15 장. 카메라와 포토 라이브러리에서 미디어 가져오기

아이폰 사용자가 전화 기능 다음으로 가장 많이 사용하는 기능이 바로 사진(비디오)촬영일 것이다, 이때 사용하는 카메라와 포토 라이브러리라는 iOS 에서 제공하는 UIImagePickerController 클래스를 이용해 구현한다. 카메라를 사용하여 사진이나 비디오를 촬영하고 포토 라이브러리에 저장하는 방법을 알아보자.

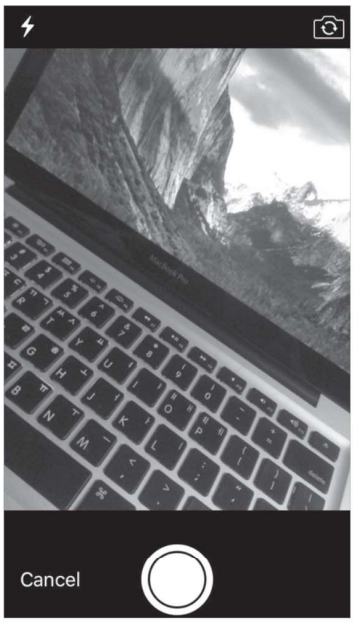


사진 촬영

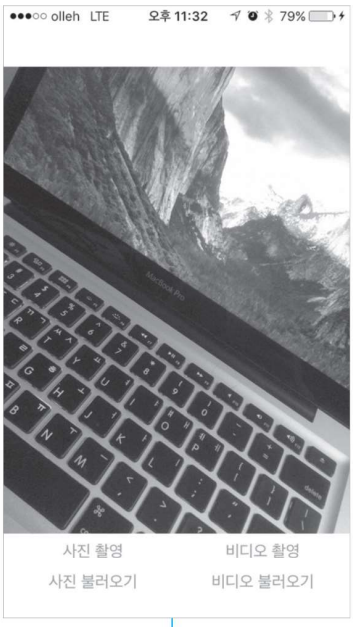
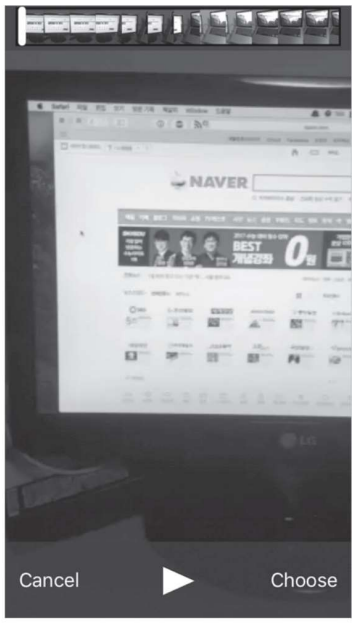


사진 불러오기



비디오 불러오기

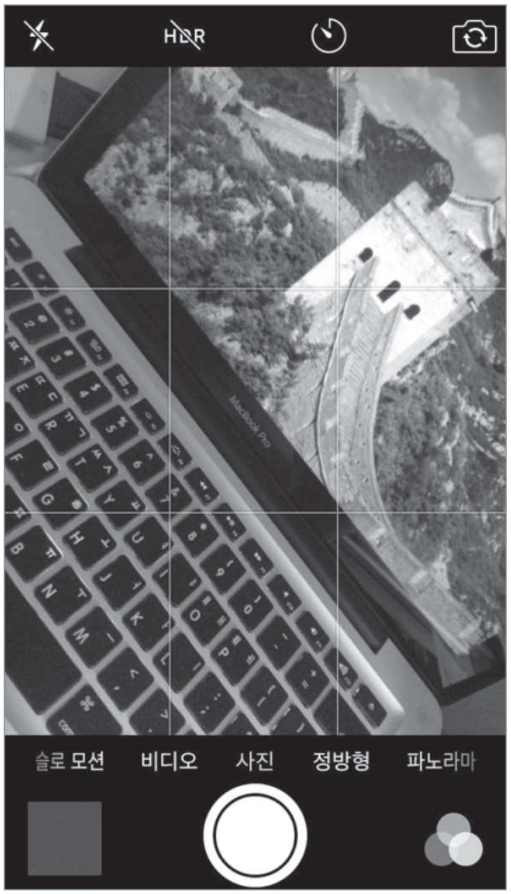
버튼 클릭만으로 사진이나 비디오를 촬영할 수 있으며 포토 라이브러리에 저장된 사진을 확인하거나 비디오를 재생할 수 있습니다.

