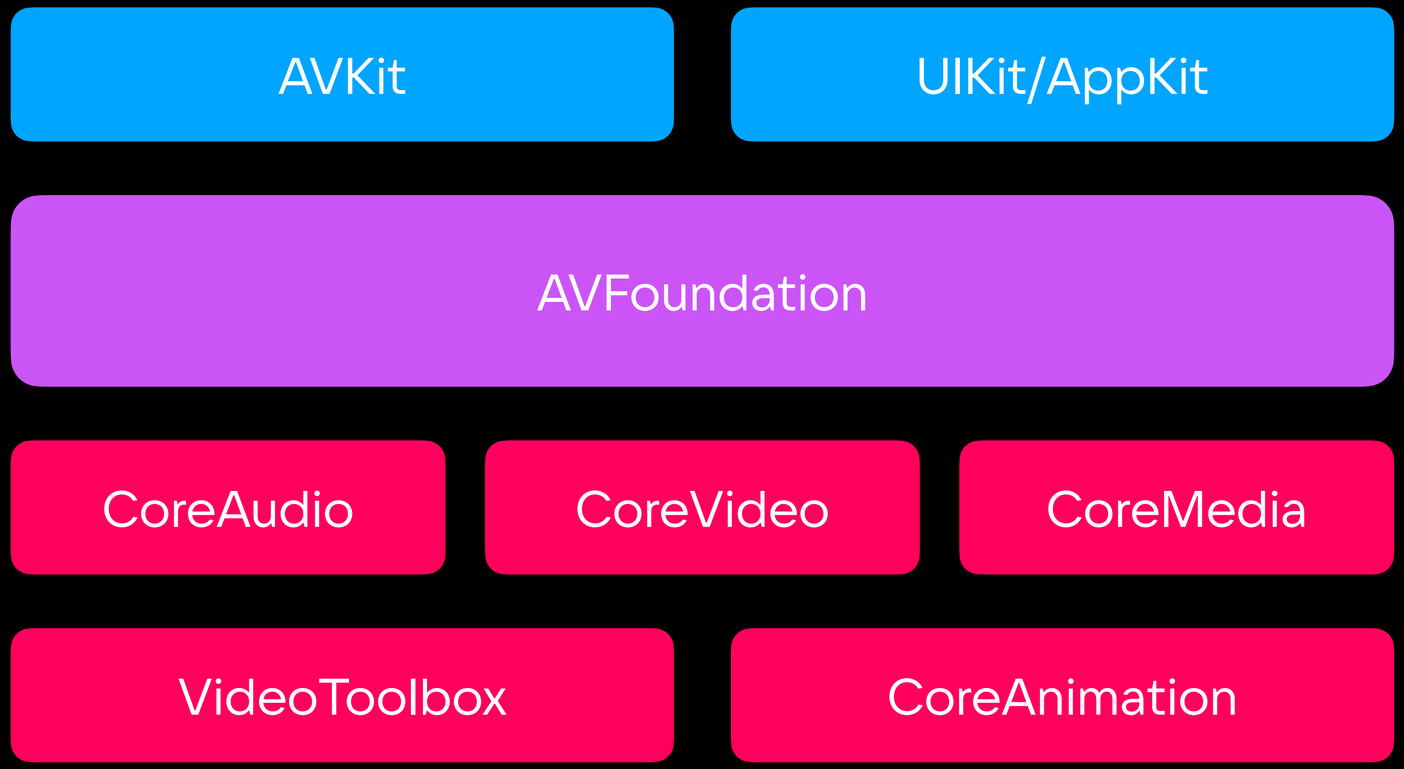
Динамически изменить альфу, наложить тронасформацию или применить кроп.

Чеклист

1. Что будем использовать?
2. Что такое медиаданные?
3. Как данные представлены у нас?
4. Как представляется время?
5. Как редактировать?

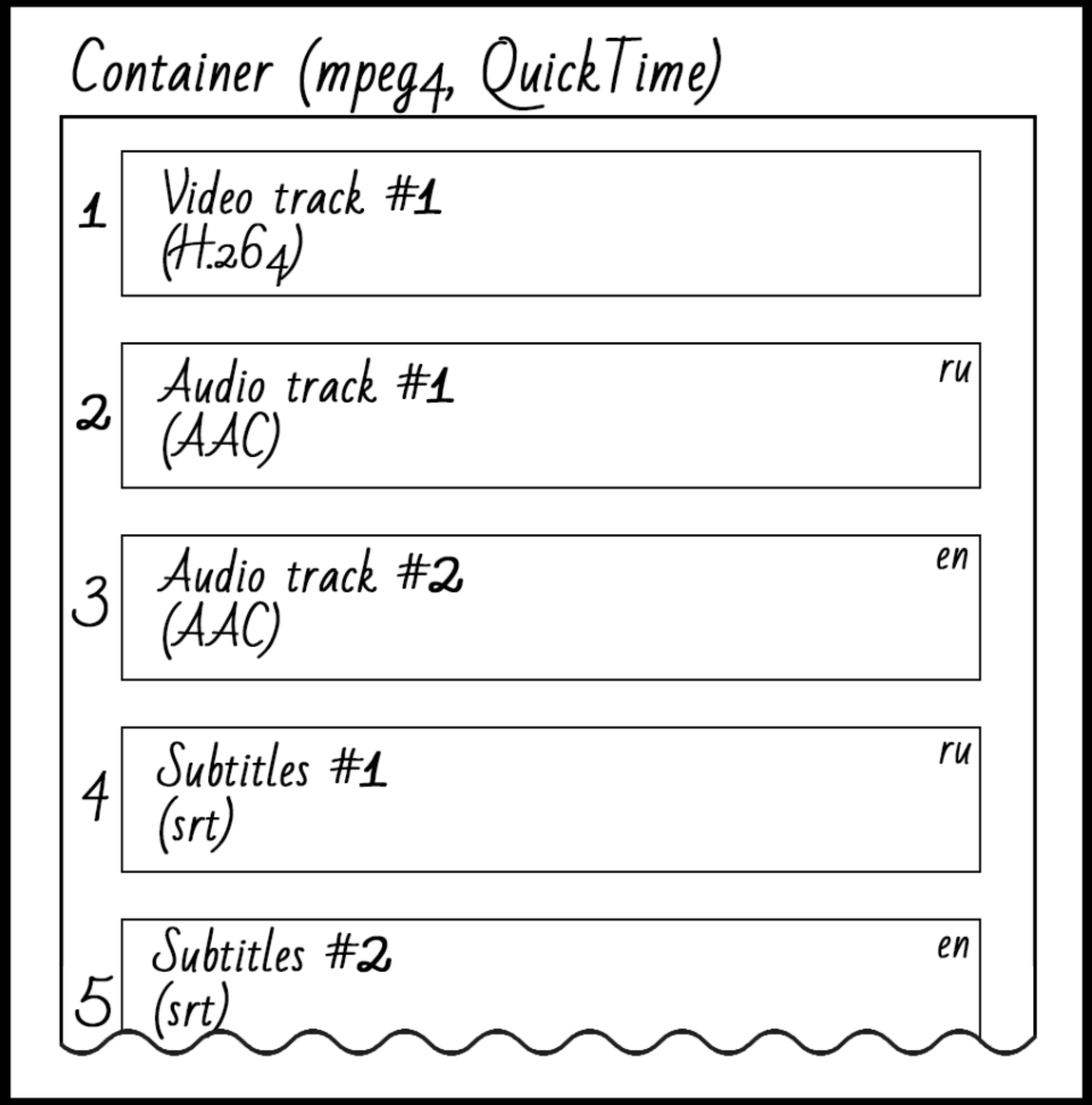
1) Что будем использовать?

Фреймворки,
отвечающие
за работу с media



Чеклист

- ✓ Что будем использовать?
- ▶ Что такое медиаданные?
- 3 Как данные представлены у нас?
- 4 Как представляется время?
- 5 Как редактировать?

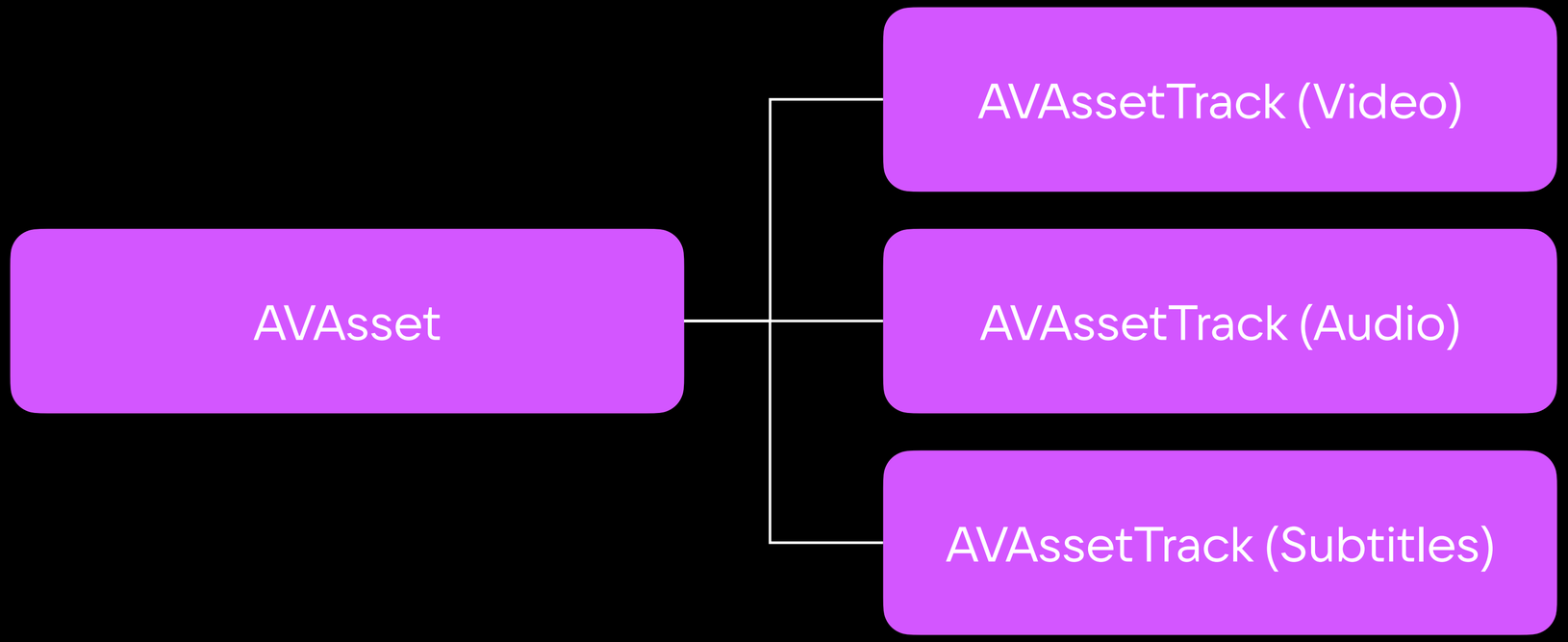


2) Что такое медиаданные?

