DEVIEW 2021

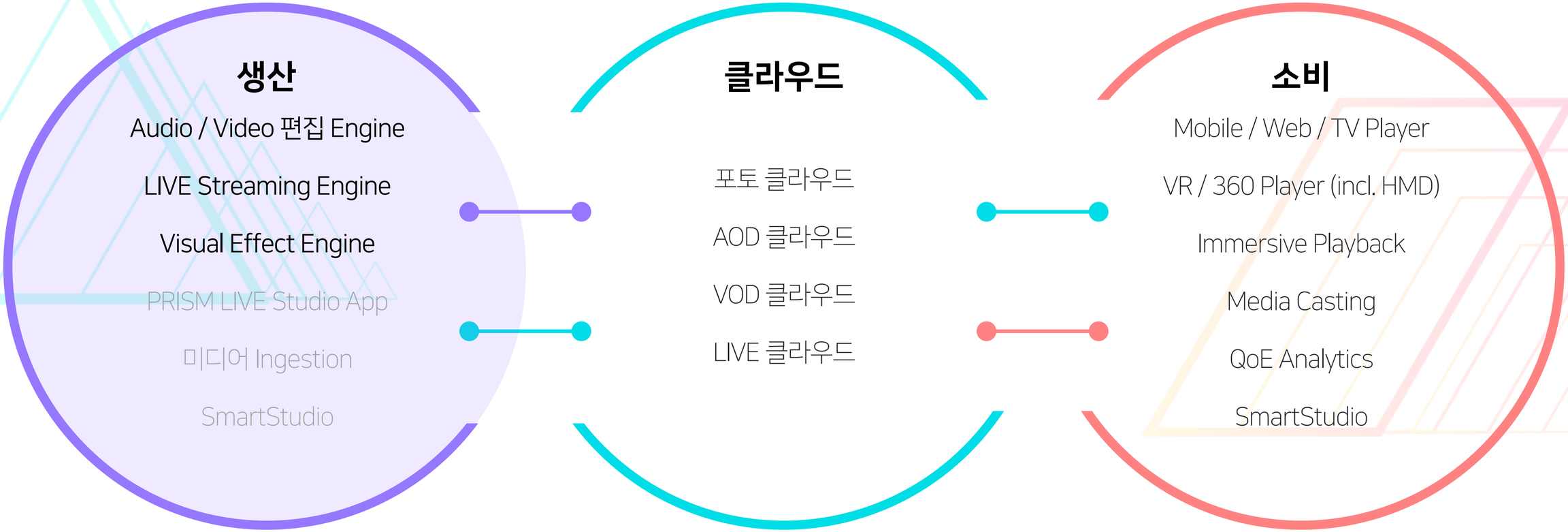
CoreImage와 Metal을 이용하여 영상에 다양한 효과 적용하기

이재현 NAVER ETECH

NAVER Emerging TECHnology

NAVER ETECH.

포토 / 오디오 / 비디오의 <생산 – 클라우드 – 소비> 워크플로의 전구간 기술 연구와 개발을 담당합니다.
글로벌 환경에서 시간 / 공간 / 용량의 제약 사항을 극복하고 생생한 현장 느낌과 안정적인 지원을 위해
이머징 기술 연구와 개발을 통한 원격의 시대를 준비하고 있습니다.