15-1 카메라와 포토 라이브러리란?

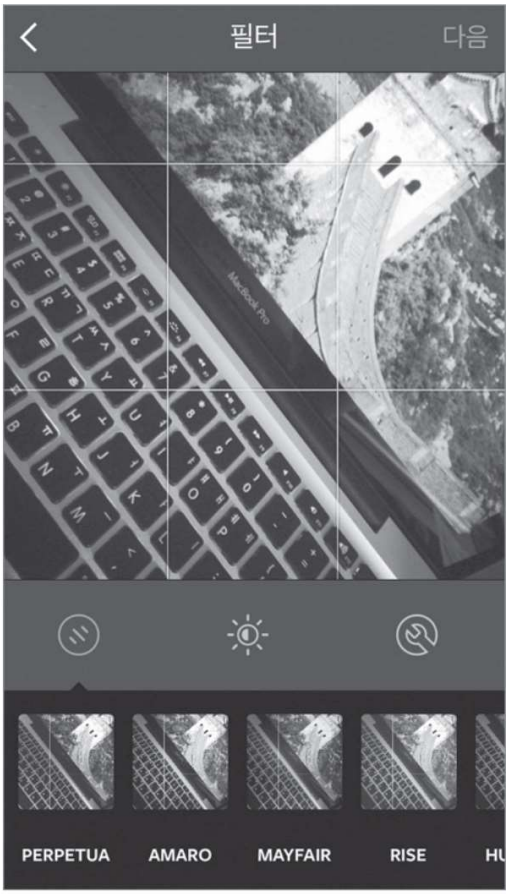
카메라(Camera)를 사용해서 찍은 사진이나 비디오는 포토 라이브러리(Photo Library)에 저장되고, 이렇게 저장된 사진이나 비디오를 포토 라이브러리에서 불러올 수 있다.

또한 사진을 편집한 후 포토 라이브러리에 저장할 수도 있다. 참고로 포토 라이브러리는 사진 앱의 앨범에서 확인할 수 있다.

이렇게 카메라 기능과 포토라이브러리 기능을 사용하면 다음과 같은 멋진 앱을 만들 수 있다.



카메라 앱



인스타그램(Instagram) 앱

15-2 카메라와 포토 라이브러리 앱을 위한 기본 환경 구성하기

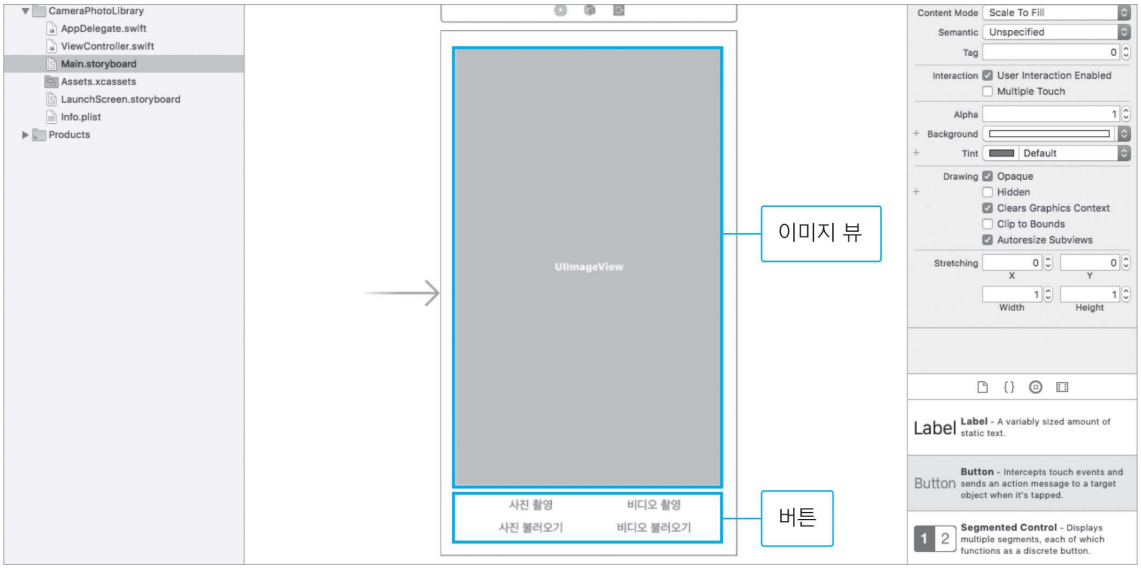
1. Xcode 를 실행 한 후 'CameraPhotoLibrary'라는 이름으로 새 프로젝트를 만든다.

