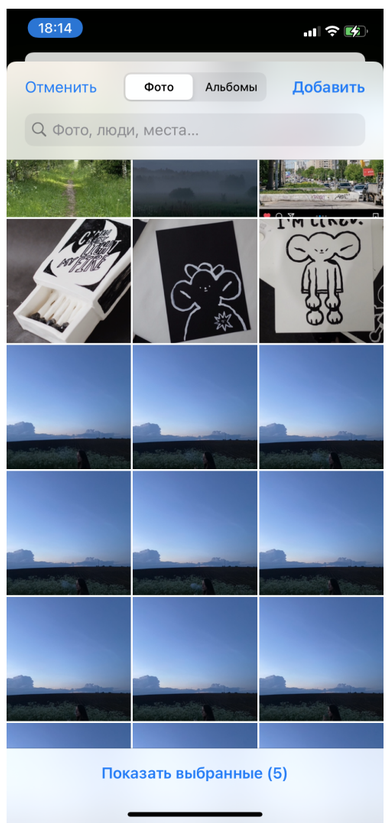
~~UIImagePickerController~~

PHPickerController

developer.apple.com

Apple Developer Documentation




```
import PhotosUI
```

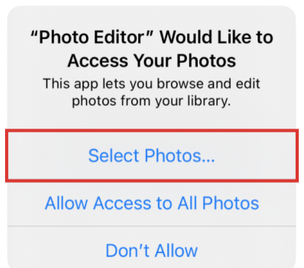
```
var newFilter = PHPickerFilter.any(of: [.livePhotos, .videos])
```

```
41      /// - Tag: PresentPicker
42      private func presentPicker(filter: PHPickerFilter?) {
43          var configuration = PHPickerConfiguration(photoLibrary: .shared())
44
45          // Set the filter type according to the user's selection.
46          configuration.filter = filter
47          // Set the mode to avoid transcoding, if possible, if your app supports arbitrary image/video encodings.
48          configuration.preferredAssetRepresentationMode = .current
49          // Set the selection behavior to respect the user's selection order.
50          configuration.selection = .ordered                                  2  Cannot infer contextual base
51          // Set the selection limit to enable multiselection.
52          configuration.selectionLimit = 0
53          // Set the preselected asset identifiers with the identifiers that the app tracks.
54          configuration.preselectedAssetIdentifiers = selectedAssetIdentifiers  Value of type 'PHPickerConfiguration' has no me
55
56          let picker = PHPickerViewController(configuration: configuration)
57          picker.delegate = self
58          present(picker, animated: true)
59      }
```

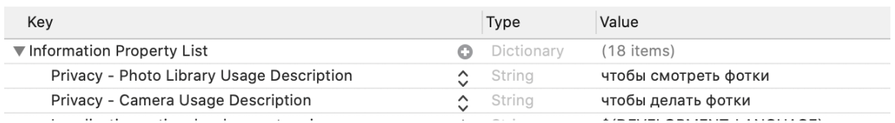
```
197
198 extension ViewController: PHPickerViewControllerDelegate {
199     func picker(_ picker: PHPickerViewController, didFinishPicking results: [PHPickerResult]) {
200         dismiss(animated: true)
201
202     }
```

Приватность

PHAuthorizationStatus



для UIImagePickerController примерно так же, но надо писать эти тексты в info.plist




```swift
// Check the app's authorization status (either read/write or add-only access).
let readWriteStatus = PHPhotoLibrary.authorizationStatus(for: .readWrite)
```

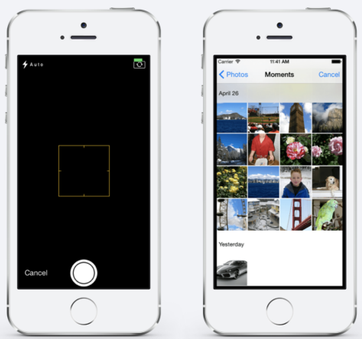
```swift
// Request read-write access to the user's photo library.
PHPhotoLibrary.requestAuthorization(for: .readWrite) { status in
    switch status {
    case .notDetermined:
        // The user hasn't determined this app's access.
    case .restricted:
        // The system restricted this app's access.
    case .denied:
        // The user explicitly denied this app's access.
    case .authorized:
        // The user authorized this app to access Photos data.
    case .limited:
        // The user authorized this app for limited Photos access.
    @unknown default:
        fatalError()
    }
}
```

Как это было раньше...

