PROJECT PLAN TWO

ATLAS 5519 iOS 6 MAD

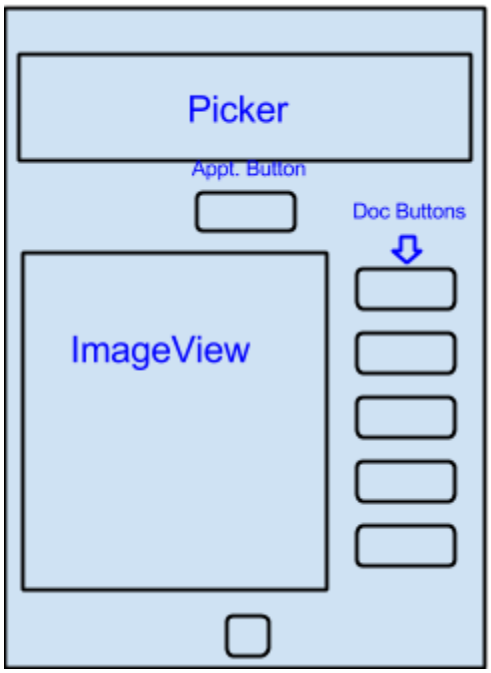
Danielle Rodriguez

10/31/2013

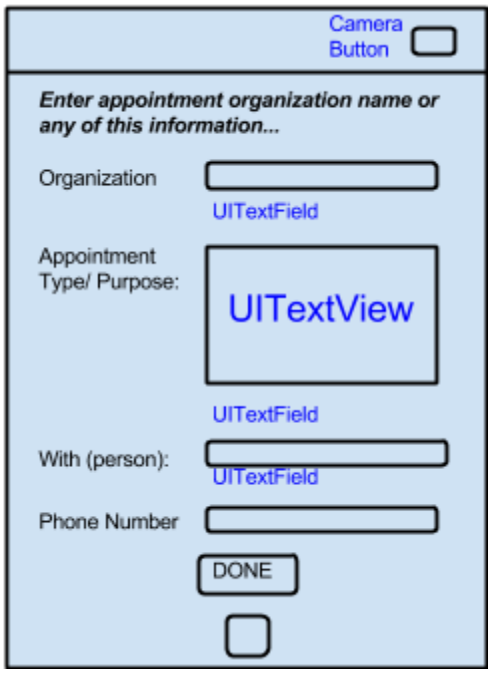
ELIMINATE PICKER

Improvement: Pickers should only be used in a detail view if on an iPad or not at all. Decided to eliminate picker all together and incorporate data persistence from one view to a second and an algorithm that draws from user input the appropriate pictures to display.

Project One UI:



FirstViewController Project Two:



Properties:

```
@property(weak, nonatomic) IBOutlet UITextField *org;
@property(weak, nonatomic) IBOutlet UITextView *apptNotes;
@property(weak, nonatomic) IBOutlet UITextField *person;
@property(weak, nonatomic) IBOutlet UITextField *phone;

@property(weak, nonatomic) IBAction UIButton *doneButton;
@property (weak, nonatomic)IBAction UIButton *shootPicture;
```

Interaction / Pseudocode:

Algorithm created that parses through input into any of UITextFields or UITextView to assign appropriate images to appointment the caseworker is accompanying client to. For example, if a

user enters “IRC”, “irc”, “International Rescue Committee”, or “international rescue committee” (other variants may be added upon testing), an action sheet will pop up that gives the user options toward viewing different appointments typically at the IRC.

Additionally, if the caseworker knows who the appointment is with or an office phone number, the algorithm can parse and determine which kind of appointment the client might be attending based on bits of that information or a combination (and statement / or statement). This section may be simplified to only the organization where the appointment is at as a caseworker should know at a minimum where they are going and why.

Through data persistence from FirstViewController to SecondViewController, a collection of images in SecondViewController display in the UIImageView (see below) based upon the button on an action sheet that pops up in FirstViewController after entering information into UITextFields and pressing UIButton Done. Here, alternatively, the action sheets and algorithm parsing the information input into the UITextField and UITextView could be initiated when the user taps out of the text field or presses enter in the case of the UI being simplified to only one text field for organization.

IMPLEMENTING A CAMERA
Utility: Allows casework to snap photos of documents not already assigned to an appointment to recall and view.

Pseudocode: `UIImagePickerController`
Description: Camera and photo library are made available by way of image picker, a mechanism that lets you select an image from a specific source that lives outside the app’s sandbox. It is implemented by way of modal controller class - `UIImagePickerController` - for which the necessary steps are to: a) create an instance of this class, b) specify it’s image source and whether user will access an image or a view, and c) launch modally.

Launch image picker:

```
UIImagePickerController *picker = [[UIImagePickerController alloc]init];
picker.delegate=self;
picker.sourceType=UIImageControllerSourceTypeCamera;
[self presentViewController:picker animated:YES completion:NULL];
```

`UIImagePickerControllerDelegate` defines two methods:

`imagePickerController:didFinishPickingMediaWithInfo:`
Called when user successfully taken a photo, or selected an item from the media library.