2. 뷰 컨트롤러 크기 조절하기

아이폰 모양의 뷰 컨트롤러 크기를 상황에 맞게 조절한다.

3. 스토리보드 꾸미기

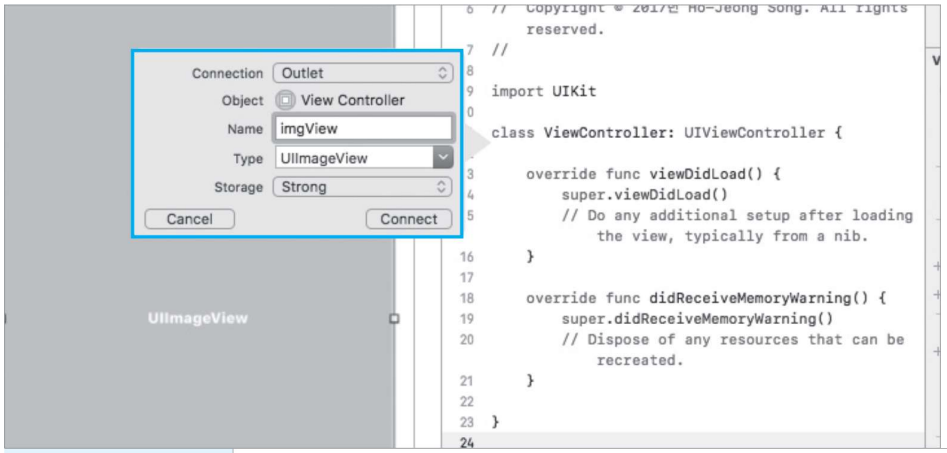
오른쪽 아랫부분의 오브젝트 라이브러리에서 [이미지 뷰(ImageView)]와 [버튼(Button)]을 찾아 스토리보드에 끌어온 후 아래 그림과 같이 배치한다. 그런 다음 버튼 네개의 이름을 각각 '사진 촬영','비디오 촬영','사진 불러오기','비디오 불러오기'로 수정한다.



15-3 아웃렛 변수와 액션 함수 추가하기

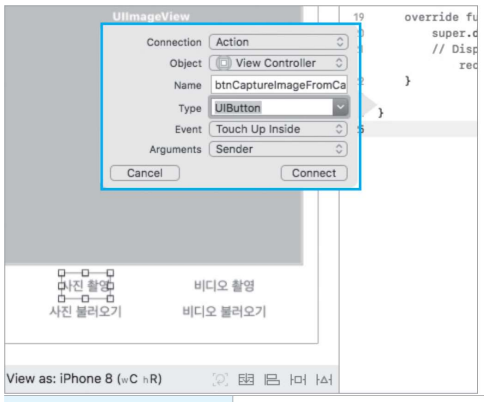
1. 아웃렛 변수와 액션 함수를 추가하기 위해 우선 오른쪽 윗부분의 [Show the Assistant editor]버튼을 클릭하여 보조 편집기 영역을 연다.

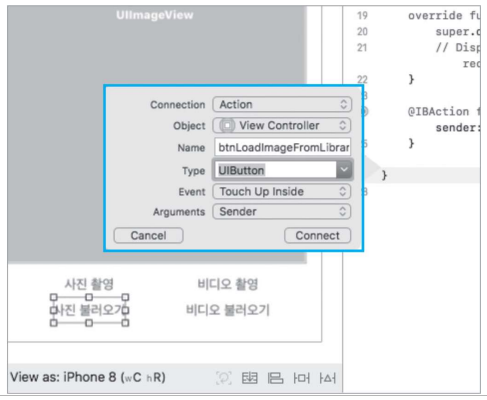
2. 이미지 뷰(ImageView)의 아웃렛 변수를 추가하자. [이미지 뷰]를 마우스 오른쪽 보픈으로 클릭한 후 드래그 앤 드롭하여 보조 편집기 영역에 끌어다 놓는다.