DEVIEW 2021

이재현

NAVER ETECH

라이브 송출 / 비디오 편집 / 비디오 필터 개발

iOS

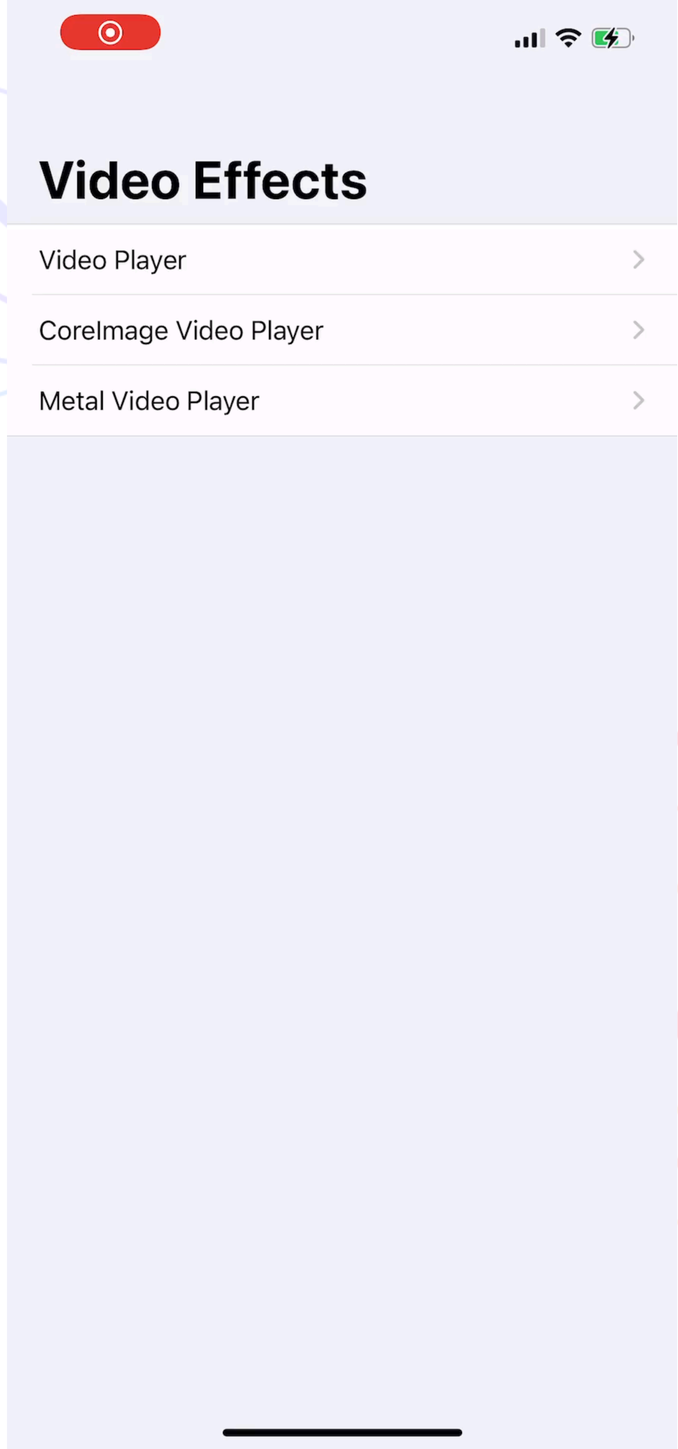
DEVIEW 2021

CONTENTS

1. Preview

1. Preview

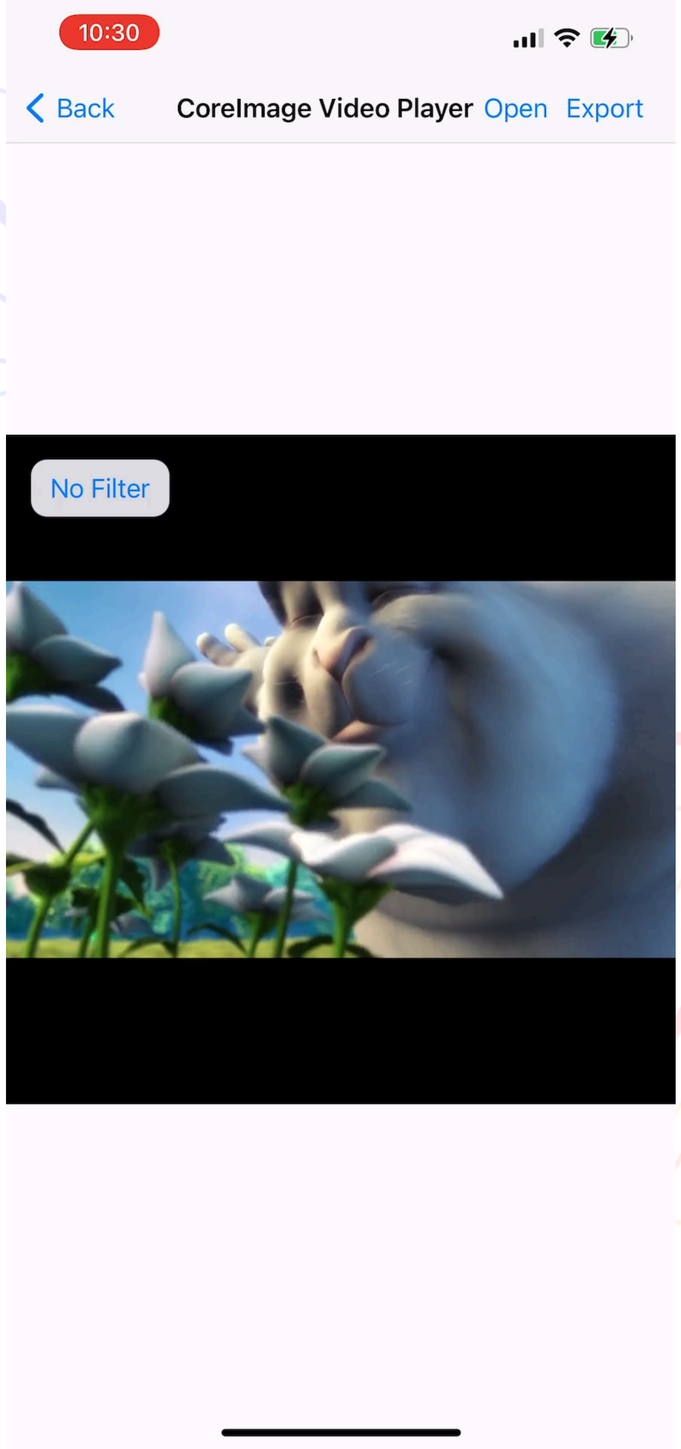
간단한 비디오 플레이어



1. Preview

간단한 비디오 플레이어

CoreImage 비디오 플레이어

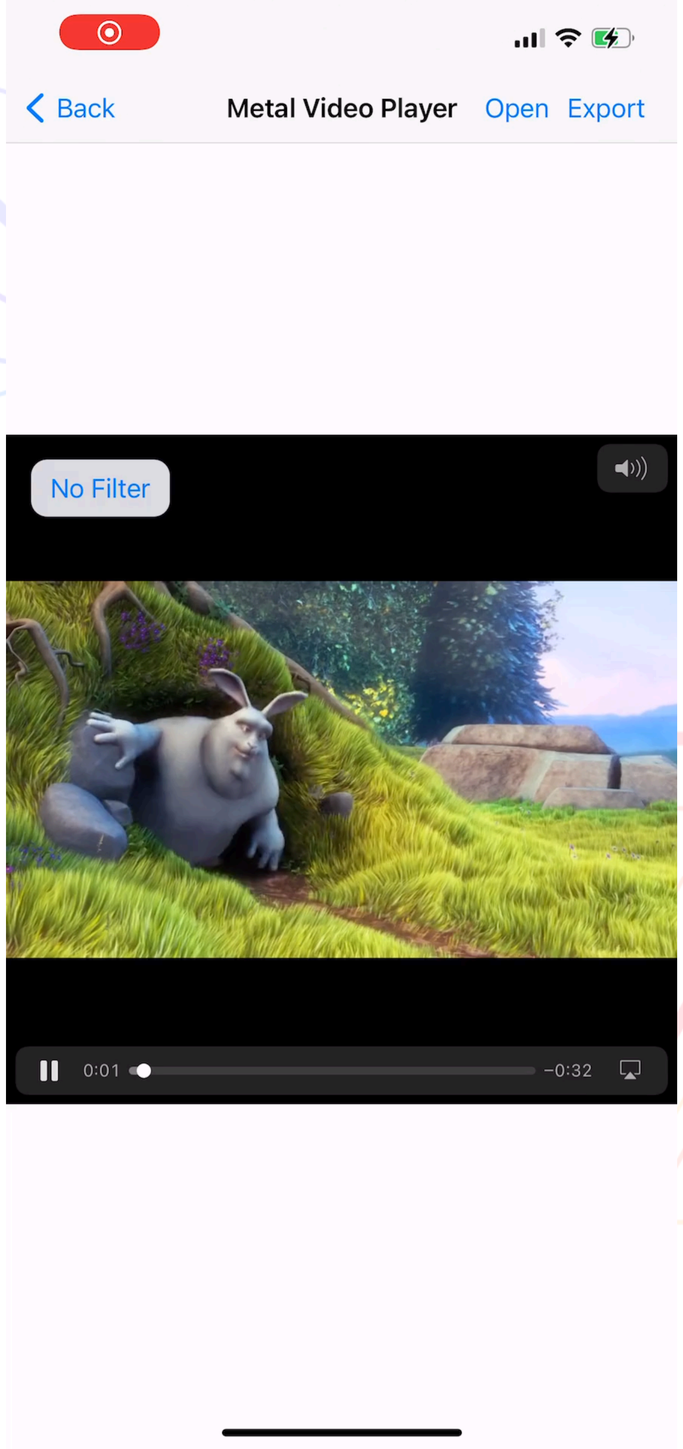


1. Preview

간단한 비디오 플레이어

CoreImage 비디오 플레이어

Metal 비디오 플레이어



1.1 오늘 다루지 않을 내용

SwiftUI

UIKit

Photos

1.2 실습 소스코드

https://github.com/johnlee92/DEVIEW2021_VideoEffects

2

간단한 비디오 플레이어를 만들어 봅시다

2.1

AVFoundation

2.1

AVFoundation

#Apple 미디어 프레임워크
#재생 #편집 #캡쳐 #AVKit

Class

AVAsset

An object that models timed audiovisual media.

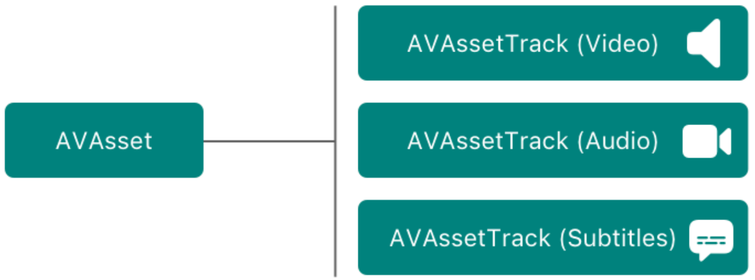
Declaration

```
class AVAsset : NSObject
```

Creating an Asset

`init(url: URL)`

Creates an asset that models the media at the specified URL.



https://developer.apple.com/documentation/avfoundation/avasset

Class

AVPlayer

An object that provides the interface to control the player's transport behavior.

Declaration

```
class AVPlayer : NSObject
```

Creating a Player

`init(url: URL)`

Creates a new player to play a single audiovisual resource referenced by a given URL.

`init(playerItem: AVPlayerItem?)`

Creates a new player to play the specified player item.

Managing the Player Item

`var currentItem: AVPlayerItem?`

The current player item.

`func replaceCurrentItem(with: AVPlayerItem?)`

Replaces the current item with a new item.

https://developer.apple.com/documentation/avfoundation/avplayer

Declaration

```
class AVPlayer : NSObject
```

Creating a Player

```
init(url: URL)
```

Creates a new player to play a single audiovisual resource referenced by a given URL.

```
init(playerItem: AVPlayerItem?)
```

Creates a new player to play the specified player item.

Managing the Player Item

```
var currentItem: AVPlayerItem?
```

The current player item.

```
func replaceCurrentItem(with: AVPlayerItem?)
```

Replaces the current item with a new item.

Controlling Playback

```
func play()
```

Begins playback of the current item.

```
func pause()
```

Pauses playback of the current item.

https://developer.apple.com/documentation/avfoundation/avplayer

AVPlayerItem

An object that models the timing and presentation state of an asset that a player object presents.

Declaration

```
class AVPlayerItem : NSObject
```

Creating a Player Item

`init(url: URL)`

Creates a player item with a specified URL.

`init(asset: AVAsset)`

Creates a player item for a specified asset.

https://developer.apple.com/documentation/avfoundation/avplayeritem

Generic Structure

VideoPlayer

A view that displays content from a player and a native user interface to control playback.

Declaration

```
struct VideoPlayer<VideoOverlay> where VideoOverlay : View
```

Creating a Video Player

```
init(player: AVPlayer?)
```

Creates a video-player user interface for the player object.

Available when `VideoOverlay` is `EmptyView`.

```
init(player: AVPlayer?, videoOverlay: () -> VideoOverlay)
```

Creates a video-player user interface for the player object.

https://developer.apple.com/documentation/avkit/videoplayer

2.2

비디오 불러오기

3

CoreImage 비디오 플레이어를 만들어 봅시다

3.1

CoreImage

3.1

CoreImage

#Apple 이미지 처리 프레임워크
#Built-In 필터 #사용하기 쉬운 API

CIImage

A representation of an image to be processed or produced by Core Image filters.

Declaration

```
class CIImage : NSObject
```

`init(cgImage: CGImage)`

Initializes an image object with a Quartz 2D image.

