Na aula passada criamos a interface gráfica do nosso app filtro de imagem, customizando nossa cena principal com um ImageView (mostrando uma imagem nele), e um botão. Bom, vamos agora ver como construir o código do app e dar uma funcionalidade a essa botão.

#01: Vá na StoryBoard, selecione o ImageView e no “attributes inspector” deixe vazio o campo Image para colocarmos a imagem no ImageView direto pelo código.

#02: Dentro da pasta do do projeto crie uma pasta chamada Resources

#03: Arraste a imagem do **bebe.jpg** que está dentro da pasta Desafios, a mesma onde você achou este arquivo, para a pasta Resources criada no passo 02.

#04: No canto superior direito clique no ícone onde mostram 2 círculos chamado “Show the Assistant Editor”. Isso irá dividir sua tela em 2, uma para a Storyboard e outra para o código do arquivo da cena que você tem selecionada, no caso, a cena ViewController.

#05: Antes de mais nada, precisamos conectar o elemento de interface, o ImageView, ao nosso código, para isso selecione o ImageView e com control pressionado arraste-o com o trackpad para qualquer lugar entre a classe e o método `viewDidLoad()`. Ao soltar, dê o nome de imageView à variável e Done. O `@IBOutlet` indica que a variável está conectada a um elemento na interface. Daqui a pouco iremos usar o ImageView no nosso código.

#06: Crie 4 variáveis abaixo do imageView:

- A primeira armazenará a imagem original

```
var imagemSemFiltro = UIImage(named: "bebe.jpg")
```

- A segunda para mais tarde armazenar a imagem após esta ter passado pelo filtro. Usamos o ponto de ! após o tipo para dizer que ela é uma optional até possuir um valor, depois disso não é preciso usar o ponto de ! para desempacotar seu valor.

```
var imagemComFiltro: UIImage!
```

- Uma terceira para armazenar o filtro futuramente

```
var filtro: CIImage!
```

- Vamos também criar uma outra para armazenar o contexto. O contexto nada mais é do que um local onde será realizado todo o processamento em cima de uma imagem do tipo CIImage, esse processamento ocorre internamente sem precisarmos conhecer toda a complexidade por trás desse processo.

```
let context = CIContext(options: nil)
```

#07: Vamos utilizar de início o método `viewDidLoad()` que é responsável por carregar os dados iniciais do App antes da cena aparecer para o usuário. Tudo abaixo deve ser acrescentado nesse método.

1) Primeiro vamos apresentar a imagem ao usuário. Para isso vamos usar o ImageView que conectamos ao código.

```
imageView.image = imagemSemFiltro
```

2) Agora vamos preparar a imagem para passar pelo filtro. Lembrando que para passar por filtros, as imagens precisam ser do tipo CIImage e não UIImage.

```
let imagemParaFiltro = CIImage(image: imagemSemFiltro)
```

3) Logo em seguida, vamos criar o filtro

```
filtro = CIFilter(name: "CISepiaTone", withInputParameters:
["inputImage":imagemParaFiltro, "inputIntensity": 1.0])
```

4) Agora vamos utilizar o contexto que criamos lá no início para extrair uma imagem do tipo CGImage que consegue preservar as dimensões da imagem de início.

```
let imagemRenderizada = context.createCGImage(filtro.outputImage,
fromRect: filtro.outputImage.extent())
```

Vamos explicar por partes a linha de código acima:

1 - context é a constante que criamos no início para armazernar um contexto do tipo CIContext, então usamos um método chamado `createCGImage()` justamente para criar uma imagem do tipo CGImage.

2 - Esse método precisa de 2 parâmetros: 1) uma imagem do tipo **CIImage** que é fornecida pela propriedade `outputImage` do filtro, e 2) as medidas da imagem que são fornecidas pelo método de CIImage chamado extent(), essas medidas são do tipo **CGRect**.

3 - O método `extent()` nos retorna um retângulo que especifica as dimênsões da imagem passada.

4 - Ao final, teremos uma imagem do tipo CGImage, que já passou pelo filtro e contém as mesmas dimensões da imagem antiga.

Agora vamos transformar essa CGImage em uma UIImage, para isso vamos usar um inicializador de UIImage que suporte um tipo CGImage como parâmetro:

```
imagemComFiltro = UIImage(CGImage: imagemRenderizada)
```

#08: Já carregamos a imagem na interface para o usuário e sem ele saber já geramos a imagem com filtro. Agora precisamos que ao clicar no botão Aplicar Filtro seja mostrada a imagem com o filtro. Para isso vamos fazer o mesmo procedimento que fizemos com o UIImageView, mas desta vez vamos selecionar o botão Aplicar Filtro e com control + trackpad arrastar até o código, coloque após o fecha chaves do `viewDidLoad()`. Ao soltar, escolha a opção Action em Connection e dê o nome de aplicarFiltro, em Type escolha UIBarButtonItem (que é o tipo do botão) e Done. Nisso foi gerado para nós um método responsável por fazer algo quando a pessoal clicar no botão. O `@IBAction` indica que o método deve ser executado quando o elemento de interface ao qual ele está ligado seja pressionado pelo usuário.

Dentro dele colocaremos a seguinte linha de código que carregará a imagem com o filtro no UIImageView quando o usuário clica no botão.

```
imageView.image = imagemComFiltro
```

#07: Agora é só mandar o App rodar e ver o resultado. Ao final seu código deve ter ficado parecido com o mostrado abaixo:

```swift
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var imageView: UIImageView!

    var imagemSemFiltro = UIImage(named: "bebe.jpg")

    var imagemComFiltro: UIImage!

    var filtro:CIFilter!

    let context = CIContext(options: nil)

    override func viewDidLoad() {
        super.viewDidLoad()

        //-- Carrega imagem sem filtro na interface

        imageView.image = imagemSemFiltro

        //-- Prepara a imagem para o filtro

        let imagemParaFiltro = CIImage(image: imagemSemFiltro)

        //-- Cria o filtro

        filtro = CIFilter(name: "CISepiaTone", withInputParameters: ["inputImage":imagemParaFiltro, "inputIntensity": 1.0])

        //-- Gera uma CGImage a partir de uma CIImage

        let imagemRenderizada = context.createCGImage(filtro.outputImage, fromRect: filtro.outputImage.extent())

        //-- Cria uma imagem do tipo UIImage a partir de uma CGImage

        imagemComFiltro = UIImage(CGImage: imagemRenderizada)

    }

    @IBAction func aplicarFiltro1(sender: UIBarButtonItem) {

        //-- Carrega a imagem com filtro na interface

        imageView.image = imagemComFiltro

    }
```