위치	뷰 컨트롤러의 클래스 선언문 바로 아래
연결(Connection)	Outlet
이름(Name)	imgView
유형(Type)	UIImageView

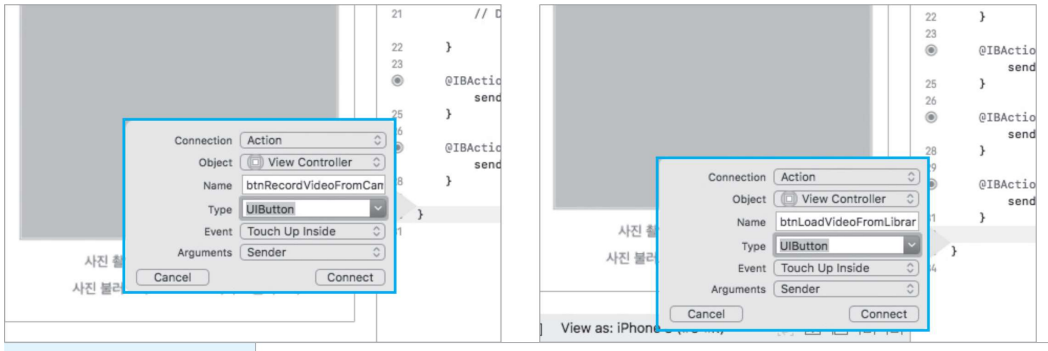
3. 이제는 버튼을 클릭했을 때 동작할 액션 함수를 추가한다, 아웃렛 변수와 같은 방법으로 보조 편집기 영역에 드래그 앤 드롭한다. 먼저 [사진 촬영]버튼과 [사진 불러오기] 버튼의 액션 함수를 만든다.





위치	소스의 가장 아래쪽 '}' 바로 위
연결(Connection)	Action
이름(Name)	- [사진 촬영] 버튼: btnCaptureImageFromCamera - [사진 불러오기] 버튼: btnLoadImageFromLibrary
유형(Type)	UIButton

4. 같은 방법으로 [비디오 촬영]버튼과 [비디오 불러오기] 버튼의 액션 함수를 추가한다.



위치	소스의 가장 아래쪽 '}' 바로 위
연결(Connection)	Action
이름(Name)	- [비디오 촬영] 버튼: btnRecordVideoFromCamera - [비디오 불러오기] 버튼: btnLoadVideoFromLibrary
유형(Type)	UIButton

15-4 카메라와 포토 라이브러리 앱 기능 구현하기

아웃렛 변수와 액션 함수에 구체적인 기능을 구현해 보자. 버튼의 이름에 맞게 '사진 촬영','사진 불러오기','비디오 촬영','비디오 불러오기'기능이 제대러 작동할 수 있도록 만들어 보자.

1. 스텐더드 에디터로 화면 모드 수정하기

코딩을 위해 화면 모드를 수정한다. 오른쪽 윗부분에서 [Show the Standard editor]버튼을 클릭한 후 왼쪽의 내비게이터 영역에서 [ViewController.swift]를 선택한다.

2. 헤더 파일과 델리게이트 프로토콜 추가하기 ImagePickerController 와 이 컨트롤러를 사용하기 위한 델리게이트 프로토콜이 필요하다. 그리고 미디어 타입이 정의된 헤더 파일이 있어야 한다, 먼저 소스 창에 있는 헤더 파일 import UIKit 아래에 MobileCoreService 헤더 파일을 추가 한다.

```
import UIKit
import MobileCoreServices

class ViewController: UIViewController{

}
```

[스위프트 문법] import는 왜 있나요?

예제를 진행하면서 프로젝트를 만들었을 때 맨 첫 줄의 import UIKit를 항상 보았을 것입니다. 지금까지는 무심코 넘겼다면 이제는 그 의미를 제대로 알아두세요.

먼저 알아두어야 할 것이 있습니다. 우리가 객체를 끌어와서 사용할 때 화면에는 아이콘만 보이지만 실제로는 미리 정의된 클래스를 사용하는 것입니다. 예를 들어 버튼 객체를 사용하는 것은 미리 정의된 버튼 클래스를 사용하는 것입니다. 그러므로 버튼 클래스를 불러와 소스에 추가해야만 합니다. 이때 필요한 것이 import입니다. import는 다른 파일이나 클래스를 추가하는 역할을 합니다.

import UIKit는 'User Interface와 관련한 클래스를 모아 놓은 파일을 추가한다'는 의미입니다. UIKit에는 우리가 오브젝트 라이브러리에서 추가하는 모든 객체의 클래스가 모여 있습니다. 그러므로 이러한 객체를 사용하기 위해서는 UIKit를 꼭 추가해야만 합니다. 이것이 자동으로 UIKit가 항상 추가되어 있는 이유입니다. 그 외에도 import를 사용해 필요한 파일이나 클래스를 직접 추가할 수 있습니다. 예를 들어 앞에서 사용한 MobileCoreServices에는 iOS에서 사용할 모든 데이터 타입들이 정의되어 있는 헤더 파일이 모여 있습니다. 그러므로 미디어 타입을 사용하기 위해서는 import를 사용해서 이 MobileCoreServices를 추가해야 합니다.