`init?(image: UIImage)`

Initializes an image object with the specified UIKit image object.

`init?(contentsOf: URL)`

Initializes an image object by reading an image from a URL.

`init(cvPixelBuffer: CVPixelBuffer)`

Initializes an image object from the contents of a Core Video pixel buffer.

`init?(data: Data)`

Initializes an image object with the supplied image data.

`init?(mtlTexture: MTLTexture, options: [CIImageOption : Any]?)`

Initializes an image object with data supplied by a Metal texture.

https://developer.apple.com/documentation/coreimage/ciimage

```
init?(image: UIImage)
```

Initializes an image object with the specified UIKit image object.

```
init?(contentsOf: URL)
```

Initializes an image object by reading an image from a URL.

```
init?(data: Data)
```

Initializes an image object with the supplied image data.

```
init?(mtlTexture: MTLTexture, options: [CIImageOption : Any]?)
```

Initializes an image object with data supplied by a Metal texture.

Creating an Image by Modifying an Existing Image

```
func applyingFilter(String, parameters: [String : Any]) -> CIImage
```

Returns a new image created by applying a filter to the original image with the specified name and parameters.

```
func applyingFilter(String) -> CIImage
```

Applies the filter to an image and returns the output.

```
func transformed(by: CGAffineTransform) -> CIImage
```

Returns a new image that represents the original image after applying an affine transform.

```
func cropped(to: CGRect) -> CIImage
```

Returns a new image with a cropped portion of the original image.

https://developer.apple.com/documentation/coreimage/ciimage

CIFilter

An image processor that produces an image by manipulating one or more input images or by generating new image data.

Declaration

```
class CIFilter : NSObject
```

Creating a Filter

`init?(name: String)`

Creates a `CIFilter` object for a specific kind of filter.

`init?(name: String, parameters: [String : Any]?)`

Creates a `CIFilter` object for a specific kind of filter and initializes the input values.

Getting Filter Parameters and Attributes

`var attributes: [String : Any]`

A dictionary of key-value pairs that describe the filter.

`var outputImage: CIImage?`

Returns a `CIImage` object that encapsulates the operations configured in the filter.

https://developer.apple.com/documentation/coreimage/cifilter

To use the type-safe API, import `CoreImage.CIFilterBuiltins`:

```
import CoreImage
import CoreImage.CIFilterBuiltins
```

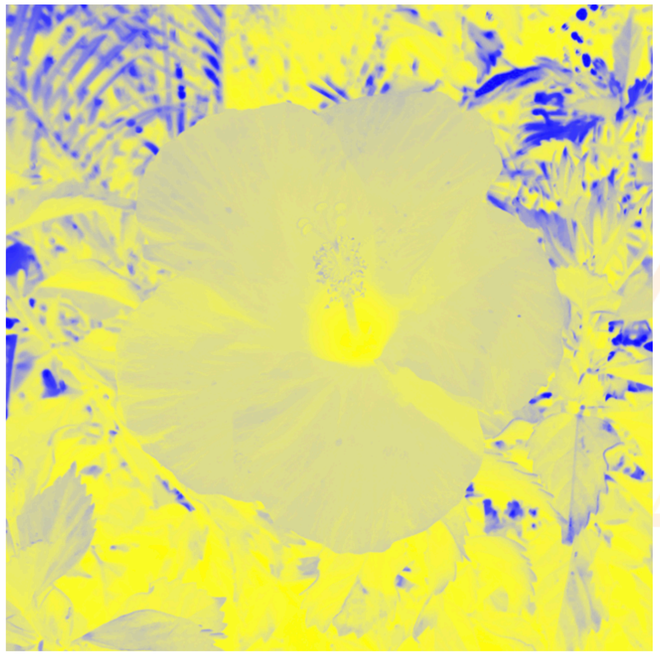
To use the type-safe API, import `CoreImage.CIFilterBuiltins`:

```
import CoreImage
import CoreImage.CIFilterBuiltins
```

```
func falseColor(inputImage: CIImage) -> CIImage? {

    let falseColorFilter = CIFilter.falseColor()

    falseColorFilter.color0 = CIColor(red: 1, green: 1, blue: 0)
    falseColorFilter.color1 = CIColor(red: 0, green: 0, blue: 1)
    falseColorFilter.inputImage = inputImage

    return falseColorFilter.outputImage
}
```

```
    falseColorFilter.color0 = CIColor(red: 1, green: 1, blue: 0)
    falseColorFilter.color1 = CIColor(red: 0, green: 0, blue: 1)
    falseColorFilter.inputImage = inputImage

    return falseColorFilter.outputImage
}
```

The false color filter maps luminance to a color ramp of two colors:

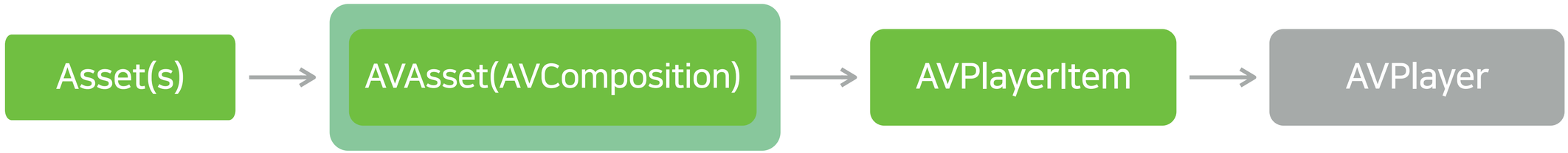


3.2

Video Processing with CoreImage

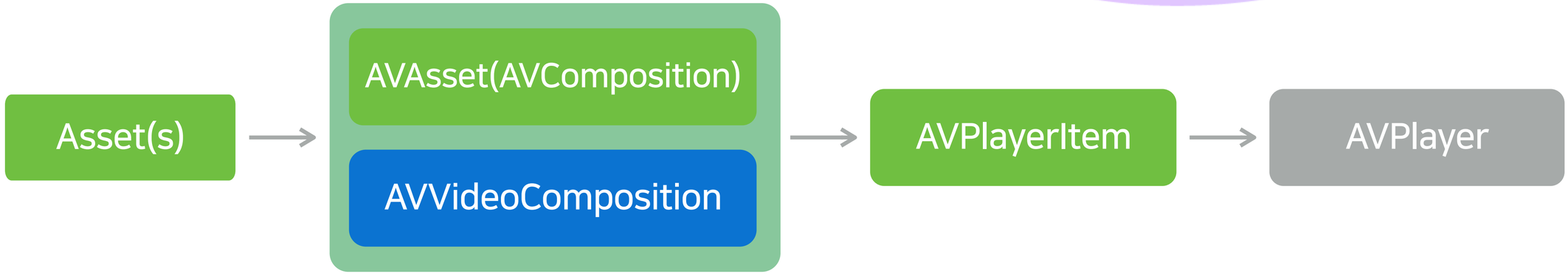
3.2 Video Processing with CoreImage

Video processing workflow



3.2 Video Processing with CoreImage

Video processing workflow



AVVideoComposition

An object that represents an immutable video composition.

Declaration

```
class AVVideoComposition : NSObject
```

Creating a Video Composition Object

`init(propertiesOf: AVAsset)`

Creates a video composition object configured to present the video tracks of the specified asset.

`init(asset: AVAsset, applyingCIFiltersWithHandler: (AVAsynchronous CIImageFilteringRequest) -> Void)`

Creates a video composition configured to apply Core Image filters to each video frame of the specified asset.

`class AVAsynchronousCIImageFilteringRequest`

An object that supprts using Core Image filters to process an individual video frame in a video composition.

Class

AVVideoComposition

An object that represents an immutable video composition.

Declaration

```
class AVVideoComposition : NSObject
```

Class

AVPlayerItem

An object that models the timing and presentation state of an asset that a player object presents.

Configuring Video Compositing

`var videoComposition: AVVideoComposition?`

The video composition settings to be applied during playback.

Class

AVAssetExportSession

An object that transcodes the contents of an asset in a format that you specify using an export preset.

Configuring Video Output

`var videoComposition: AVVideoComposition?`

An optional object that provides instructions for how to composite frames of video.

https://developer.apple.com/documentation/avfoundation/avvideocomposition

3.3

비디오 내보내기

Class

AVAssetExportSession

An object that transcodes the contents of an asset in a format that you specify using an export preset.

Declaration

```
class AVAssetExportSession : NSObject
```

Creating an Export Session

```
init?(asset: AVAsset, presetName: String)
```

Creates an asset export session that uses the specified preset.

Configuring Output

```
var outputURL: URL?
```

A URL where an asset export session writes its output.

```
var outputFileType: AVFileType?
```

The file type of the output an asset export session writes.