Чеклист

- ✓ Что будем использовать?
- ✓ Что такое медиаданные?
- ▶ Как данные представлены у нас?
- 4 Как представляется время?
- 5 Как редактировать?

3) Как представляет данные AVFoundation?



Внутреннее представление данных AVFoundation

```
let asset = AVAsset(url: url)
asset.loadValuesAsynchronously(
    forKeys: [
        #keyPath(AVAsset.tracks),
        #keyPath(AVAsset.duration),
        #keyPath(AVAsset.commonMetadata)
    ],
    completionHandler: {
      completion?(true)
    }
)
```

Чеклист

- ✓ Что будем использовать?
- ✓ Что такое медиаданные?
- ✓ Как данные представлены у нас?
- ▶ Как представляется время?
- 5 Как редактировать?

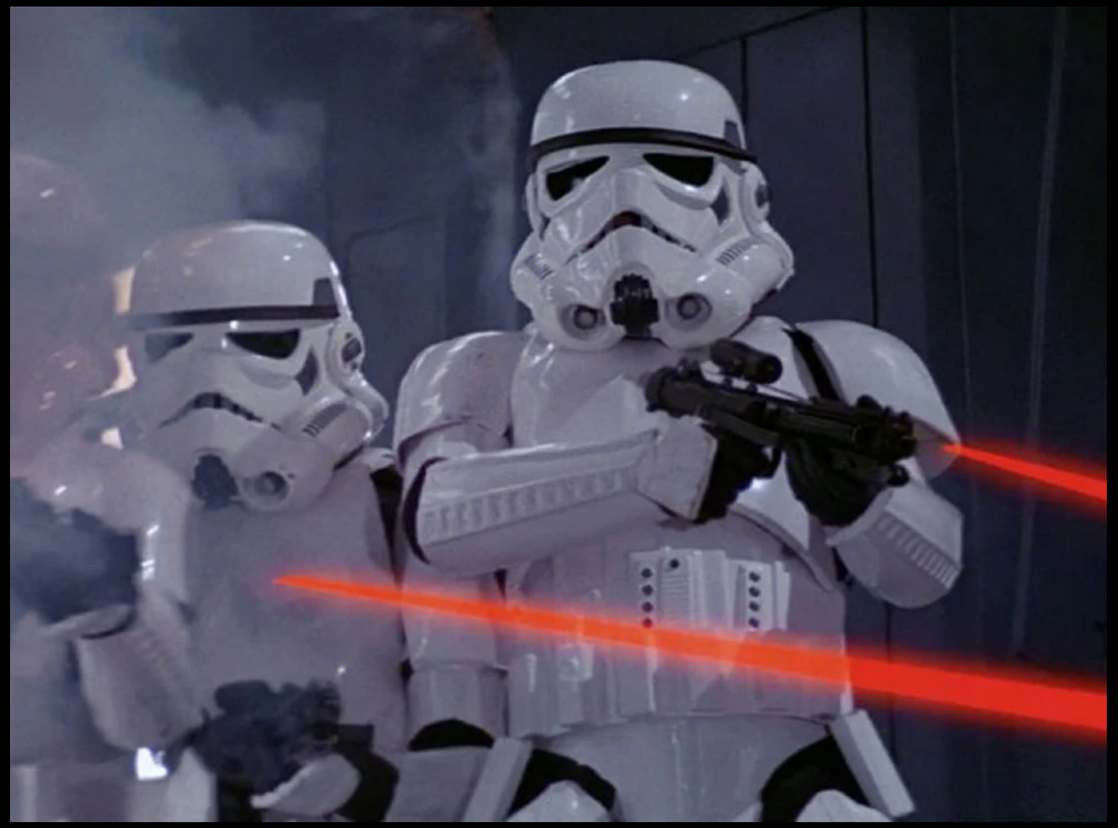
4) Как представляется время?

В чем проблема
TimeInterval (Double)?

Хочу взять 4 кадр у видео с фреймрейтом 60 к / с

Для этого обращаюсь к 4 / 60 =
= 0.066666666...

Точность отсутствует



4) Как представляется время?

```
typedef struct {
    CMTimeValue value;
    CMTimeScale timescale;
    CMTimeFlags flags;
    CMTimeEpoch epoch;
} CMTime;
```

CMTime

```
let time1 = CMTime(value: 3, timescale: 1)
let time2 = CMTime(value: 1800, timescale: 600)
let time3 = CMTime(value: 3000, timescale: 1000)
```

4) Как представляется время?

```
typedef struct {
    CMTime start;
    CMTime duration;
} CMTimeRange;
```

CMTimeRange

```
let time = CMTime(value: 1800, timescale: 600)
let timeRange1 = CMTimeRange(start: time, duration: time)
let timeRange2 = CMTimeRange(start: time, duration: CMTimeMultiplyByFloat64(time, multiplier: 2.0))
```

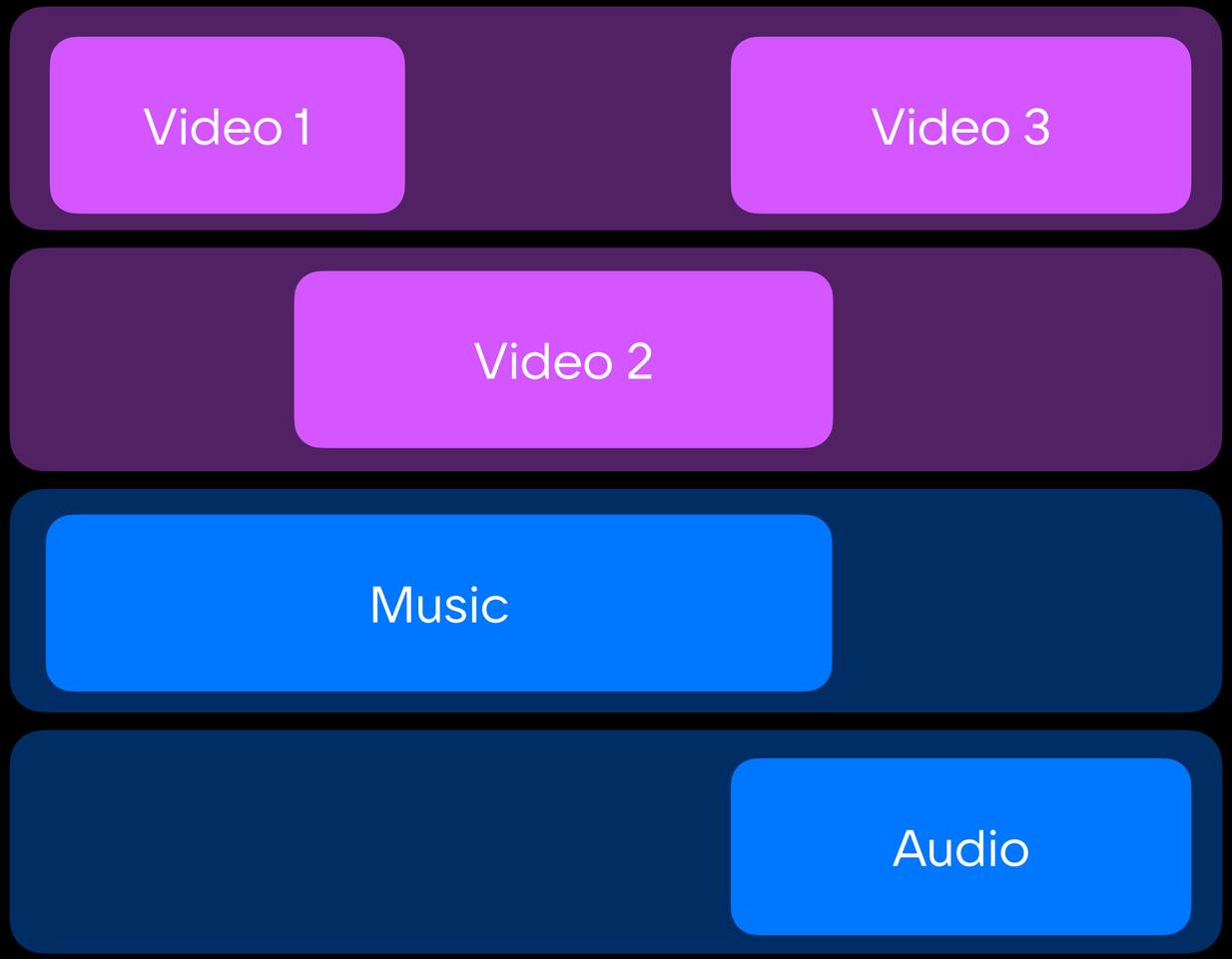
Время — это рациональное число

Чеклист

- ✓ Что будем использовать?
- ✓ Что такое медиаданные?
- ✓ Как данные представлены у нас?
- ✓ Как представляется время?
- ▶ Как редактировать?