3. 또한 ImagePickerController 를 사용하기 위한 델리게이트 프로토콜 선언을 추가한다. 뷰 컨트롤러의 클래스 선언문에 다음 코드를 추가한다.

```
class ViewController: UIViewController, UINavigationControllerDelegate, UIImagePickerControllerDelegate {
```

4. 변수 추가하기

필요한 변수들을 추가하자. 이미지 뷰의 아웃렛 변수 아래에 다음 소스를 추기힌다.

```
1       @IBOutlet var imgView: UIImageView!
2
3       let imagePicker: UIImagePickerController! =
4                                                   UIImagePickerController()
5       var captureImage: UIImage!
6       var videoURL: URL!
7       var flagImageSave = false
```

3 행은 UIImagePickerController 의 인스턴스 변수 생성

5 행은 촬영을 하거나 포토 라이브러리에서 불러온 사진(Image)을 저장할 변수

6 행은 녹화한 비디오의 URL 을 저장할 변수

7 행은 이미지 저장 여부를 나타낼 변수

5. 경고 표시용 메서드 작성하기

문제가 생겼을 때 화면에 표시해 줄 경고 표시용 메서드를 작성한다. 다음 코드는 타이틀과 메시지를 받아 경고 창(6 장 얼럿)을 표시해 주는 메서드이다.

```
1   func myAlert(_ title: String, message: String) {
2           let alert = UIAlertController(title: title, message: message, preferredStyle:
3   UIAlertController.Style.alert)
4           let action = UIAlertAction(title: "Ok", style: UIAlertAction.Style.default,
5   handler: nil)
6           alert.addAction(action)
7           self.present(alert, animated: true, completion: nil)
8   }
```

6. '사진 촬영'코드 작성하기

카메라를 사용해 사진을 촬영하고 포토 라이브러리에 저장하는 코드를 작성한다. '사진 촬영'기능을 추가하기 위해 btnCaptureImageFromCamera 액션 함수에 다음 코드를 입력한다.

```
1   @IBAction func btnCaptureImageFromCamera(_ sender: UIButton) {
2           if (UIImagePickerController.isSourceTypeAvailable(.camera)) {
3               flagImageSave = true
4
5               imagePicker.delegate = self
6               imagePicker.sourceType = .camera
7               imagePicker.mediaTypes = [kUTTypeImage as String]
8               imagePicker.allowsEditing = false
9
10              present(imagePicker, animated: true, completion: nil)
11          }
12          else {
```

```
13                  myAlert("Camera inaccessable", message: "Application cannot
14  access the camera.")
15              }
16  }
17
18
19
20
```

2~3 행은 카메라의 사용가능 여부를 확인한다. 사용할 수 있는 경우에만 아래 내용을 수행한다.

5 행은 카메라 촬영 후 저장할 것이기 때문에 이미지 저장을 허용한다.

7 행은 이미지 피터의 델리게이트를 self 로 설정한다.

8 행은 이미지 피커의 소스 타입을 camera 로 설정한다.

9 행은 미디어 타입은 kUTTypeImage 로 설정한다.

10 행은 편집을 허용하지 않는다.

12~13 행은 현재 뷰 컨트롤러를 imagePicker 로 대체한다. 즉, 뷰에 imagePicker 가 보이게 된다.

17~18 행은 카메라를 사용할 수 없을 때는 경고 창을 나타난다.