UIImagePickerController

```
class UIImagePickerController : UINavigationController
```

ViewController для работы с камерой и галерей



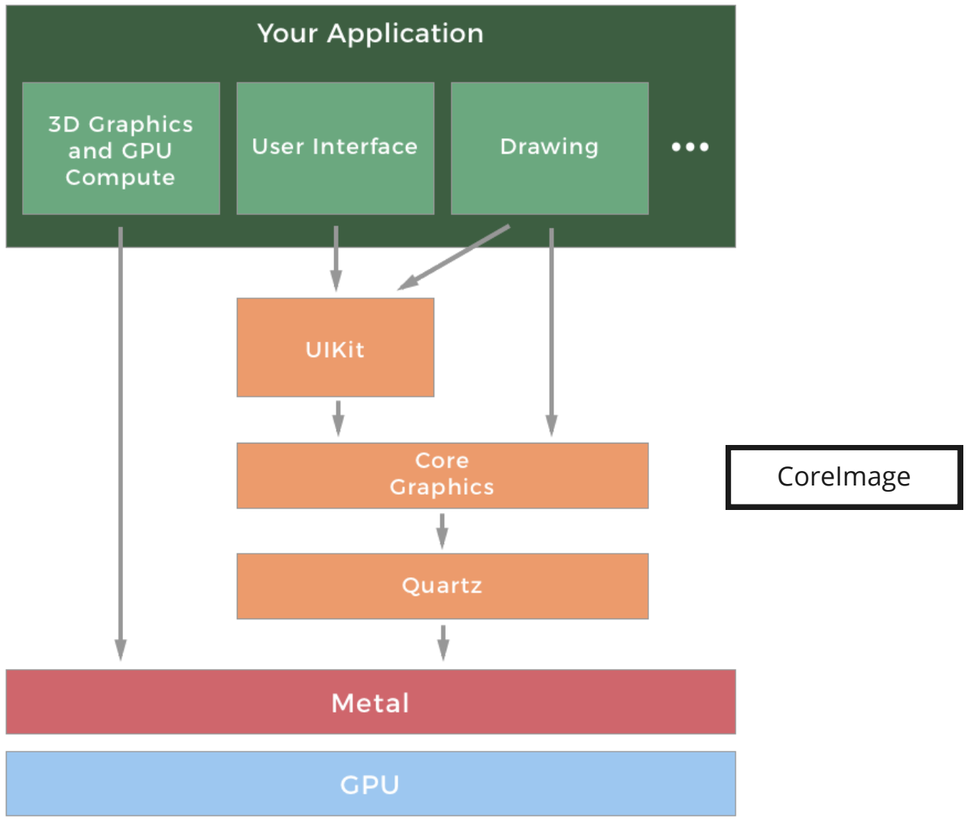
Camera Overlay View

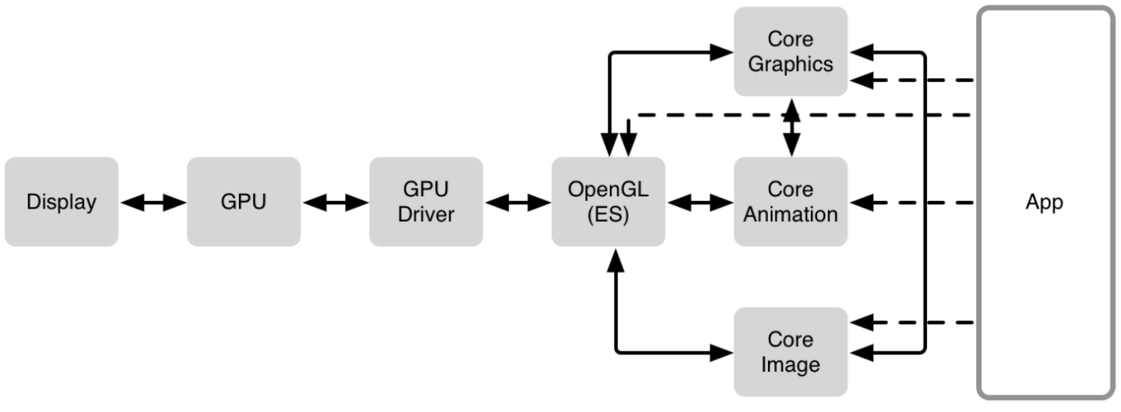
```
@property(nonatomic, strong) __kindof UIView *cameraOverlayView;
```

Позволяет создавать кастомные интерфейс камеры без использования, например, AVFoundation

К этому свойству можно обращаться только в случает, если SourceType у нас задан в **UIImagePickerControllerSourceTypeCamera**. В противном случае мы схвати крэш NSInvalidArgumentException