5) Как редактировать?

Создание AVComposition



Редактирование медиа

Создание AVComposition

```
let mutableAVComposition = AVMutableComposition()

let videoTrack = mutableAVComposition.addMutableTrack(
    withMediaType: .video,
    preferredTrackID: kCMPersistentTrackID_Invalid
)!
let audioTrack = mutableAVComposition.addMutableTrack(
    withMediaType: .audio,
    preferredTrackID: kCMPersistentTrackID_Invalid
)!
```

```
let firstAsset = AVAsset(url: firstURL)
let secondAsset = AVAsset(url: secondURL)

let firstVideoTrack = firstAsset.tracks(withMediaType: .video).first!
let secondVideoTrack = secondAsset.tracks(withMediaType: .video).first!
```

```
try? videoTrack.insertTimeRange(
    firstVideoTrack.timeRange,
    of: firstVideoTrack,
    at: .zero
)
try? videoTrack.insertTimeRange(
    secondVideoTrack.timeRange,
    of: secondVideoTrack,
    at: firstVideoTrack.timeRange.end
)
```

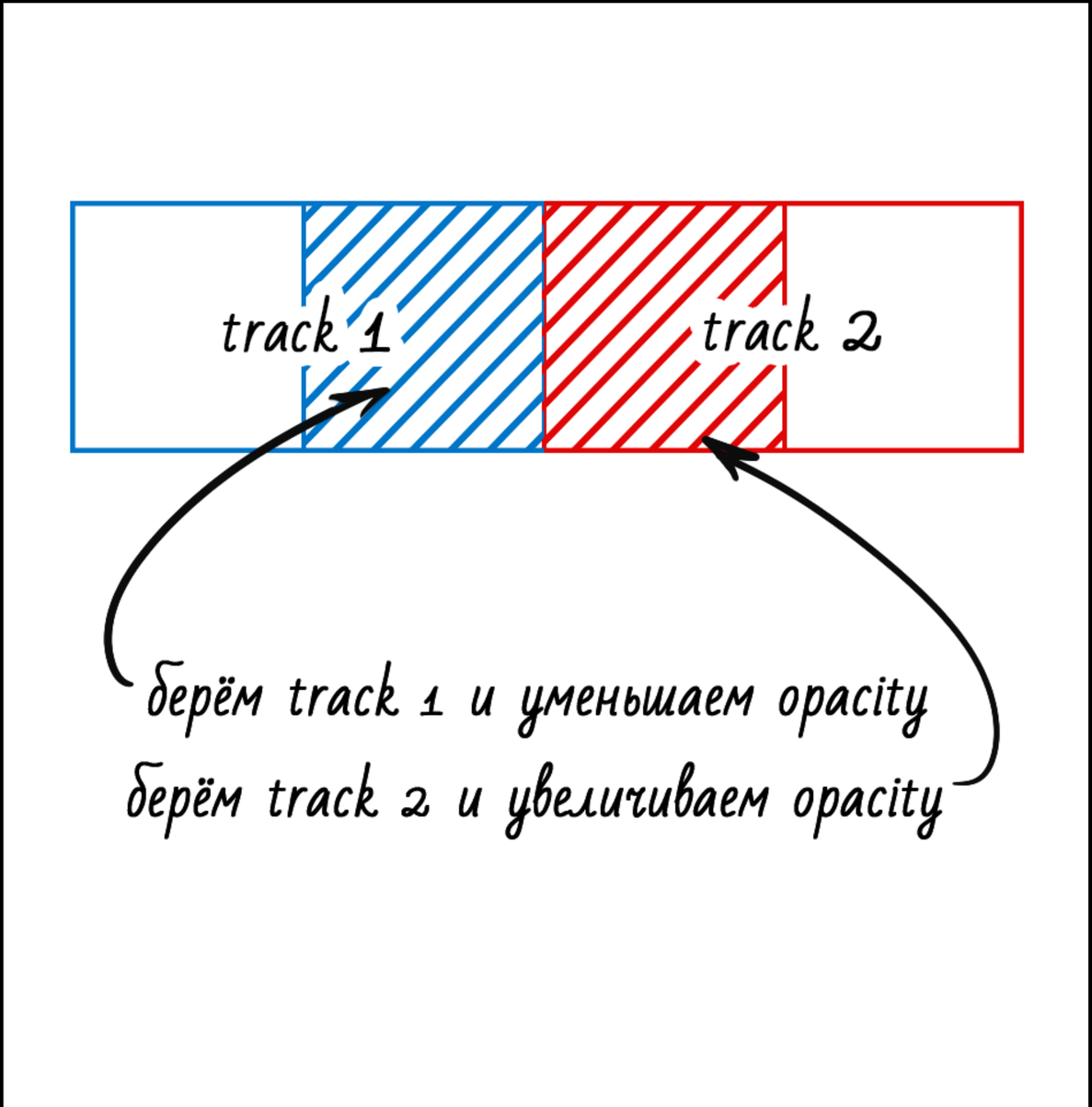
Редактирование медиа

Создание AVComposition

AVComposition наследник AVAsset
(Без ссылки на локальное хранение файла)

Редактирование медиа

AVVideoComposition



Редактирование медиа

AVVideoComposition

```
let firstTrack = firstAsset.tracks(withMediaType: .video).first!

// Создаем Video Composition Instruction
let fromTrackInstruction = AVMutableVideoCompositionInstruction()
let halfOfFirstVideoDuration = CMTimeMultiplyByFloat64(firstTrack.timeRange.duration, multiplier: 0.5)
let fromTrackInterval = CMTimeRange(start: halfOfFirstVideoDuration, duration: halfOfFirstVideoDuration)
fromTrackInstruction.timeRange = fromTrackInterval

// Помещаем в нее Layer Instruction
let fromTrackLayerInstruction = AVMutableVideoCompositionLayerInstruction(assetTrack: firstTrack)
fromTrackLayerInstruction.setOpacityRamp(fromStartOpacity: 1, toEndOpacity: 0, timeRange:
fromTrackInterval)
fromTrackInstruction.layerInstructions = [fromTrackLayerInstruction]

let videoComposition = AVMutableVideoComposition()
videoComposition.instructions = [fromTrackInstruction]
```

Редактирование медиа

AVAudioMix

```
let audioTrack = firstAsset.tracks(withMediaType: .audio).first!

// Задаем громкость конкретному аудио-треку
let audioMixInputParametersMusic = AVMutableAudioMixInputParameters()
audioMixInputParametersMusic.trackID = audioTrack.trackID
audioMixInputParametersMusic.setVolume(0.5, at: .zero)

// Помещаем параметр в AudioMix
let audioMix = AVMutableAudioMix()
audioMix.inputParameters = [audioMixInputParametersMusic]
```

Редактирование медиа

AVComposition + AVVideoComposition + AVAudioMix

AVComposition

положение всех треков во времени

AVVideoComposition

как представляем видео в определенный момент времени

AVAudioMix

как представляем аудио в определенный момент времени

Редактирование медиа

AVComposition + AVVideoComposition + AVAudioMix

```
let playerItem = AVPlayerItem(asset: composition)
playerItem.videoComposition = videoComposition
playerItem.audioMix = audioMix

let player = AVPlayer(playerItem: playerItem)
player.play()
```

```
let destinationURL =
FileManager.default.temporaryDirectory.appendingPathComponent(
    "\(UUID().uuidString).mov"
)
let exporter = AVAssetExportSession(
    asset: composition,
    presetName: AVAssetExportPresetHighestQuality
)
exporter?.outputURL = destinationURL
exporter?.videoComposition = videoComposition
exporter?.audioMix = audioMix
exporter?.exportAsynchronously {
    // В этот момент видео сохранено локально по destinationURL
}
```

Необъяснимый баг

Так выглядит композиция,
когда я ее создаю

Видео

Аудио

Так все работает,
когда я использую AVPlayer

Видео

Аудио

Необъяснимый баг

```
let assetForComposition = AVURLAsset(
    url: firstURL, options: [AVURLAssetPreferPreciseDurationAndTimingKey: true]
)

// Теперь можем спокойно доставать из него tracks и заполнять AVComposition
```

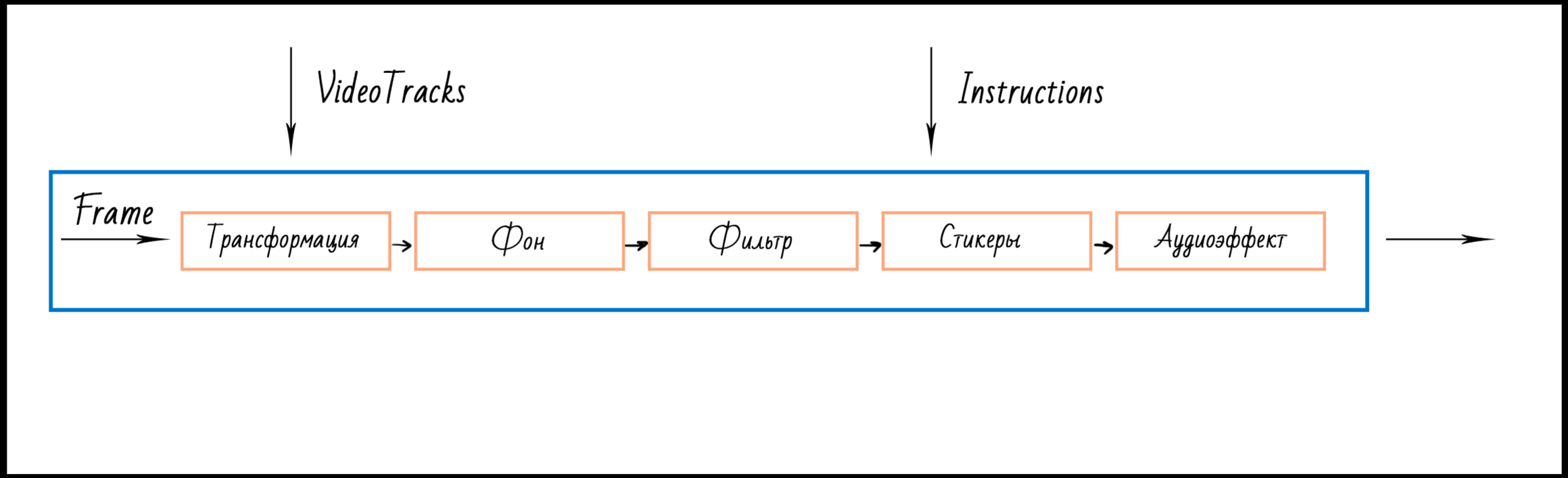
Чеклист

- Что будем использовать?
- Что такое медиаданные?
- Как данные представлены у нас?
- Как представляется время?
- Как редактировать?