Creating an Export Session

```
init?(asset: AVAsset, presetName: String)
```

Creates an asset export session that uses the specified preset.

Configuring Output

```
var outputURL: URL?
```

A URL where an asset export session writes its output.

```
var outputFileType: AVFileType?
```

The file type of the output an asset export session writes.

Configuring Video Output

```
var videoComposition: AVVideoComposition?
```

An optional object that provides instructions for how to composite frames of video.

Exporting Media

```
func exportAsynchronously(completionHandler: () -> Void)
```

Starts the asynchronous execution of an export session.

```
func cancelExport()
```

Cancels the execution of an export session.

Determining Export Status

```
var progress: Float
```

A value that indicates the progress of the export.

https://developer.apple.com/documentation/avfoundation/avassetexportsession

4

Metal 비디오 플레이어를 만들어 봅시다

4.1

Metal

4.1

Metal

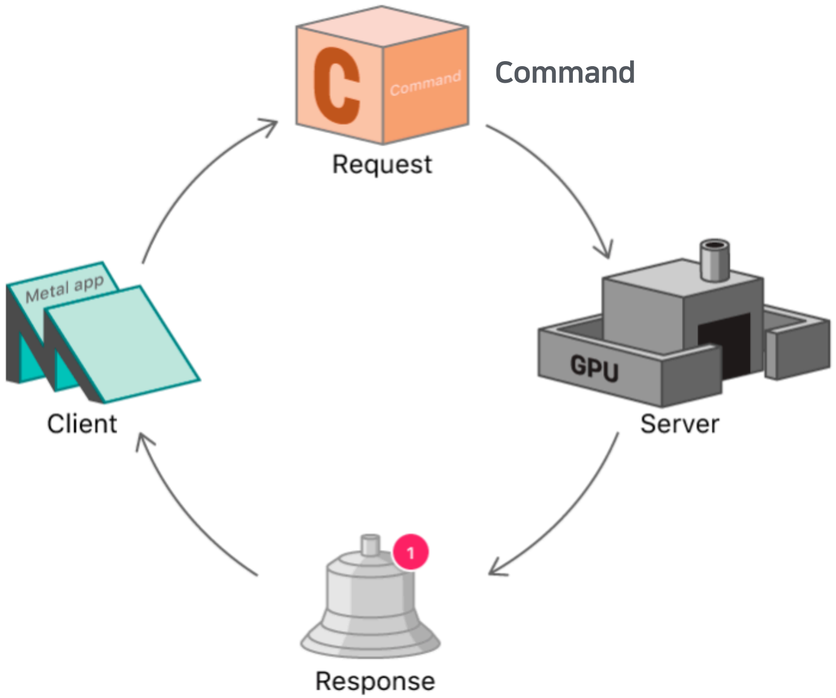
#Apple low-level 그래픽스 API
#작성할 것이 많음 #강력한 성능 #확장성

4.1 Metal

	CoreImage	Metal
이미지 처리	O	O
CPU 렌더링	O	X
3D 렌더링	X	O
SpriteKit / SceneKit 지원	X	O
Metal Performance Shader	X	O

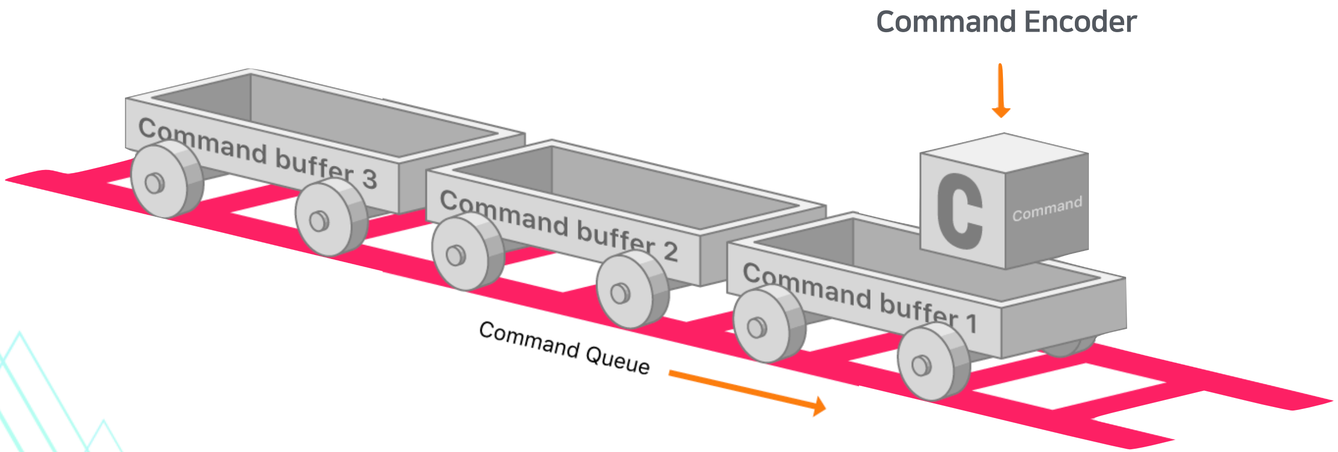
4.1 Metal

Metal workflow



4.1 Metal

Metal workflow



4.1 Metal

Protocol	Task
`MTLRenderCommandEncoder`	Graphics rendering
`MTLComputeCommandEncoder`	Computation
`MTLBlitCommandEncoder`	Memory management
`MTLParallelRenderCommandEncoder`	Multiple graphics rendering tasks encoded in parallel.

4.1 Metal

Protocol	Task
`MTLRenderCommandEncoder`	Graphics rendering
`MTLComputeCommandEncoder`	Computation
`MTLBlitCommandEncoder`	Memory management
`MTLParallelRenderCommandEncoder`	Multiple graphics rendering tasks encoded in parallel.

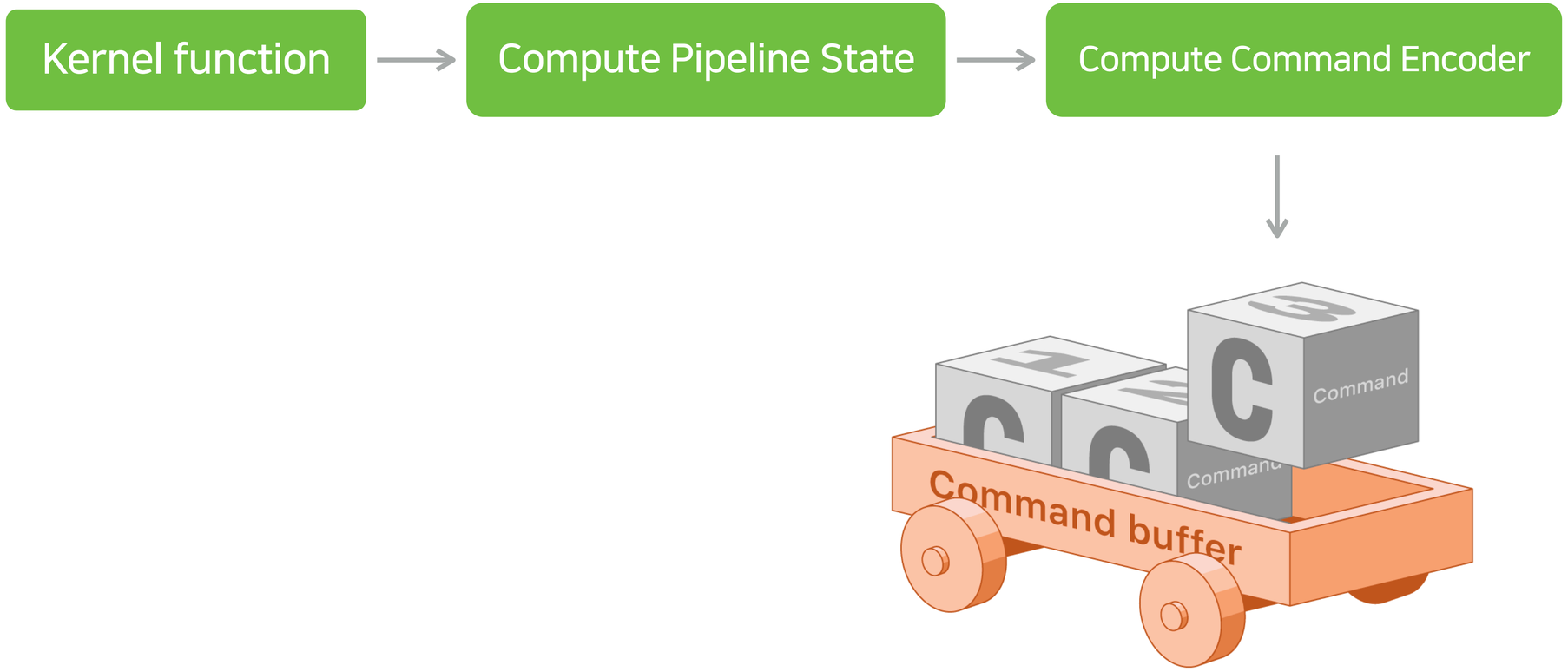
4.1 Metal

Compute programming workflow

MSL

Executable

+ Textures, Arguments



MTLDevice

The Metal interface to a GPU that you use to draw graphics or do parallel computation.