Camera Overlay View





Just above the display sits the GPU, the *graphics processing unit* . The GPU is a highly concurrent processing unit, which is tailored specifically for parallel computation of graphics. That's how it's possible to update all those pixels and

UIGraphicsRenderer

developer.apple.com

Apple Developer Documentation

Нарисовать квадраты

```swift
func drawImage() {
    let renderer = UIGraphicsImageRenderer(size: CGSize(width: 200, height: 200))
    let image = renderer.image { (context) in
        UIColor.darkGray.setStroke()
        context.stroke(renderer.format.bounds)

        UIColor.randomColor.setFill()
        context.fill(CGRect(x: 1, y: 1, width: 140, height: 140))

        UIColor.randomColor.setFill()
        context.fill(CGRect(x: 60, y: 60, width: 140, height: 140), blendMode: .multiply)
    }
    imageView.image = image
}
```

и что угодно... (не забывайте про UIBezierPath)

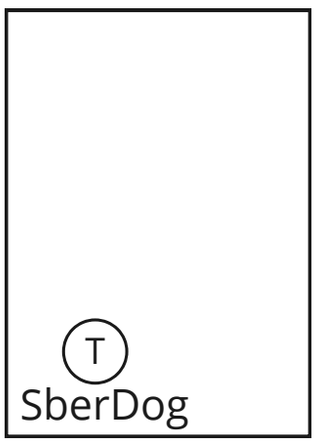
```swift
func drawYourBest() {
    let renderer = UIGraphicsImageRenderer(size: CGSize(width: 200, height: 200))
    let image = renderer.image { (context) in
        UIColor.darkGray.setStroke()
        context.stroke(renderer.format.bounds)
        UIColor.randomColor.setFill()
        context.fill(CGRect(x: 1, y: 1, width: 140, height: 140))
        UIColor.randomColor.setFill()
        context.fill(CGRect(x: 60, y: 60, width: 140, height: 140), blendMode: .multiply)

        UIColor.randomColor.setFill()
        context.cgContext.fillEllipse(in: CGRect(x: 60, y: 60, width: 140, height: 140))
    }
    imageView.image = image
}
```

Превратить view в UIImage

```swift
    let watermarkView = UIView(frame: CGRect(origin: .zero, size: size))
    watermarkView.addSubview(imageView)
    watermarkView.addSubview(label)
    watermarkView.backgroundColor = .clear

    let renderer = UIGraphicsImageRenderer(bounds: watermarkView.bounds)
    let image = renderer.image { rendererContext in
        watermarkView.layer.render(in: rendererContext.cgContext)
    }
```



rotate

```swift
func rotate(inputImage: UIImage, angle: CGFloat) -> UIImage {
    let radians = CGFloat(angle * .pi) / 180.0 as CGFloat

    var newSize = CGRect(
        origin: CGPoint.zero,
        size: inputImage.size
    ).applying(CGAffineTransform(rotationAngle: radians)).size

    // Trim off the extremely small float value to prevent core graphics from rounding it up
    newSize.width = floor(newSize.width)
    newSize.height = floor(newSize.height)
    let renderer = UIGraphicsImageRenderer(size: newSize)

    let image = renderer.image { rendederContext in
        let context = rendederContext.cgContext
        //rotate from center
        context.translateBy(x: newSize.width/2, y: newSize.height/2)
        context.rotate(by: radians)
        inputImage.draw(
            in: CGRect(
                origin: CGPoint(x: -inputImage.size.width/2,
                                y: -inputImage.size.height/2),
                size: inputImage.size
            )
        )
    }

    return image
}
```

crop

```swift
func crop(inputImage: UIImage, to cropRect: CGRect) -> UIImage? {
    let out = inputImage.cgImage!.cropping(to: cropRect)
    return UIImage(cgImage: out!)
}

extension UIImage {
    public func cropped(to cropRect: CGRect) -> UIImage? {
        let renderer = UIGraphicsImageRenderer(size: cropRect.size)
        return renderer.image { _ in
            self.draw(at: cropRect.origin)
        }
    }
}
```

Аффинные преобразования. CGAffineTransform

CGAffineTransform - Это структура, которая представляет особый вид преобразования, которое можно применить к любому подклассу UIView.

Существуют функции для увеличения масштаба представления, функции для поворота, функции для перемещения и функции для возврата к значениям по умолчанию.