База закончилась. Начались фичи

Интересные фичи, касающиеся разных аспектов фреймворка AVFoundation

1) Наложение на кадр видео (стикеры, фильтры, трансформация, фон)



AVVideoComposition

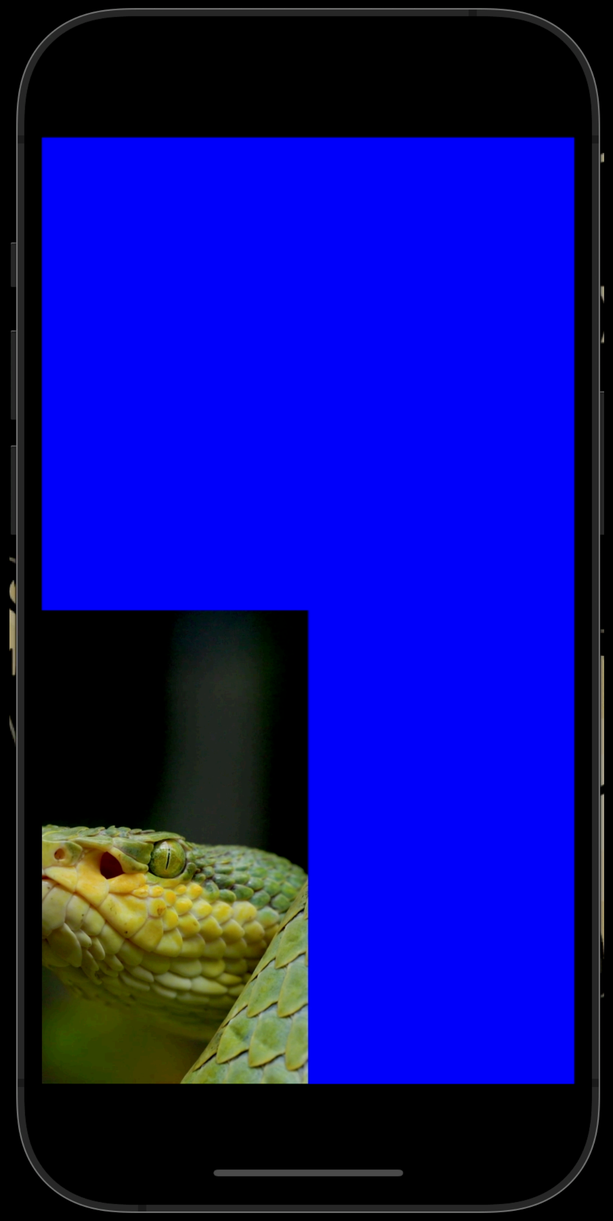
customVideoCompositorClass

AVVideoCompositing

```
class CustomVideoCompositing: NSObject, AVVideoCompositing {

    private let renderingQueue = DispatchQueue(label: "com.example.fullcompositing.render")
    /// Создаем Metal-контекст для рендеринга картинок
    /// Можно использовать Core Image уровень вместо этого
    private lazy var context: CIContext! = {
        guard let device = MTLCreateSystemDefaultDevice() else {
            fatalError("Failed to get Metal device")
        }
        return CIContext(
            mtlDevice: device,
            options: [
                CIContextOption.workingColorSpace: NSNull(),
                CIContextOption.cacheIntermediates: false
            ]
        )
    }()

    var sourcePixelBufferAttributes: [String: Any]? {
        return [
            kCVPixelBufferPixelFormatTypeKey as String: kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange
        ]
    }

    var requiredPixelBufferAttributesForRenderContext: [String: Any] {
        return [
            kCVPixelBufferPixelFormatTypeKey as String: kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange
        ]
    }
```

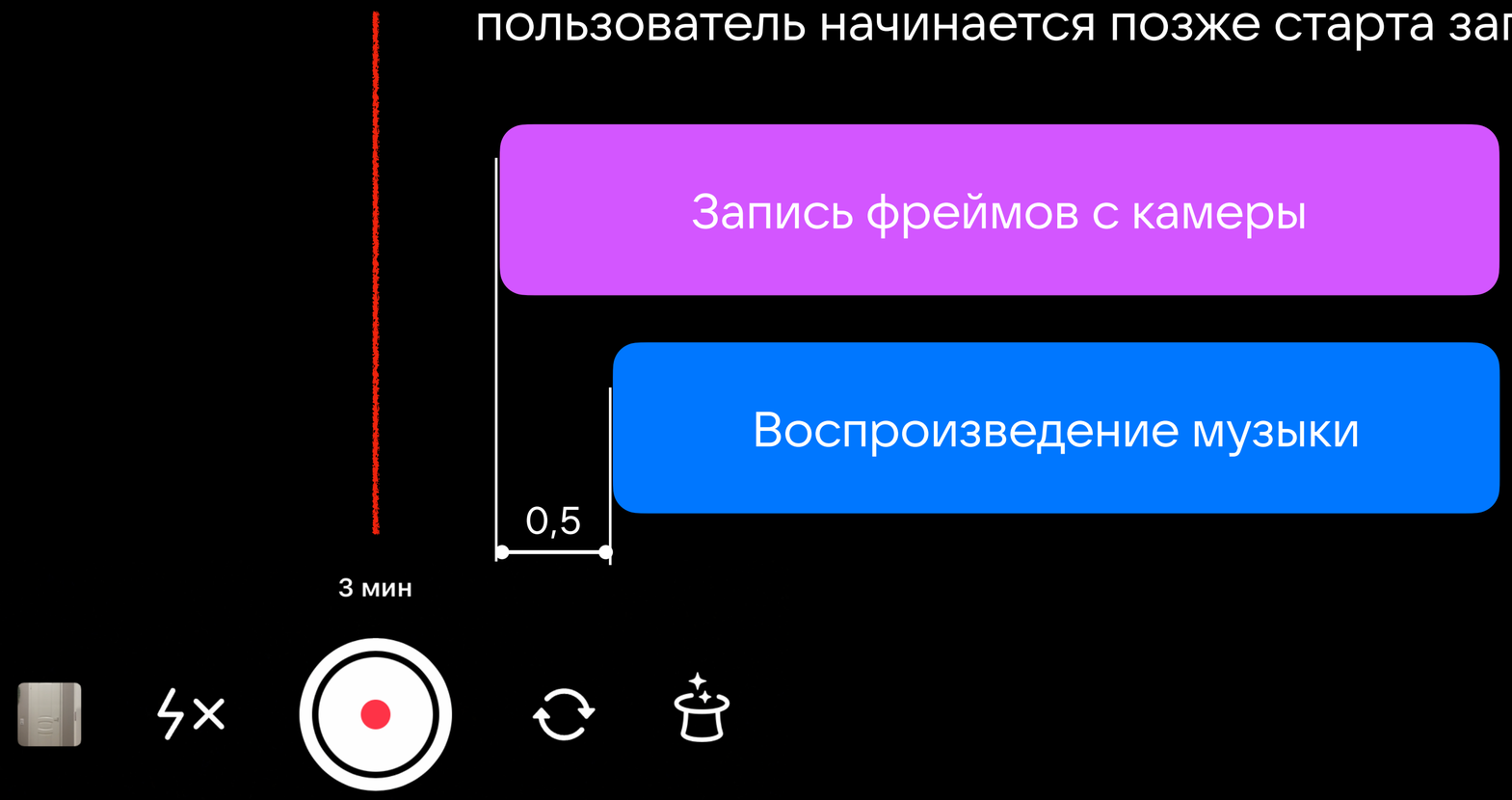
AVVideoCompositing

```
func startRequest(_ asyncVideoCompositionRequest: AVAsynchronousVideoCompositionRequest) {
    renderingQueue.async { [weak self] in
        guard let self = self else { return }

        autoreleasepool {
            let newPixelBuffer = asyncVideoCompositionRequest.renderContext.newPixelBuffer()!
            let compositingTime = asyncVideoCompositionRequest.compositionTime

            if let instruction = asyncVideoCompositionRequest.videoCompositionInstruction as? AVVideoCompositionInstruction {
                for layerInstruction in instruction.layerInstructions {
                    if let pixelBuffer = asyncVideoCompositionRequest.sourceFrame(
                        byTrackID: layerInstruction.trackID
                    ) {
                        /// Получаем текущую картинку видео
                        var image = CIImage(cvPixelBuffer: pixelBuffer)

                        /// Применяем к ней трансформацию скейла
                        let timeMulti = compositingTime.seconds.truncatingRemainder(dividingBy: 6.0) / 3.0
                        let scale = max(timeMulti < 1.0 ? timeMulti : 2.0 - timeMulti, 0.5)
                        image = image.transformed(by: .init(scaleX: scale, y: scale))

                        /// Задаем синий цвет фона
                        image = image.composited(over: .init(color: .blue))

                        /// Рендерим на пустой новый буфер картинку
                        self.context.render(image, to: newPixelBuffer)
                    }
                }
            }

            /// Отправляем новый буфер реквесту, как бы говоря, отправь его в AVPlayer или Exporter
            asyncVideoCompositionRequest.finish(withComposedVideoFrame: newPixelBuffer)
        }
    }
}
```