7. '사진 불러오기' 코드 작성하기

이제 '사진 불러오기' 기능을 추가하기 위해 btnLoadImageFromLibrary 액션 함수에 다음 코드를 작성해야 한다. 각 소스의 설명은 [사진 촬영] 버튼에서 작성한 코드와 유사하다. 다른 점은 포토 라이브러리(PhotoLibrary)의 사용 가능 여부를 확인한 후 소스 타입은 [photoLibrary], 미디어 타입은 [kUTTypeImage]로 설정하고 편집은 허용한다는 점이다.

```
1   @IBAction func btnLoadImageFromLibrary(_ sender: UIButton){
2               if (UIImagePickerController.isSourceTypeAvailable(
    .photoLibrary)) {
3                   flagImageSave = false
4
5                   imagePicker.delegate = self
6                   imagePicker.sourceType = .photoLibrary
7                   imagePicker.mediaTypes = [kUTTypeImage as String]
8                   imagePicker.allowsEditing = true
9
10                  present(imagePicker, animated: true, completion: nil)
            }
11          else {
12                  myAlert("Photo album inaccessable", message: "Application cannot
13  access the photo album.")
            }
```

```
14  }
15
```

8. '비디오 촬영' 코드 작성하기

카메라를 사용해 비디오를 촬영하고 포토 라이브러리에 저장하는 코드를 작성한다. 이번에도 앞에서 입력한 소스와 유사하지만 '비디오 촬영' 코드에서는 카메라 사용 여부를 확인한 후 소스 타입은 [camera], 미디어 타입은 [kUTTypeMovie]로 설정하고 편집은 허용하지 않는다.

```
1   @IBAction func btnRecordVideoFromCamera(_ sender: UIButton) {
2           if (UIImagePickerController.isSourceTypeAvailable(.camera)) {
3               flagImageSave = true
4
5               imagePicker.delegate = self
6               imagePicker.sourceType = .camera
7               imagePicker.mediaTypes = [kUTTypeMovie as String]
8               imagePicker.allowsEditing = false
9
10              present(imagePicker, animated: true, completion: nil)
11          }
            else {
12              myAlert("Camera inaccessable", message: "Application cannot
13  access the camera.")
14          }
    }
15
16
```

9. '비디오 불러오기' 코드 작성하기

마지막으로 포토 라이브러리에서 비디오를 불러오는 코드를 작성한다. 포토 라이브러리(PhotoLibrary) 사용 여부를 확인한 후 소스 타입은 [photoLIbrary], 미디어 타입은 [kUTTypeMovie]로 설정하고 편집은 허용하지 않는다.

```
1   @IBAction func btnLoadVideoFromLibrary(_ sender: UIButton) {
2           if (UIImagePickerController.isSourceTypeAvailable(
    .photoLibrary)) {
3               flagImageSave = false
4
5               imagePicker.delegate = self
6               imagePicker.sourceType = .photoLibrary
7               imagePicker.mediaTypes = [kUTTypeMovie as String]
8               imagePicker.allowsEditing = false
9
10              present(imagePicker, animated: true, completion: nil)
            }
11          else {
```

```
12                  myAlert("Photo album inaccessable", message: "Application cannot
13  access the photo album.")
            }
14  }
15
```

10. 델리게이트 메서드 구현

앞에서 구현한 기능들이 경우에 따라 매끄럽게 작동하도록 만든다. 첫번째로 사용자가 사진이나 비디오를 촬영하거나 포토 라이브러리에서 선택이 끝났을 때 호출되는 didFinishPickingMediaWithInfo 메서드를 구현한다.

```
1   func imagePickerController(_ picker:
    UIImagePickerController,didFinishPickingMediaWithInfo info:
    [UIImagePickerController.InfoKey : Any]) {
2           let mediaType    = info[(UIImagePickerController.InfoKey).mediaType] as!
    NSString
            if mediaType.isEqual(to: kUTTypeImage as NSString as String) {
3               captureImage = info[UIImagePickerController.InfoKey.originalImage]
    as? UIImage
4
                if flagImageSave {
                    UIImageWriteToSavedPhotosAlbum(captureImage, self, nil, nil)
5               }
6               imgView.image = captureImage
            }else if mediaType.isEqual(to: kUTTypeMovie as NSString as String) {
                if flagImageSave {
7                   videoURL = (info[UIImagePickerController.InfoKey.mediaURL]
8   as! URL)
9
                    UISaveVideoAtPathToSavedPhotosAlbum(
10  videoURL.relativePath, self, nil, nil)
11              }
            }
12          self.dismiss(animated: true, completion: nil)
13  }

14
15
16
17
```

2 행은 미디어 종류를 확인 한다.