Acquiring Device Objects

```
func MTLCreateSystemDefaultDevice() -> MTLDevice?
```

Returns a reference to the preferred default Metal device object.

Creating Command Queues

```
func makeCommandQueue() -> MTLCommandQueue?
```

Creates a command submission queue.

Required.

Creating Shader Libraries

```
func makeDefaultLibrary() -> MTLLibrary?
```

Creates a library object that contains the functions in the app's default Metal library.

Required.

Creating Compute Pipelines

```
func makeComputePipelineState(function: MTLFunction) -> MTLCompute
PipelineState
```

Synchronously creates a new compute pipeline state object using a function object.

Required.

https://developer.apple.com/documentation/metal/mtldevice

MTLLibrary

A collection of Metal shader functions.

Declaration

```
protocol MTLLibrary
```

Creating Shader Function Objects

```
func makeFunction(name: String) -> MTLFunction?
```

Creates an object that represents a shader function in the library.

Required.

https://developer.apple.com/documentation/metal/mtllibrary

MTLCommandQueue

A queue that organizes command buffers for the GPU to execute.

Declaration

```
protocol MTLCommandQueue
```

Creating Command Buffers

```
func makeCommandBuffer() -> MTLCommandBuffer?
```

Creates a command buffer.

Required.

Protocol

MTLComputeCommandEncoder

An object for encoding commands in a compute pass.

Declaration

```
protocol MTLComputeCommandEncoder
```

Specifying the Compute Pipeline State

```
func setComputePipelineState(MTLComputePipelineState)
```

Sets the current compute pipeline state object.

Required.

Specifying Arguments for a Compute Function

```
func setBuffer(MTLBuffer?, offset: Int, index: Int)
```

Sets a buffer for the compute function.

Required.

```
func setBytes(UnsafeRawPointer, length: Int, index: Int)
```

Sets a block of data for the compute shader.

Required.

```
func setTexture(MTLTexture?, index: Int)
```

Sets a texture for the compute function.

Required.

https://developer.apple.com/documentation/metal/mtlcomputecommandencoder

Declaration

```
protocol MTLComputeCommandEncoder
```

Specifying the Compute Pipeline State

```
func setComputePipelineState(MTLComputePipelineState)
```

Sets the current compute pipeline state object.

Required.

Specifying Arguments for a Compute Function

```
func setBuffer(MTLBuffer?, offset: Int, index: Int)
```

Sets a buffer for the compute function.

Required.

```
func setBytes(UnsafeRawPointer, length: Int, index: Int)
```

Sets a block of data for the compute shader.

Required.

```
func setTexture(MTLTexture?, index: Int)
```

Sets a texture for the compute function.

Required.

Executing a Compute Function Directly

```
func dispatchThreadgroups(MTLSize, threadsPerThreadgroup: MTLSize)
```

Encodes a compute command using a grid aligned to threadgroup boundaries.

Required.

```
func dispatchThreads(MTLSize, threadsPerThreadgroup: MTLSize)
```

Encodes a compute command using an arbitrarily sized grid.

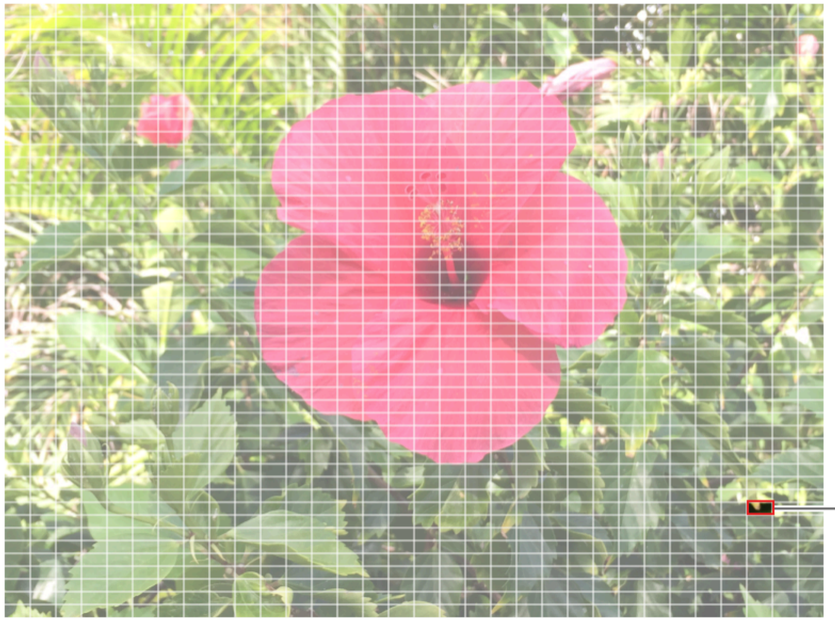
Required.

https://developer.apple.com/documentation/metal/mtlcomputecommandencoder

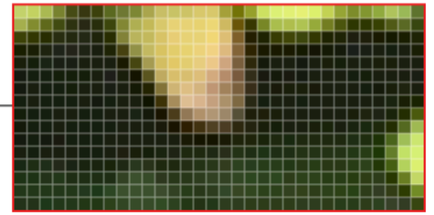
4.1 Metal

Calculating The number of threads in one threadgroup

Grid: 1024 x 768 pixels / 32 x 48 threadgroups



Threadgroup: 32 x 16 threads



4.1 Metal

Calculating The number of threads in one threadgroup

MTLComputePipelineState

↓

maxTotalThreadsPerThreadgroup

512

threadExecutionWidth

32

↓

32 x (512 / 32) = 32 x 16

4.1 Metal

```
func dispatchThreadgroups(MTLSize, threadsPerThreadgroup: MTLSize)
```

Encodes a compute command using a grid aligned to threadgroup boundaries.

Required.

```
func dispatchThreads(MTLSize, threadsPerThreadgroup: MTLSize)
```

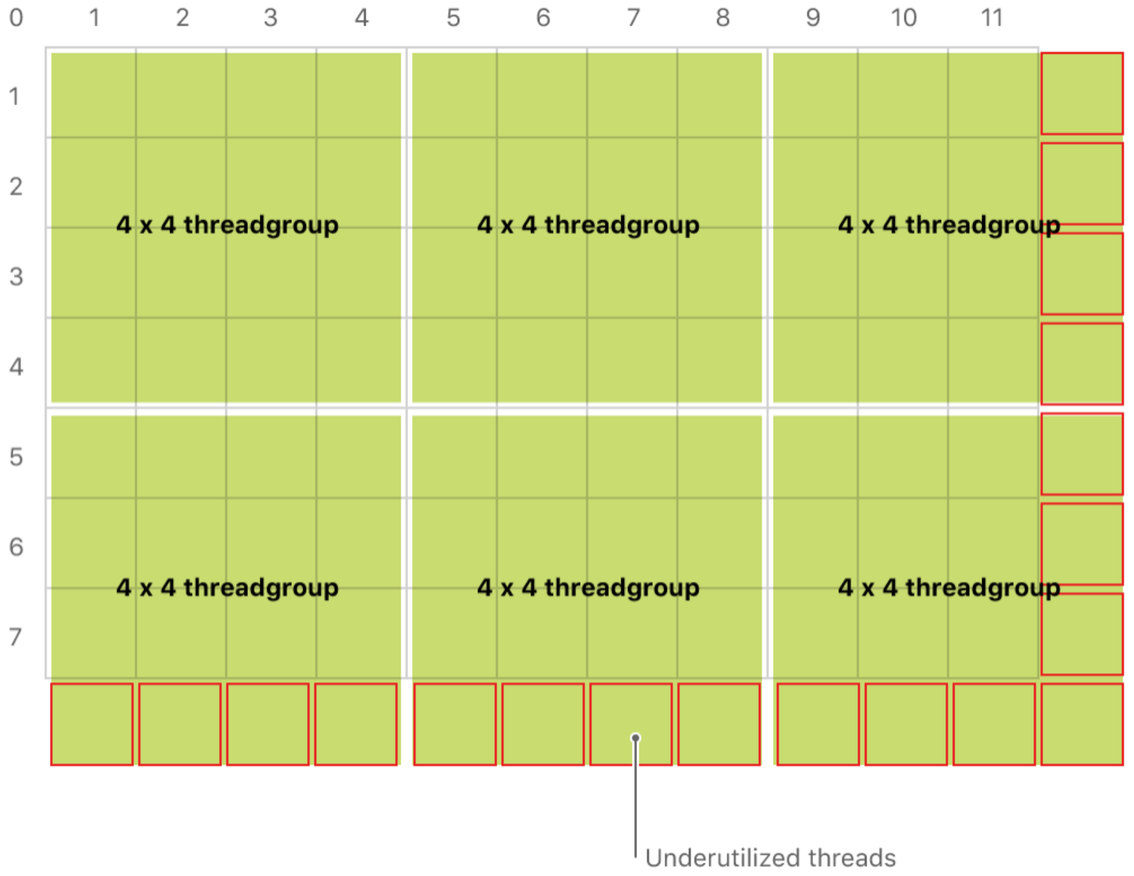
Encodes a compute command using an arbitrarily sized grid.