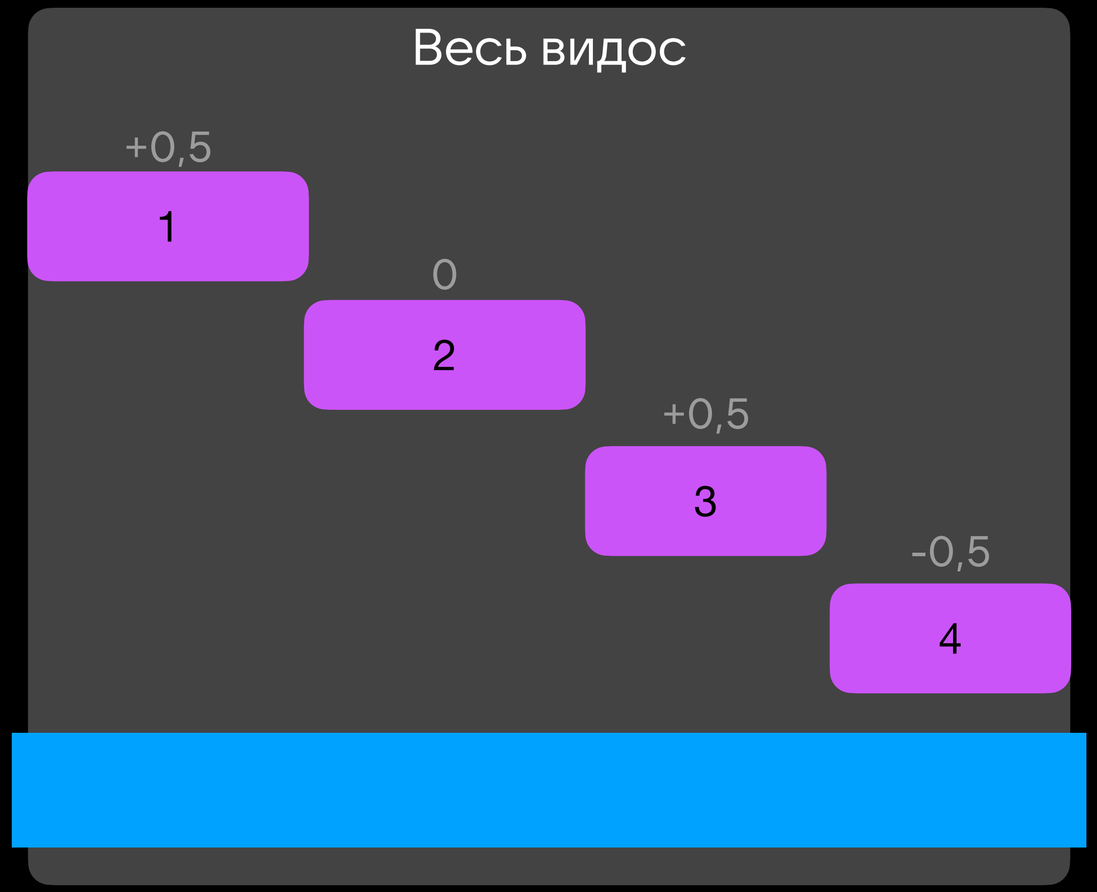
2) Липсинк

Воспроизведение музыки под которую поет пользователь начинается позже старта записи



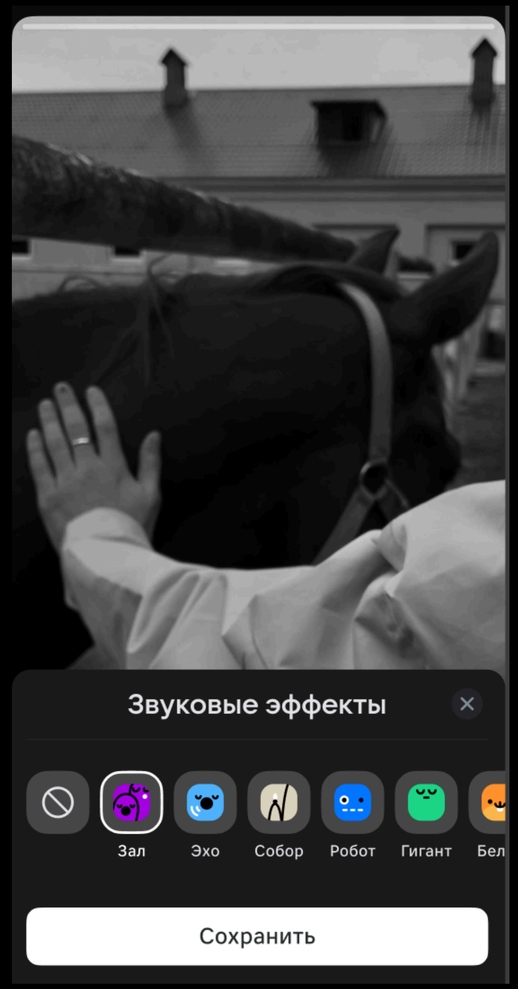
2) Липсинк

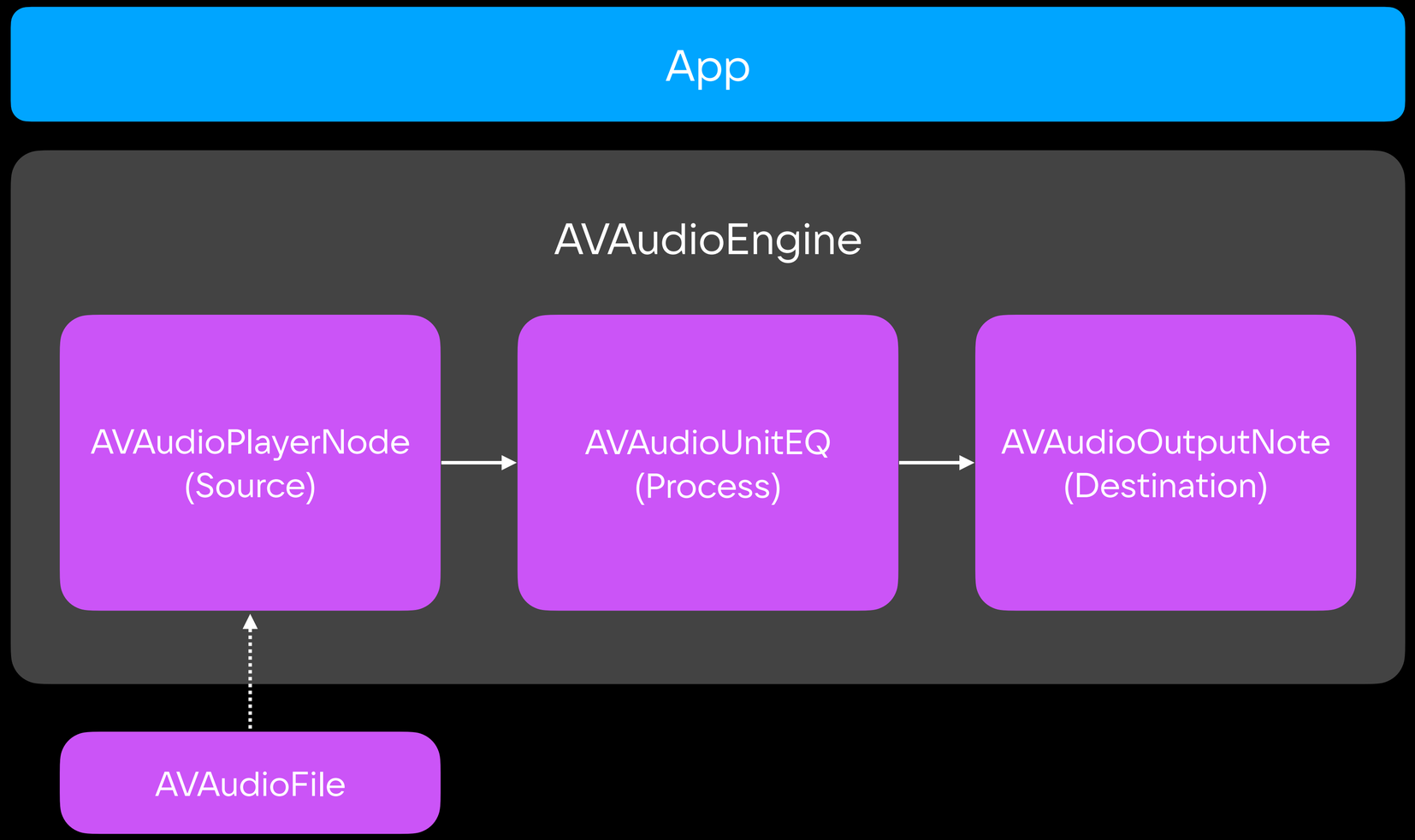
Съемка и заполнение



Итог

3) Аудио эффекты (AVAudioEngine)





Работа с AVAudioEngine

```
let sourceFile = try! AVAudioFile(forReading: audioURL)
let format = sourceFile.processingFormat

let engine = AVAudioEngine()
let player = AVAudioPlayerNode()
let reverb = AVAudioUnitReverb()

engine.attach(player)
engine.attach(reverb)

reverb.loadFactoryPreset(.mediumHall)
reverb.wetDryMix = 50

engine.connect(player, to: reverb, format: format)
engine.connect(reverb, to: engine.mainMixerNode, format: format)

player.scheduleFile(sourceFile, at: nil)

let maxFrames: AVAudioFrameCount = 4096
try? engine.enableManualRenderingMode(.offline, format: format, maximumFrameCount: maxFrames)

try? engine.start()
player.play()
```