Все эти функции возвращают значение **CGAffineTransform**, которое можно поместить в свойство преобразования view. А если делать это внутри блока анимации, преобразование будет автоматически анимировано

```
self.imageView.transform = CGAffineTransform(scaleX: 2, y: 2)
```

crop

```
func cropping(to: CGRect) -> CGImage?
```

Creates a bitmap image using the data contained within a subregion of an existing bitmap image.

```
func masking(CGImage) -> CGImage?
```

Creates a bitmap image from an existing image and an image mask.

resize

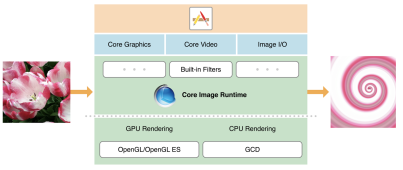
```swift
import UIKit
import CoreGraphics

// Technique #2
func resizedImage(at url: URL, for size: CGSize) -> UIImage? {
    guard let imageSource = CGImageSourceCreateWithURL(url as NSURL, nil),
        let image = CGImageSourceCreateImageAtIndex(imageSource, 0, nil)
    else {
        return nil
    }

    let context = CGContext(data: nil,
                            width: Int(size.width),
                            height: Int(size.height),
                            bitsPerComponent: image.bitsPerComponent,
                            bytesPerRow: 0,
                            space: image.colorSpace ?? CGColorSpace(name: CGColo
                            bitmapInfo: image.bitmapInfo.rawValue)
    context?.interpolationQuality = .high
    context?.draw(image, in: CGRect(origin: .zero, size: size))

    guard let scaledImage = context?.makeImage() else { return nil }

    return UIImage(cgImage: scaledImage)
}
```



CIContext

Вся обработка изображений происходит в контексте. Также, например, как в Core Graphics.

```
@interface CIContext : NSObject
```

Контекст представляет много вещей. Контекст хранит все состояния, он представляет кадровый буфер, в нем ведется вся работа с нашими изображениями.

Класс **CIContext** предоставляет контекст для обработки базового изображения с помощью Quartz 2D, Metal или OpenGL. Объекты CIContext в сочетании с другими классами Core Image, такими как **CIFilter**, **CIImage** и **CIColor** используются для обработки изображений с использованием фильтров **Core Image**. Также контекст Core Image используется с классом **CIDetector** для анализа изображений, например, для обнаружения лиц или штрих-кодов.

CIImage

```
@interface CIImage : NSObject
```

Этот класс содержит данные изображения. **Может быть создан из UIImage или файла изображения.**

Хотя объект CIImage связан с данными изображения, он не является изображением сам. Можно представить объект CIImage как о «чертеж изображения». Объект CIImage имеет всю информацию, необходимую для создания изображения, но Core Image на самом деле не рендерит изображение, пока не получит указание сделать это. **Это ленивое исполнение позволяет Core Image работать максимально эффективно.**

CIFilter

Figure 1-1 Construct a filter chain by connecting filter inputs and outputs

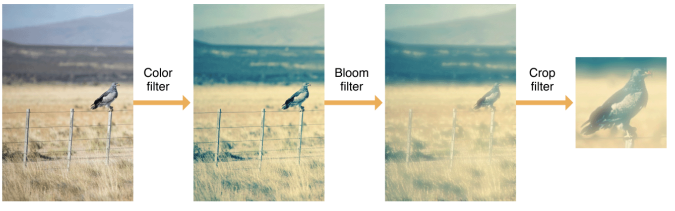
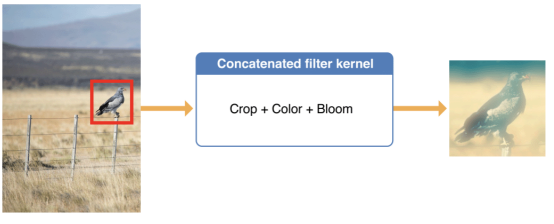


Figure 1-2 Core Image optimizes a filter chain to a single operation



Listing 1-1 The basics of applying a filter to an image

```
import CoreImage

let context = CIContext()                                          // 1

let filter = CIFilter(name: "CISepiaTone")!                        // 2
filter.setValue(0.8, forKey: kCIInputIntensityKey)
let image = CIImage(contentsOfURL: myURL)                          // 3
filter.setValue(image, forKey: kCIInputImageKey)
let result = filter.outputImage!                                   // 4
let cgImage = context.createCGImage(result, from: result.extent)   // 5
```

Here's what the code does:

1. Create a `CIContext` object (with default options). You don't always need your own Core Image context—often you can integrate with other system frameworks that manage rendering for you. Creating your own context lets you more precisely control the rendering process and the resources involved in rendering. Contexts are heavyweight objects, so if you do create one, do so as early as possible, and reuse it each time you need to process images. (See Building Your Own Workflow with a Core Image Context.)
2. Instantiate a `CIFilter` object representing the filter to apply, and provide values for its parameters. (See Filters Describe Image Processing Effects.)
3. Create a `CIImage` object representing the image to be processed, and provide it as the input image parameter to the filter. Reading image data from a URL is just one of many ways to create an image object. (See Images are the Input and Output of Filters.)
4. Get a `CIImage` object representing the filter's output. The filter has not yet executed at this point—the image object is a "recipe" specifying how to create an image with the specified filter, parameters, and input. Core Image performs this recipe only when you request rendering. (See Images are the Input and Output of Filters.)
5. Render the output image to a Core Graphics image that you can display or save to a file. (See Building Your Own Workflow with a Core Image Context.)

Нарисовать квадраты

и что угодно... (не забывайте про UIBezierPath)

rotate

resize

combine

```swift
func combine(background: CIImage, watermark: CIImage, watermarkSize: CGSize) -> CIImage {

    let resizeFilter = CIFilter(name: "CILanczosScaleTransform")!
    // Compute scale and corrective aspect ratio
    let scale = watermarkSize.height / (watermark.extent.height)
    let aspectRatio = watermarkSize.width/(watermark.extent.width * scale)

    // Apply resizing
    resizeFilter.setValue(watermark, forKey: kCIInputImageKey)
    resizeFilter.setValue(scale, forKey: kCIInputScaleKey)
    resizeFilter.setValue(aspectRatio, forKey: kCIInputAspectRatioKey)
    let scaledWatermark = resizeFilter.outputImage!

    let moveTransform = CGAffineTransform(
        translationX: background.extent.width - scaledWatermark.extent.width - 36,
        y: background.extent.height - scaledWatermark.extent.height - 36
    )

    let movedWatermark = scaledWatermark.transformed(by: moveTransform)

    let watermarkFilter = CIFilter(name: "CISourceOverCompositing")!
    watermarkFilter.setValue(background, forKey: kCIInputBackgroundImageKey)
    watermarkFilter.setValue(movedWatermark, forKey: kCIInputImageKey)

    let result = watermarkFilter.outputImage!
    return result
}
```

```
func composited(over: CIImage) -> CIImage
```

Returns a new image created by compositing the original image over the specified destination image.

crop

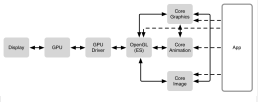
```swift
let croppedImage = ciImage.cropped(to: newRect)
```

nshipster.com

Image Resizing Techniques

Since time immemorial, iOS developers have been perplexed by a singular question: "How do you resize an image?" It's a question of beguiling clarity, spurred on by a mutual mistrust of developer and platform. Myriad code samples litter Stack Overflow, e...

Куча способов ресайзнуть картинку

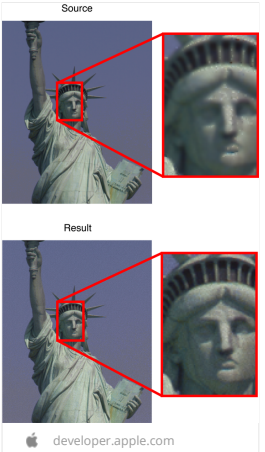


www.objc.io

Getting Pixels onto the Screen

How does a pixel get onto the screen? There are many ways to get something onto the display and they involve many different frameworks and many different combinations of functions and methods. Here we'll walk through some of the things that happen behin...

Как всё устроено



developer.apple.com

Core Image Filter Reference

This reference describes the built-in filters available through the Core Image API. You can also find out about the built-in filters on a system by using the Core Image API. See Core Image Programming Guide .

Таблица с фильтрами

developer.apple.com

Processing Images

Processing images means applying filters-an image filter is a piece of software that examines an input image pixel by pixel and algorithmically applies some effect in order to create an output image. In Core Image, image processing relies on the CIFilte...

core image гайд

AVCaptureSession

delegate

```
714     func captureOutput(_ output: AVCaptureOutput, didOutput sampleBuffer: CMSampleBuffer, from connection:
            AVCaptureConnection) {
715         if let videoDataOutput = output as? AVCaptureVideoDataOutput {
716             processVideoSampleBuffer(sampleBuffer, fromOutput: videoDataOutput)
717         } else if let audioDataOutput = output as? AVCaptureAudioDataOutput {
718             processsAudioSampleBuffer(sampleBuffer, fromOutput: audioDataOutput)
719         }
720     }
```

AVAssetWriter

AVAsset

PreviewLayer

Приложение с демотиваторами