Required.

```
func dispatchThreadgroups(MTLSize, threadsPerThreadgroup: MTLSize)
```

Encodes a compute command using a grid aligned to threadgroup boundaries.

Required.



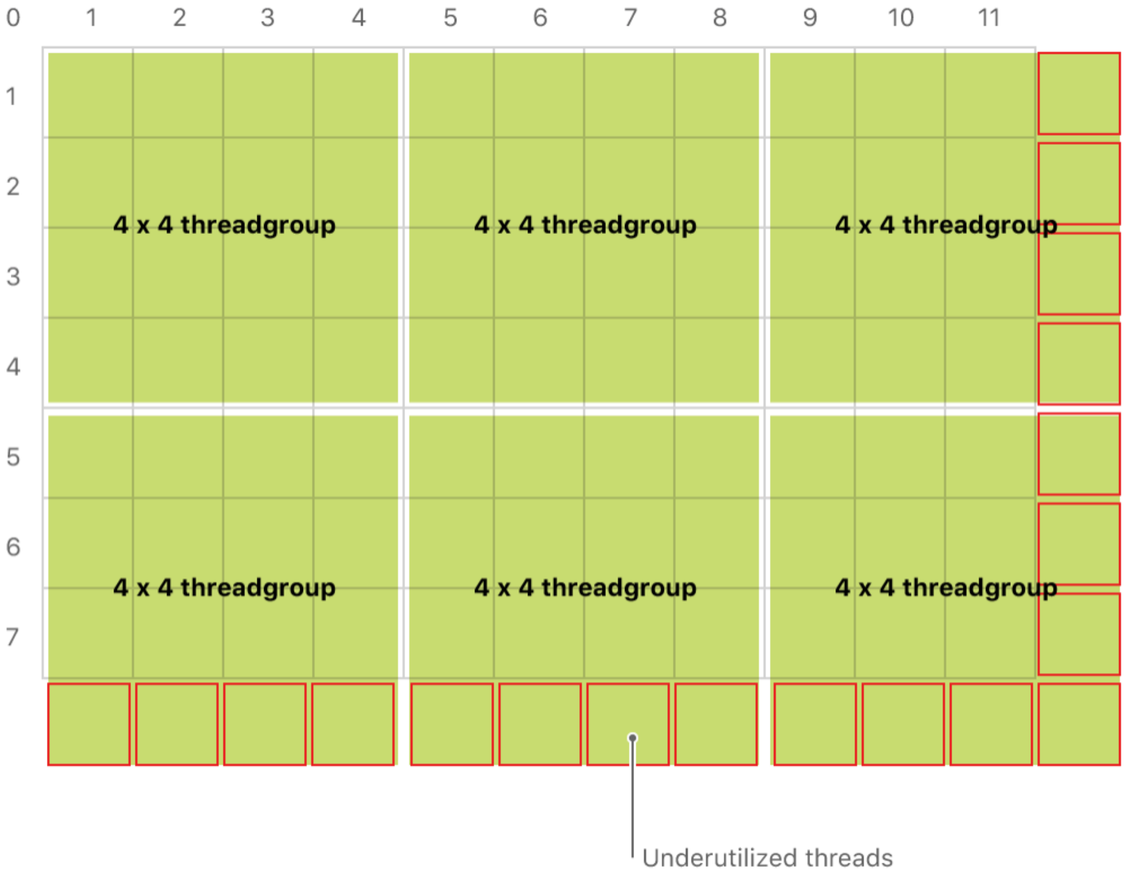
https://developer.apple.com/documentation/metal/mtlcomputecommandencoder

```
func dispatchThreadgroups(MTLSize, threadsPerThreadgroup: MTLSize)
```

Encodes a compute command using a grid aligned to threadgroup boundaries.

Required.

3x2

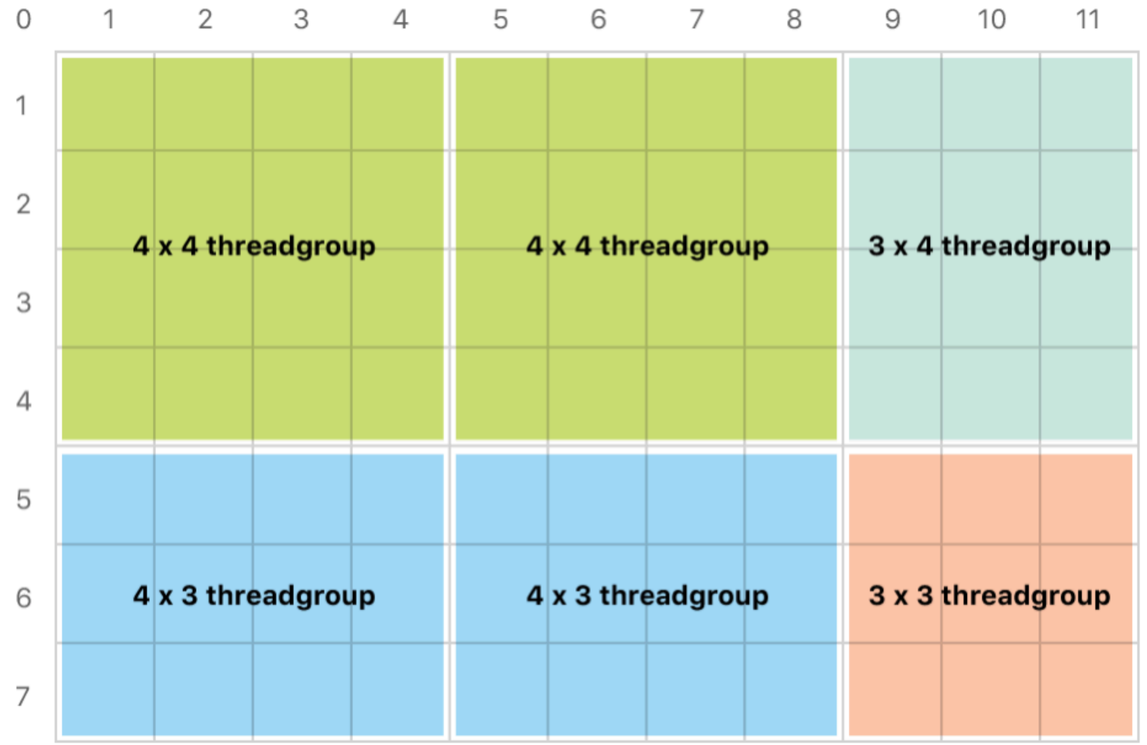


https://developer.apple.com/documentation/metal/mtlcomputecommandencoder

```
func dispatchThreads(MTLSize, threadsPerThreadgroup: MTLSize)
```

Encodes a compute command using an arbitrarily sized grid.