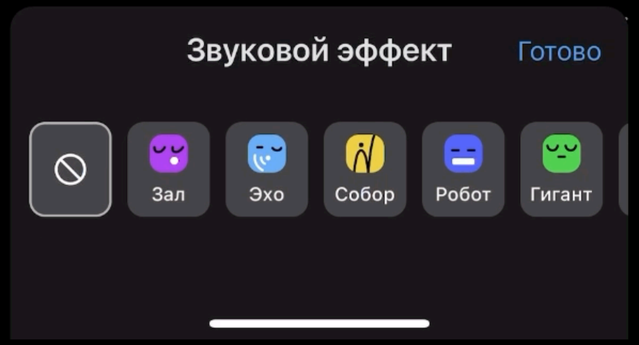
Работа с AVAudioEngine

```
let outputURL = URL.init(fileURLWithPath: NSTemporaryDirectory())
    .appendingPathComponent("\(UUID.init().uuidString).m4a")
var outputFile: AVAudioFile?
do {
    outputFile = try AVAudioFile(
        forWriting: outputURL,
        settings: sourceFile.fileFormat.settings
    )
} catch {
    outputFile = try? AVAudioFile(
        forWriting: outputURL,
        settings: [
            AVFormatIDKey: kAudioFormatMPEG4AAC,
            AVNumberOfChannelsKey: 2,
            AVSampleRateKey: 44100,
            AVEncoderBitRateKey: 128000
        ]
    )
}
```

Работа с AVAudioEngine

```
let buffer = AVAudioPCMBuffer(
    pcmFormat: engine.manualRenderingFormat,
    frameCapacity: engine.manualRenderingMaximumFrameCount
)!
while engine.manualRenderingSampleTime < sourceFile.length {
    let frameCount = sourceFile.length - engine.manualRenderingSampleTime
    let framesToRender = min(AVAudioFrameCount(frameCount), buffer.frameCapacity)

    let status = try? engine.renderOffline(framesToRender, to: buffer)

    if status == .success {
        try? outputFile?.write(from: buffer)
    }
}

player.stop()
engine.stop()

let url = outputFile!.url
outputFile = nil
return url
```

0:01 .9 / 0:22 .6



Работа с AVAudioEngine

```
let timePitch = AVAudioUnitTimePitch()
let reverb = AVAudioUnitReverb()
let distortion = AVAudioUnitDistortion()
```

```
@available(iOS 8.0, *)
public enum AVAudioUnitReverbPreset : Int, @unchecked Sendable {

    case smallRoom = 0
    case mediumRoom = 1
    case largeRoom = 2
    case mediumHall = 3
    case largeHall = 4
    case plate = 5
    case mediumChamber = 6
    case largeChamber = 7
    case cathedral = 8
    case largeRoom2 = 9
    case mediumHall2 = 10
    case mediumHall3 = 11
    case largeHall2 = 12
}
```

```
@available(iOS 8.0, *)
public enum AVAudioUnitDistortionPreset : Int, @unchecked Sendable {

    case drumsBitBrush = 0
    case drumsBufferBeats = 1
    case drumsLoFi = 2
    case multiBrokenSpeaker = 3
    case multiCellphoneConcert = 4
    case multiDecimated1 = 5
    case multiDecimated2 = 6
    case multiDecimated3 = 7
    case multiDecimated4 = 8
    case multiDistortedFunk = 9
    case multiDistortedCubed = 10
    case multiDistortedSquared = 11
    case multiEcho1 = 12
    case multiEcho2 = 13
    case multiEchoTight1 = 14
    case multiEchoTight2 = 15
    case multiEverythingIsBroken = 16
    case speechAlienChatter = 17
    case speechCosmicInterference = 18
    case speechGoldenPi = 19
    case speechRadioTower = 20
    case speechWaves = 21
}
```

4) Видео из фото (AVAssetWriter)

```
/// Создаем файл для сохранения видео из фото
let outputFileURL = FileManager.default.temporaryDirectory.appendingPathComponent(
    "\(UUID().uuidString).mov"
)

/// Инициализируем AVAssetWriter
let videoWriter = try! AVAssetWriter(
    outputURL: outputFileURL,
    fileType: .mov
)

/// Заполняем настройки для AVAssetWriterInput
let videoSettings: [String: Any] = [
    AVVideoCodecKey: AVVideoCodecType.h264,
    AVVideoWidthKey: videoSize.width,
    AVVideoHeightKey: videoSize.height
]

/// Создаем AVAssetWriterInput
let videoWriterInput = AVAssetWriterInput(
    mediaType: .video,
    outputSettings: videoSettings
)
```

Создание AVAssetWriter и AVAAssetWriterInput

4) Видео из фото (AVAssetWriter)

```
/// Создаем AVAssetWriterInputPixelBufferAdaptor и указываем что он будет работать для типа RGB
let adaptor = AVAssetWriterInputPixelBufferAdaptor(
    assetWriterInput: videoWriterInput,
    sourcePixelBufferAttributes: [
        kCVPixelBufferPixelFormatTypeKey as String: Int(kCVPixelFormatType_32ARGB)
    ]
)

/// Проверяем AVAssetWriter на возможность начать запись объектами
guard videoWriter.canAdd(videoWriterInput) else { return }

videoWriterInput.expectsMediaDataInRealTime = true
/// Добавляем AVAssetWriterInput в AVAssetWriter
videoWriter.add(videoWriterInput)

/// Сообщаем AVAssetWriter-у о начале записи и указываем начальное положение времени к .zero
videoWriter.startWriting()
videoWriter.startSession(atSourceTime: .zero)
```

Создание AVAssetWriterInput PuxelBuffetAdapter и начало записи

4) Видео из фото (AVAssetWriter)

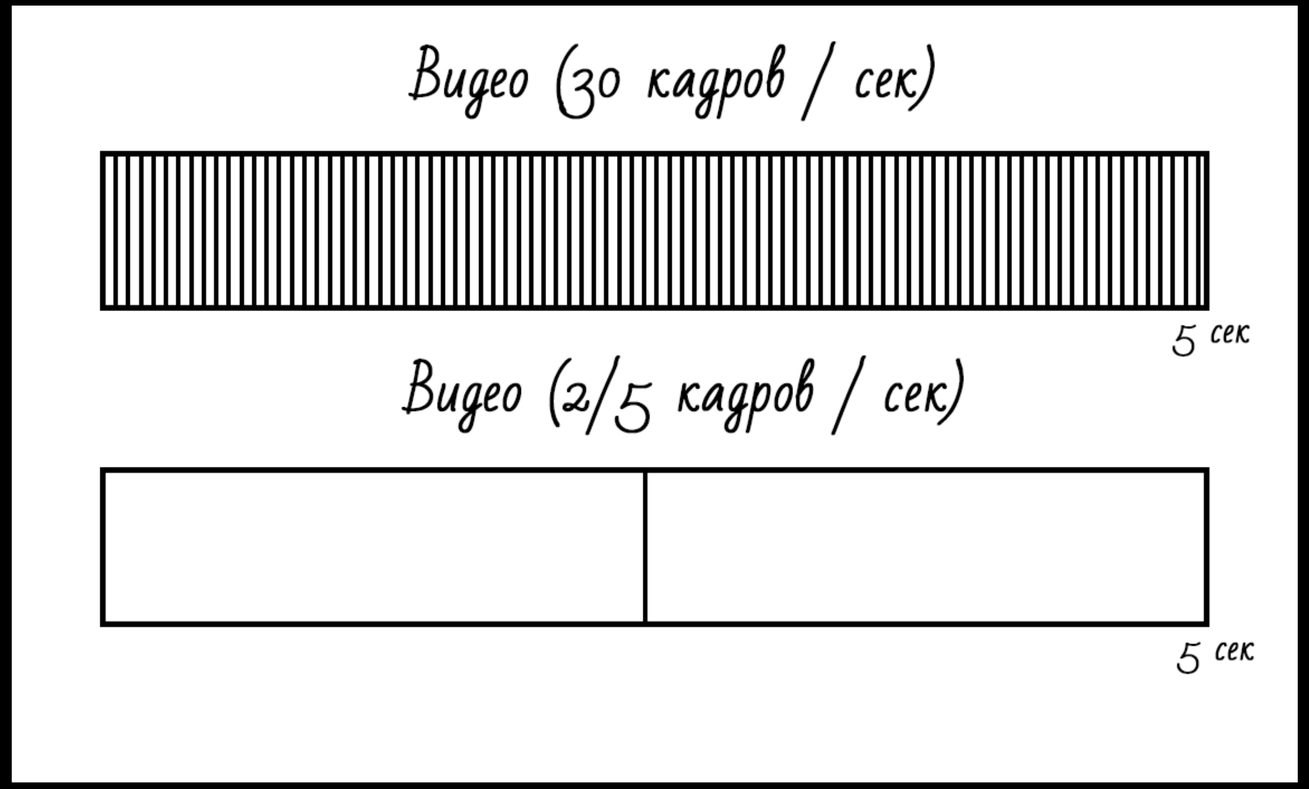
```
private func pixelBuffer(fromImage image: CGImage, size: CGSize) throws -> CVPixelBuffer {
    let options: CFDictionary = [kCVPixelBufferCGImageCompatibilityKey as String: true, kCVPixelBufferCGBitmapContextCompatibilityKey as String:
true] as CFDictionary
    var pxbuffer: CVPixelBuffer?
    let status = CVPixelBufferCreate(
        kCFAllocatorDefault,
        Int(size.width),
        Int(size.height),
        kCVPixelFormatType_32ARGB,
        options,
        &pxbuffer
    )
    guard let buffer = pxbuffer, status == kCVReturnSuccess else {
        throw NSError("CVPixelBufferCreate \(status) != kCVReturnSuccess")
    }

    let bufferLockStatus = CVPixelBufferLockBaseAddress(buffer, [])
    guard bufferLockStatus == kCVReturnSuccess else {
        throw NSError("CVPixelBufferLockBaseAddress \(bufferLockStatus) != kCVReturnSuccess")
    }

    guard let pxdata = CVPixelBufferGetBaseAddress(buffer) else {
        throw NSError("Failed to create CVPixelBufferGetBaseAddress")
    }
    let bytesPerRow = CVPixelBufferGetBytesPerRow(buffer)

    let rgbColorSpace = CGColorSpaceCreateDeviceRGB()
    guard let context = CGContext(
        data: pxdata,
        width: Int(size.width),
        height: Int(size.height),
        bitsPerComponent: 8,
        bytesPerRow: bytesPerRow,
        space: rgbColorSpace,
        bitmapInfo: CGImageAlphaInfo.noneSkipFirst.rawValue
    ) else {
        throw NSError(
            "Failed to create CGContext with CVPixelBufferGetBaseAddress, width: \(Int(size.width)), height: \(Int(size.height))"
        )
    }
    context.concatenate(CGAffineTransform(rotationAngle: 0))
    context.draw(image, in: CGRect(x: 0, y: 0, width: size.width, height: size.height))

    let bufferUnlockStatus = CVPixelBufferUnlockBaseAddress(buffer, [])
    guard bufferUnlockStatus == kCVReturnSuccess else {
        throw NSError("CVPixelBufferUnlockBaseAddress \(bufferUnlockStatus) != kCVReturnSuccess")
    }

    return buffer
}
```