3 행은 미디어 종류가 사진(Image)일 경우

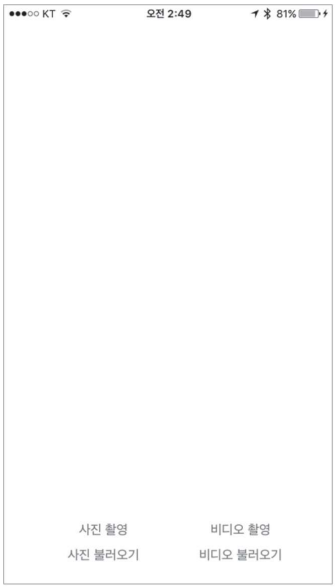
4 행은 사진을 가져와 captureImage 에 저장한다.

6 행은 flagImageSave 가 true 이면 가져온 사진을 포토라이브러리에 저장한다.
9 행은 미디어 종류가 비디오(Movie)일 경우
11 행은 flagImageSave 가 true 이면 촬영한 비디오를 가져와 포토라이브러리에 저장한다.
24 행은 현재의 뷰 컨트롤러를 제거한다. 즉 뷰에서 이미지 피커 화면을 제거하여 초기 뷰를보여 준다.

11. 두번째로 사용자가 사진이나 비디오를 찍지 않고 취소하거나 선택하지 않고 취소를 하는 경우 호출되는 imagePickerControllerDidCancel 메서드를 구현한다. 이 경우 처음의 뷰 상태로 돌아가야 하므로 현재의 뷰 컨트롤러에 보이는 이미지 피커를 제거하여 초기 뷰를 보여줘야 한다.

```
1  func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
2          self.dismiss(animated: true, completion: nil)
3  }
4
```

12. 결과 보기
[실행] 버튼을 클릭하여 앱을 실행한 후 [사진 촬영]과 [비디오 촬영] 버튼을 클릭하면 사진 및 비디오를 촬영할 수 있고, [비디오 불러오기] 버튼을 클릭하면 비디오를 재생할 수 있다.