Required.



https://developer.apple.com/documentation/metal/mtlcomputecommandencoder

DEVIEW 2021

4.2

Video Processing with Metal

4.2 Video Processing with Metal

Metal로 Video Processing하려면
비디오 프레임을 나타내는 MTLTexture 필요

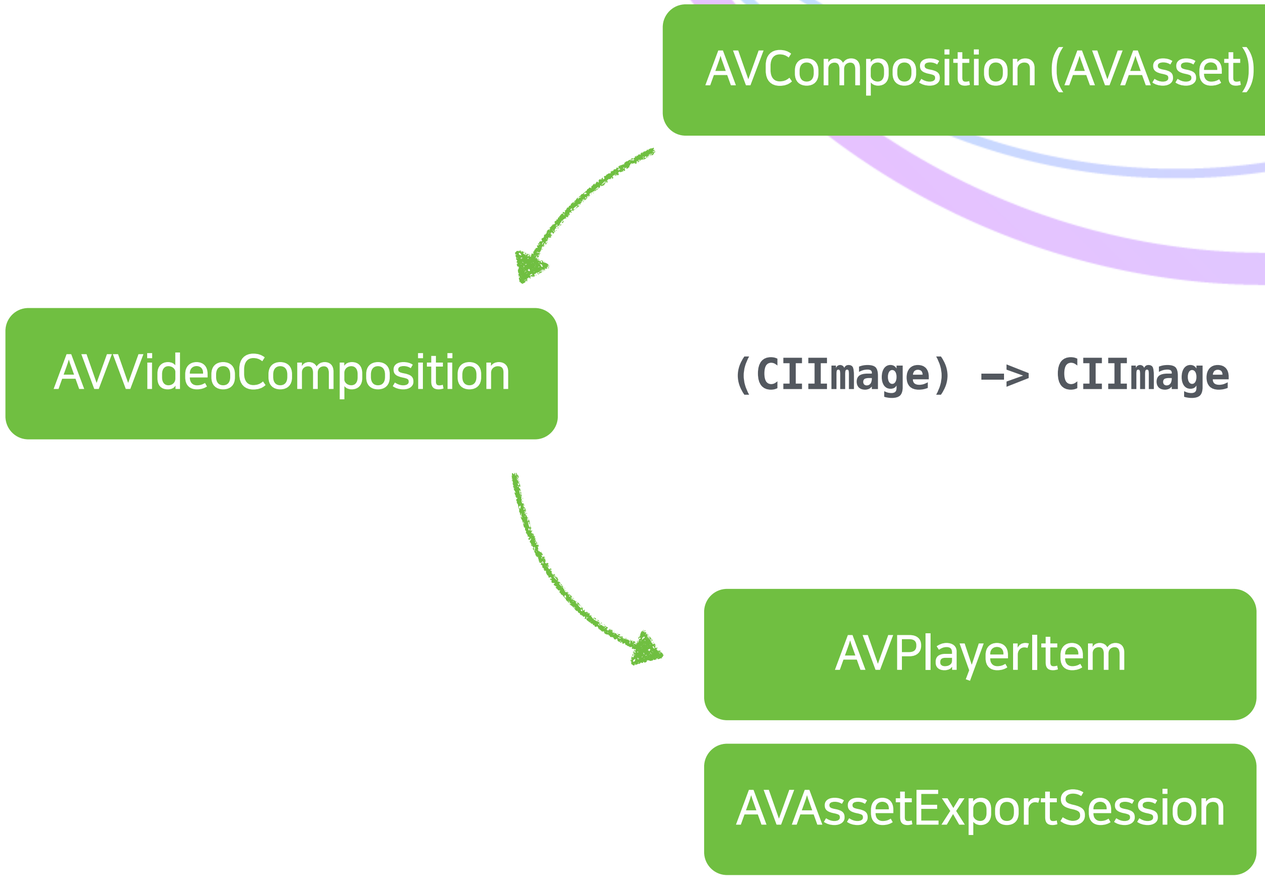
4.2 Video Processing with Metal

AVFoundation에서 **MTLTexture**를 얻을 수는 없고,
CVPixelBuffer를 이용해서 **MTLTexture**를 생성해야 함

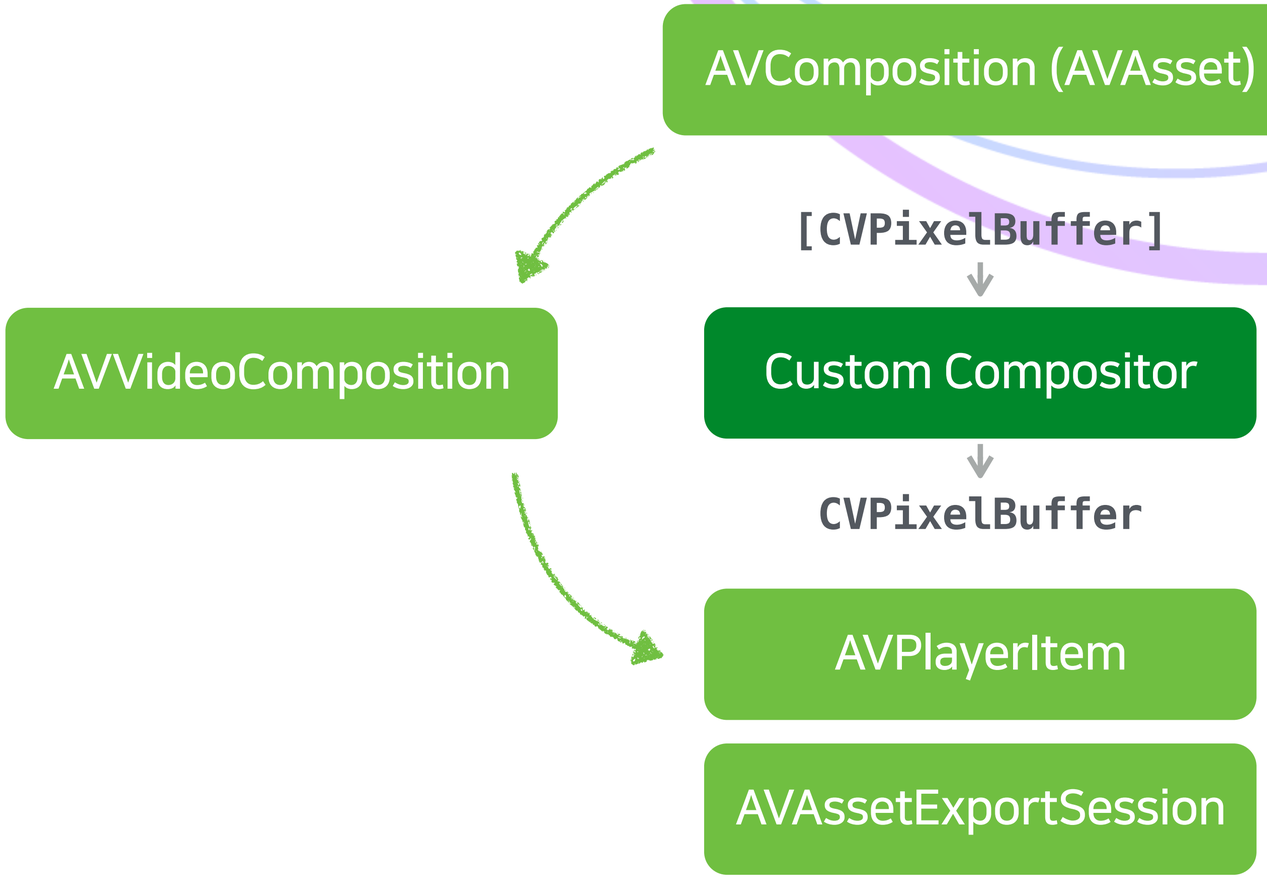
4.2 Video Processing with Metal

그럼
CVPixelBuffer는 어떻게..?