Создание CVPixelBuffer из картинки

4) Видео из фото (AVAssetWriter)

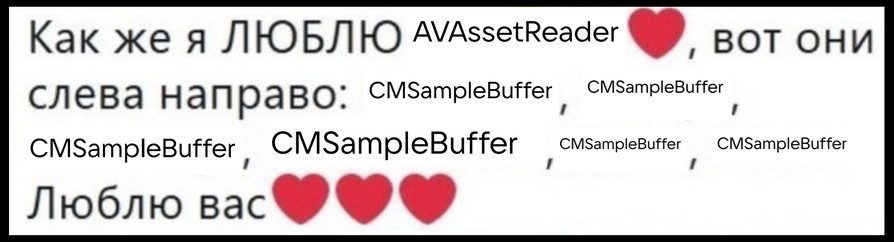
Процесс записи

```
adaptor.assetWriterInput.requestMediaDataWhenReady(on: queue) {
    var isFinished = true

    while index < buffersCount {
        if adaptor.assetWriterInput.isReadyForMoreMediaData == false {
            isFinished = false
            break
        }

        adaptor.append(buffer, withPresentationTime: startTime)
        startTime = CMTimeAdd(startTime, frameDuration)
        index += 1
    }

    if isFinished {
        adaptor.assetWriterInput.markAsFinished()
        if writer.status == .cancelled {
            completion(nil)
            return
        }
        writer.finishWriting {
            completion(writer.outputURL)
        }
    }
}
```



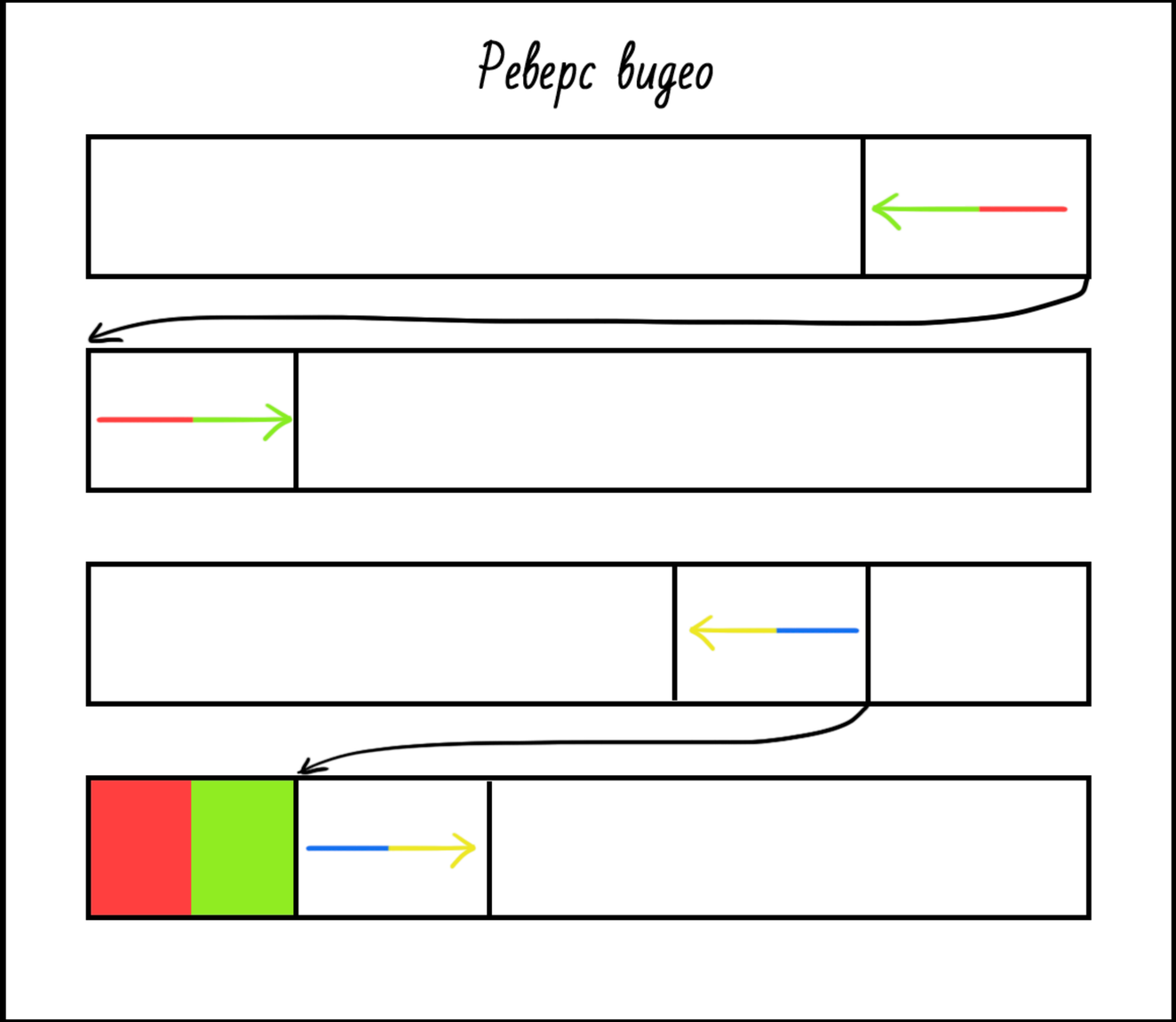
5) Реверс (AVAssetReader и AVAssetWriter)

```
let assetReader = try? AVAssetReader(asset: asset)
guard let videoTrack = asset.tracks(withMediaType: .video).first else { return }

let assetReaderOutput = AVAssetReaderTrackOutput(
    track: videoTrack,
    outputSettings: [
        kCVPixelBufferPixelFormatTypeKey as String: kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange
    ]
)
assetReaderOutput.supportsRandomAccess = true
assetReader?.add(assetReaderOutput)
assetReader?.startReading()

while let buffer = assetReaderOutput.copyNextSampleBuffer() {
    /// Здесь вы можете работать с буферами
    print(buffer.presentationTimeStamp)
}
```

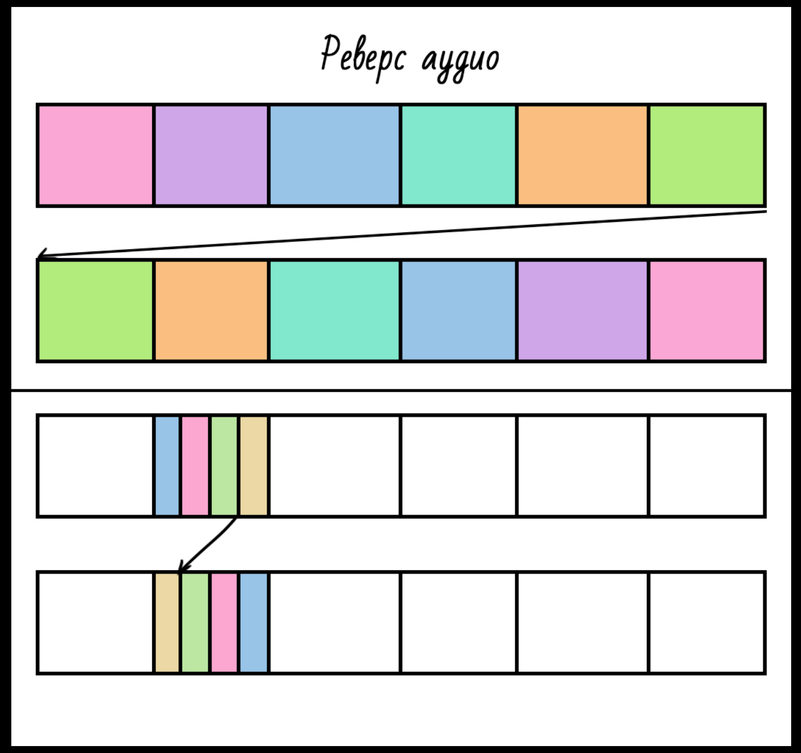



```
let nextTimeRange = CMTimeRange(
    start: .init(value: 2400, timescale: 600),
    duration: .init(value: 40, timescale: 600)
)
assetReaderOutput.reset(forReadingTimeRanges: [NSValue(timeRange: nextTimeRange)])
```