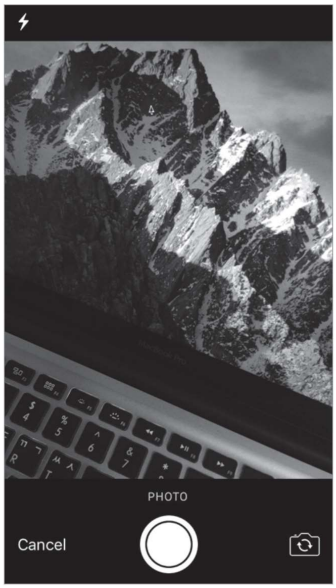


사진 촬영

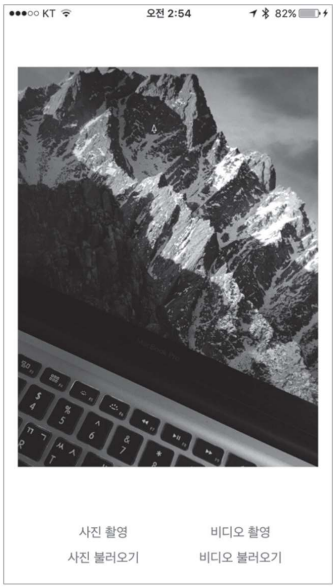


사진 불러오기

비디오 촬영

비디오 불러오기

```
//
//  ViewController.swift
//  CameraPhotoLibrary
//
//  Created by SoongM00 Rhee on 2018. 12. 6..
//  Copyright © 2018년 SoongMoo Rhee. All rights reserved.
//

import UIKit
import MobileCoreServices

class ViewController: UIViewController,UINavigationControllerDelegate,
UIImagePickerControllerDelegate  {

    @IBOutlet var imgView: UIImageView!
    //
    let imagePicker: UIImagePickerController! = UIImagePickerController()
    var captureImage: UIImage! // 이미지를 저장하기 위한 변수
    var videoURL: URL! // 파일의 경로를 가져오기 위한 URL
    var flagImageSave = false // 사진이나 동영상을 저장할 것인지
```

```
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }
    // 6장에서 배웠던 alert
    func myAlert(_ title: String, message: String) {
        // alert객체 생성 = title : alert의 출력, alert의 내용
        let alert = UIAlertController(title: title, message: message,
preferredStyle: UIAlertController.Style.alert)
        // alert에 예/아니오
        let action = UIAlertAction(title: "Ok", style: UIAlertAction.Style.default,
handler: nil)
        alert.addAction(action)
        self.present(alert, animated: true, completion: nil)
    }
    @IBAction func btnCaptureImageFromCamera(_ sender: UIButton) {
        // 카메라 사용여부 확인
        if (UIImagePickerController.isSourceTypeAvailable(.camera)) {
            flagImageSave = true
            // imagePickerController의 delegate에게 현재
ViewController객체를 저장
            // imagePickerController의 delegate를 ViewController객체로
사용하기 위해
            imagePicker.delegate = self
            // imagePicker.sourceType 를 카메라로 사용할 수 있게 저장
            imagePicker.sourceType = .camera
            // 미디어 타입으로 사용
            imagePicker.mediaTypes = [kUTTypeImage as String]
            // 편집허용 여부
            imagePicker.allowsEditing = false
            // 뷰 컨트롤러에 있는 present 함수를 이용해서 뷰 컨트롤러를
imagePickercontroller로 대체
            present(imagePicker, animated: true, completion: nil)
        }
        else {
            // 카메라가 익식하지 않을 때 alert메세지 출력
```

```
                myAlert("Camera inaccessable", message: "Application cannot
access the camera.")
            }
        }
        @IBAction func btnLoadImageFromLibrary(_ sender: UIButton) {
            // photoLibrary를 사용할 수 있는지 확인
            if (UIImagePickerController.isSourceTypeAvailable(
                .photoLibrary)) {
                flagImageSave = false

                imagePicker.delegate = self
                 // imagePicker.sourceType 를 포토라이브러리로 사용할 수
있게 저장
                imagePicker.sourceType = .photoLibrary
                imagePicker.mediaTypes = [kUTTypeImage as String]
                imagePicker.allowsEditing = true
                present(imagePicker, animated: true, completion: nil)
            }
            else {
                myAlert("Photo album inaccessable", message: "Application
cannot access the photo album.")
            }

        }

        @IBAction func btnRecordVideoFromCamera(_ sender: UIButton) {
            if (UIImagePickerController.isSourceTypeAvailable(.camera)) {
                flagImageSave = true

                imagePicker.delegate = self
                imagePicker.sourceType = .camera
                imagePicker.mediaTypes = [kUTTypeMovie as String]
                imagePicker.allowsEditing = false

                present(imagePicker, animated: true, completion: nil)
            }
            else {
                myAlert("Camera inaccessable", message: "Application cannot
access the camera.")
            }
```

```
    }
    @IBAction func btnLoadVideoFromLibrary(_ sender: UIButton) {
        if (UIImagePickerController.isSourceTypeAvailable(
            .photoLibrary)) {
            flagImageSave = false

            imagePicker.delegate = self
            imagePicker.sourceType = .photoLibrary
            imagePicker.mediaTypes = [kUTTypeMovie as String]
            imagePicker.allowsEditing = false

            present(imagePicker, animated: true, completion: nil)
        }
        else {
            myAlert("Photo album inaccessable", message: "Application
cannot access the photo album.")
        }
    }
    // 구현된 기능을 매끄럼게 실행되게 하기위해서 필요
    func imagePickerController(_ picker:
UIImagePickerController,didFinishPickingMediaWithInfo info:
[UIImagePickerController.InfoKey : Any]) {
        // 미디어 종류를 확인
        let mediaType    = info[(UIImagePickerController.InfoKey).mediaType]
as! NSString
        // 타입이 이미지인 경우
        if mediaType.isEqual(to: kUTTypeImage as NSString as String) {
            // 이미지를 가져와 이미지 변수에 저장
            captureImage =
info[UIImagePickerController.InfoKey.originalImage] as? UIImage
            // flagImageSave가 true이면 포토라이블리에 저장
            if flagImageSave {
                UIImageWriteToSavedPhotosAlbum(captureImage, self, nil,
nil)
            }
            // 이미지를 가져와 이미지 뷰에 출력
            imgView.image = captureImage
        }// 타입이 미디어인 경우
        else if mediaType.isEqual(to: kUTTypeMovie as NSString as String) {
```

```
                    //
                    if flagImageSave {
                        videoURL =
(info[UIImagePickerController.InfoKey.mediaURL] as! URL)

                        UISaveVideoAtPathToSavedPhotosAlbum(
                            videoURL.relativePath, self, nil, nil)
                    }
                }
                self.dismiss(animated: true, completion: nil)
            }
            func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
                self.dismiss(animated: true, completion: nil)
            }
}
```