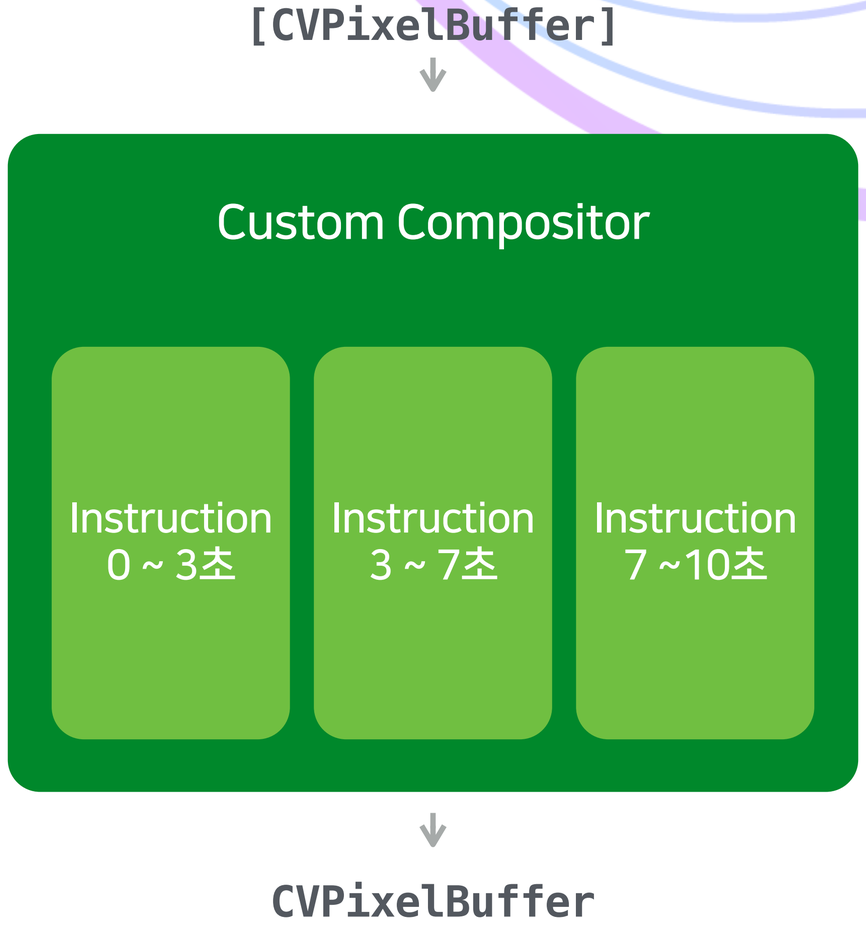
4.2 Video Processing with Metal



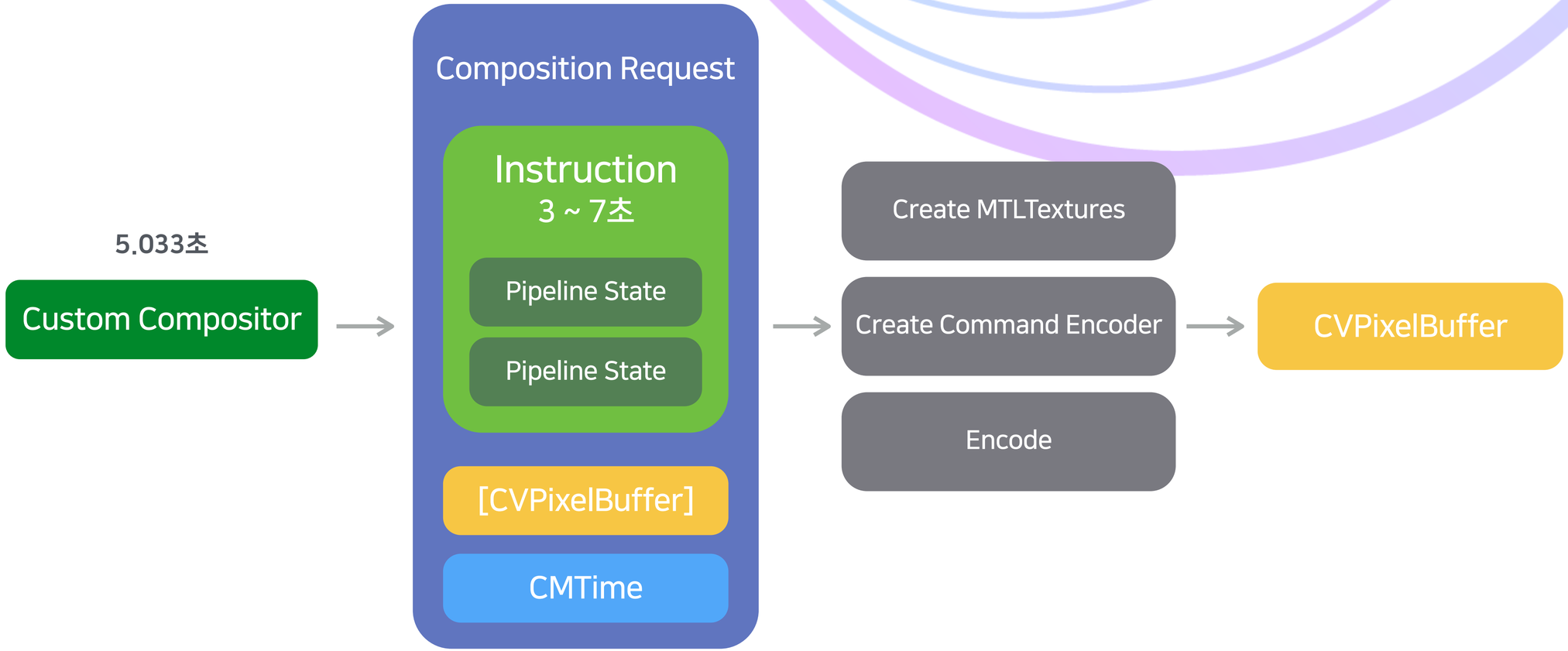
4.2 Video Processing with Metal



4.2 Video Processing with Metal



4.2 Video Processing with Metal



Protocol

AVVideoCompositing

A protocol that defines the methods custom video compositors must implement.

Declaration

```
protocol AVVideoCompositing
```

Inspecting Processing Requirements

`var sourcePixelBufferAttributes: [String : Any]?`

The pixel buffer attributes that the compositor accepts for source frames.

Required.

`var requiredPixelBufferAttributesForRenderContext: [String : Any]`

The pixel buffer attributes that the compositor requires for pixel buffers that it creates.

Required.

Observing Render Context Changes

`func renderContextChanged(AVVideoCompositionRenderContext)`

Tells the compositor that the composition changed render contexts.

Required.

https://developer.apple.com/documentation/avfoundation/avvideocompositing

Declaration

```
protocol AVVideoCompositing
```

Inspecting Processing Requirements

`var sourcePixelBufferAttributes: [String : Any]?`

The pixel buffer attributes that the compositor accepts for source frames.

Required.

`var requiredPixelBufferAttributesForRenderContext: [String : Any]`

The pixel buffer attributes that the compositor requires for pixel buffers that it creates.

Required.

Observing Render Context Changes

`func renderContextChanged(AVVideoCompositionRenderContext)`

Tells the compositor that the composition changed render contexts.

Required.

Rendering the Composition

`func startRequest(AVAsynchronousVideoCompositionRequest)`

Directs a custom video compositor object to create a new pixel buffer composed asynchronously from a collection of sources.

Required.

`func cancelAllPendingVideoCompositionRequests()`

Directs a custom video compositor object to cancel or finish all pending video composition requests.

https://developer.apple.com/documentation/avfoundation/avvideocompositing

Class

AVAsynchronousVideoCompositionRequest

An object that contains information a video compositor needs to render an output pixel buffer.

Declaration

```
class AVAsynchronousVideoCompositionRequest : NSObject
```

Accessing Source Data

```
var sourceTrackIDs: [NSNumber]
```

The identifiers of tracks that contain source video.

```
func sourceFrame(byTrackID: CMPersistentTrackID) -> CVPixelBuffer?
```

Returns a source pixel buffer for the track that contains the specified identifier.

Finishing the Request

```
func finish(withComposedViceoFrame: CVPixelBuffer)
```

Finishes the request to compose the frame.

```
func finish(with: Error)
```

Finishes the request with an error.

```
func finishCancelledRequest()
```

Cancels the request to compose a video frame.

https://developer.apple.com/documentation/avfoundation/avasynchronousvideocompositionrequest

AVVideoCompositionInstructionProtocol

A protocol that defines the interface for a video composition instruction.

Getting Track ID Settings

```
var passthroughTrackID: CMPersistentTrackID
```

The track identifier of the video composition when a single source frame should be displayed for the duration of the instruction.

Required.

```
var requiredSourceTrackIDs: [NSValue]?
```

The video track identifiers required to compose frames for this instruction.

Required.

Getting Timing Settings

```
var timeRange: CMTimeRange
```

The time range during which the instruction is effective.

Required.

https://developer.apple.com/documentation/avfoundation/avvideocompositioninstructionprotocol

4.3 Summary

1. 비디오 재생과 편집은 **AVPlayer**와 **AVVideoComposition**
2. 간단한 이미지 필터는 **CoreImage Built-In** 필터
3. 심화된 이미지 필터는 **Metal Shader**
4. Metal을 비디오 필터로 사용할 때 **AVVideoCompositing**

NAVER
Emerging
TECHnology

함께 성장하실 동료분을 모십니다

FE / BE / 앱 / SDK 개발 기술과 함께 멀티미디어의 핵심 기술을 배우고,

포토 / 오디오 / 비디오 / UGC 기술 도메인 전문가로 성장할 수 있도록 적극 지원하겠습니다.

ETECH 직무 소개

- **ETECH 직무 소개** : https://naver-career.gitbook.io/kr/service/etech
- **ETECH 기술 문의** : etech@navercorp.com
- **ETECH 채용 문의** : etech-recruit@navercorp.com

Thank You