Реверс видео

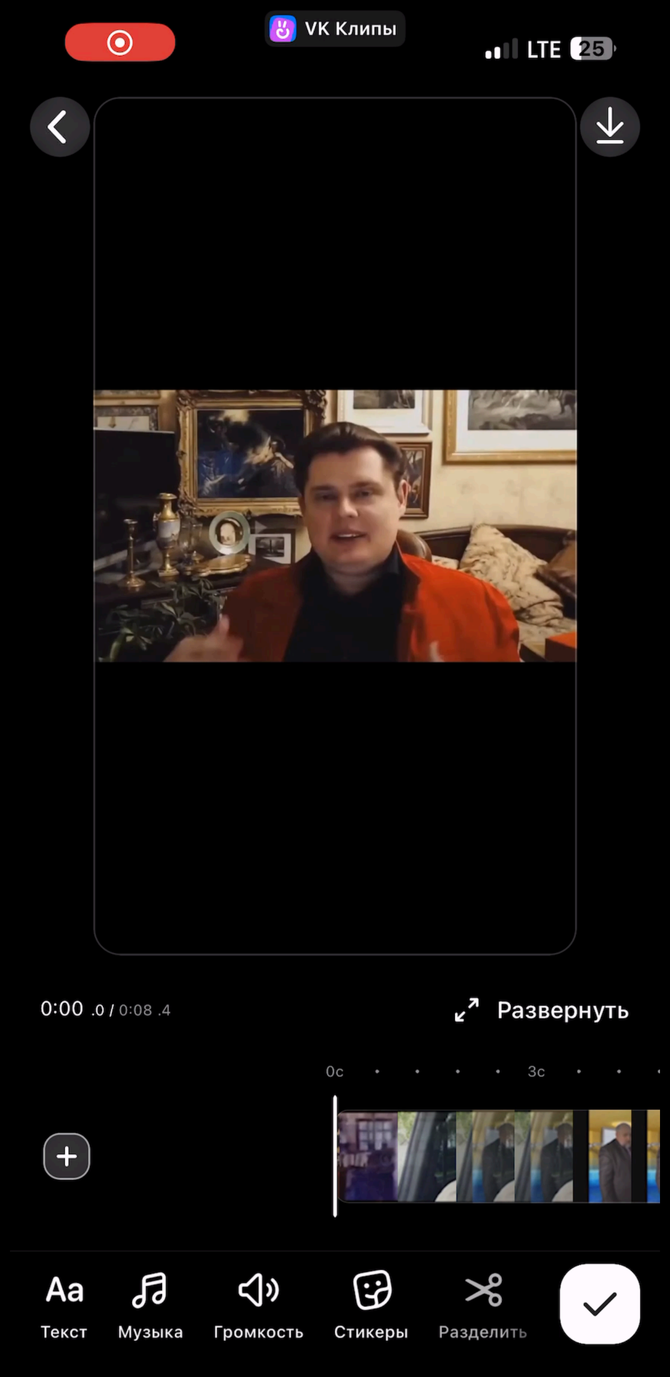


Реверс аудио



Реверс аудио-буфера

```
+ (CMSampleBufferRef)reverseCMSampleBufferWithBuffer:(CMSampleBufferRef)buffer timingInfo:(CMSampleTimingInfo)timingInfo {
    CMBlockBufferRef blockBuffer;
    AudioBufferList audioBufferList;
    audioBufferList.mNumberBuffers = 1;
    CMSampleBufferGetAudioBufferListWithRetainedBlockBuffer(buffer,
                                                            nil,
                                                            &audioBufferList,
                                                            sizeof(AudioBufferList),
                                                            nil,
                                                            nil,
                                                            kCMSampleBufferFlag_AudioBufferList_Assure16ByteAlignment,
                                                            &blockBuffer);
    int16_t* samples = (int16_t*)audioBufferList.mBuffers[0].mData;
    size_t sizeofInt16 = sizeof(int16_t);
    NSUInteger dataSize = audioBufferList.mBuffers[0].mDataByteSize;
    NSUInteger dataCount = dataSize / sizeofInt16;

    int16_t *reversedPointer = (int16_t*)malloc(dataSize);
    NSUInteger index = 0;
    for (index = 0; index < dataCount; index+=1) {
        memcpy(reversedPointer + (dataCount - (index+1)), samples + index, sizeofInt16);
    }

    CMBlockBufferReplaceDataBytes(reversedPointer, blockBuffer, 0, dataSize);

    CMFormatDescriptionRef formatDescription = CMSampleBufferGetFormatDescription(buffer);
    NSUInteger numberOfSamples = CMSampleBufferGetNumSamples(buffer);

    CMSampleBufferRef newBuffer;
    CMSampleBufferCreate(
                         kCFAllocatorDefault,
                         blockBuffer,
                         YES,
                         nil,
                         nil,
                         formatDescription,
                         numberOfSamples,
                         0,
                         &timingInfo,
                         0,
                         nil,
                         &newBuffer);

    free(reversedPointer);

    return newBuffer;
}
```


VK Клипы
LTE 25
0:00 .0 / 0:08 .4
Развернуть
0с
3с
Аа
Текст
Музыка
Громкость
Стикеры
Разделить

AVFoundation все-таки БОЛЬ

- Нельзя оставлять промежутки между треками в одной дорожке (даже 1/600)
- Нельзя допускать, чтобы треки в одной дорожке пересекались (даже на 1/600)
- Нельзя брать ассет длительностью меньше фреймрейта
- Нельзя чтобы аудио-дорожка была длиннее видео-дорожки

Как хранить данные и работать с ними?

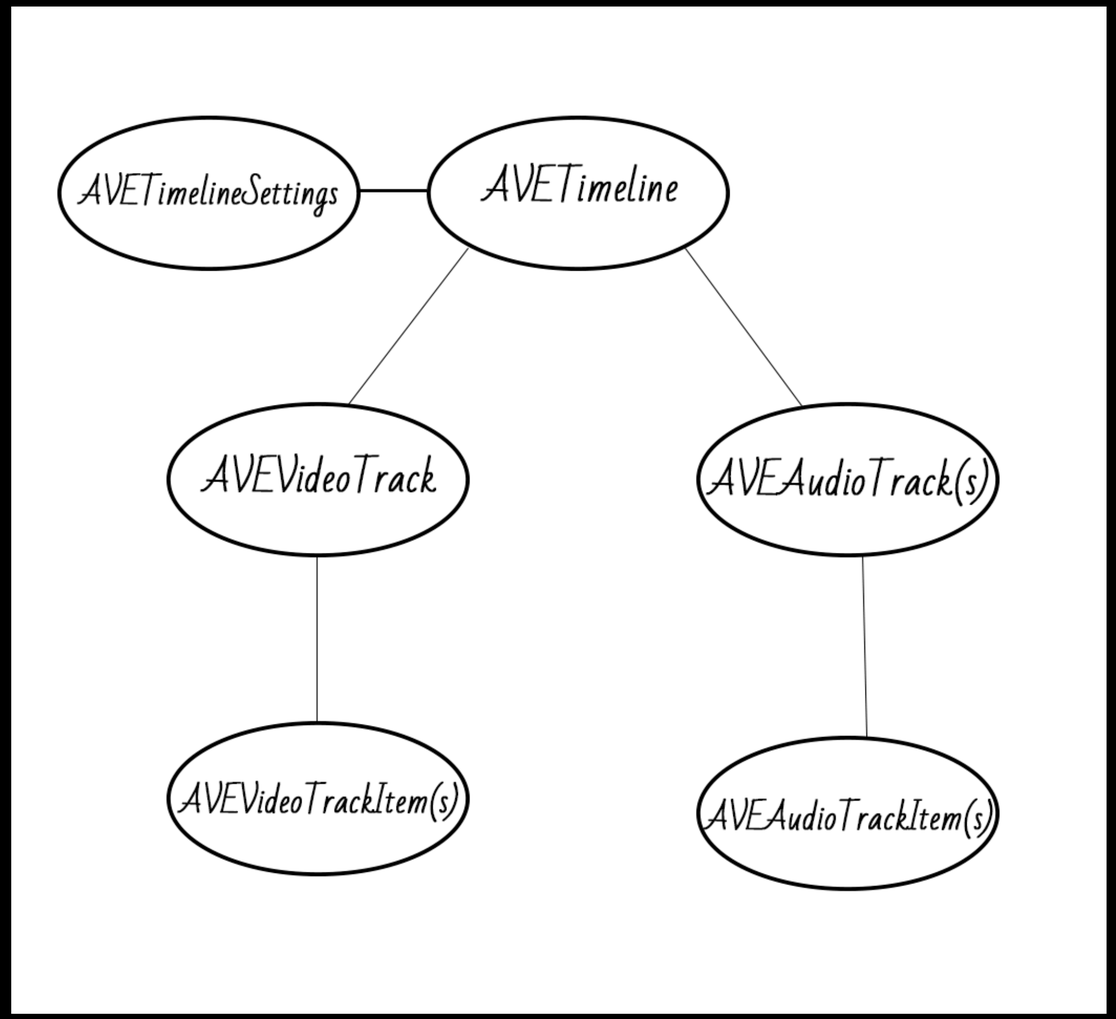
Зачем все это?

- AVFoundation - все-таки низкоуровневое API
- Отсутствуют реализации фич
- Легко сломать
- Оно не красивое

Core-часть расширенного редактора

Внутреннее устройство хранения данных

- Тайм-рендж
- Громкость
- Скорость
- Аудио-эффекты
- Фильтры
- .....



Что получили?

- Высокоуровневое API
- Легкое применение оптимизаций и фич по изменению видео (перемещение, дублирование, удаление, разделение)
- Любое редактирование кадров видео (наложение стикеров, фона, трансформации)
- Изменение громкости
- Реверс видео
- Аудио-эффекты
- Видео из фото
- Липсинк

Если уметь правильно работать, то боль утихает

Тема AVFoundation очень обширна

- Обработка аудио-семплов
- Построение гистограмм звука
- И многое другое

VK Клипы

Михаил Дементьев

@MikelDementyev