`imagePickerControllerDidCancel:`
Called when user decides to cancel the process without capturing or selecting any media.

Modal views like image picker must be told when to dismiss themselves, so at a minimum, `imagePickerControllerDidCancel:` will look like:

```
- (void)imagePickerControllerDidCancel: (UIImagePickerController *)picker
{
        [picker dismissViewControllerAnimated: YES completion: NULL];
}
```

Protocols in FirstViewController: Conform class of `UIImagePickerController` to UIImagePickerControllerDelegate and `UINavigationControllerDelegate`. UIImagePicker is a subclass of `UINavigationController`, so we need to conform to the protocol even if we don't use the method.

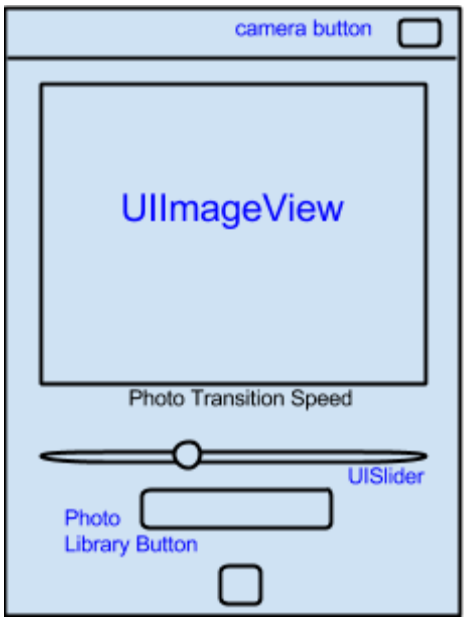
// FirstViewController.h

```
#import <UIKit/UIKit.h>

@interface FirstViewController: UIViewController <UIImagePickerControllerDelegate,
UINavigationControllerDelegate>
....
@end
```

SecondViewController

Where the images will display.



Properties:

```
@property (weak, nonatomic) IBAction UIButton *photoLibrary;
@property (weak, nonatomic) IBOutlet UISlider *transitionSpeed;
@property (weak, nonatomic) IBOutlet UIImageView *imageWindow;
```

Important first to control drag from File's Owner icon to image view and select `imageWindow` outlet. Control drag from File's Owner to Photo Library button and select `photoLibrary:` action. Next select photo library button and bring up connections inspector. Drag from *Touch Up Inside* event to File's Owner and select `photoLibrary:` action.

To implement the Camera View Controller, go in SecondViewController.m to make the changes:

```
#import 'SecondViewController.h'
#import <MediaPlayer/MediaPlayer.h>
#import <MobileCoreServices/UTCoreTypes.h>

@interface SecondViewController ()
@property (strong, nonatomic) MPMoviePlayerController *moviePlayerController;
@property (strong, nonatomic) UIImage *image;
@property (strong, nonatomic) NSURL *movieURL;
@property (copy, nonatomic) NSString *lastChosenMediaType;
@end

@implementation SecondViewController

-(void)viewDidLoad
{
        [super viewDidLoad];
        if (![UIImagePickerController isSourceTypeAvailable:
UIImagePickerControllerSourceTypeCamera])
{
        self.takePictureButton.hidden = YES;
}
}

-(void)viewDidAppear:(BOOL)animated
{
        [super viewDidAppear:animated];
        [self updateDisplay];
}
@end
```

Add three utility methods. updateDisplay method is called both when the view is first created and then again after the user picks an image or video and dismisses the image picker. pickMediaFromSource: utility method creates and configures an image picker, using the passed in sourceType to determine whether to bring up the camera or the media. The third utility method is shrinkImage:toSize: which is used to shrink our image down to the size of the view that's going to show it. These are a series of calls that create a new image based on the specified size and render the old image into the new one.

Finally, insert action methods from header file. Each of these calls out to one of utility methods defined earlier, passing a value defined by UIImagePickerController to specify where the picture or video should come from.

Finish by implementing delegate methods for picker view. The first method checks to see whether a picture or video was chosen, makes note of the selection (shrinking the chosen image, if any, to precisely fit the display size along the way, and then dismisses the modal image picker. The second dismisses the image picker.

```
if ([UIImagePickerController
     isSourceTypeAvailable:sourceType] && [mediaTypes count] > 0) {
   NSArray *mediaTypes = [UIImagePickerController
                          availableMediaTypesForSourceType:sourceType];
   UIImagePickerController *picker = UIImagePickerController alloc] init];
    picker.mediaTypes = mediaTypes;
    picker.delegate = self;
    picker.allowsEditing = YES;
   picker.sourceType = sourceType;
   [self presentViewController:picker animated:YES completion:NULL];
} else {
   UIAlertView *alert =
    [[UIAlertView alloc] initWithTitle:@'Error accessing media'
message:@'Device doesn't support that media source.'
         delegate:nil
cancelButtonTitle:@'Drat!'
otherButtonTitles:nil];
 [alert show];
    }
}

- (UIImage *)shrinkImage:(UIImage *)original toSize:(CGSize)size
{
   UIGraphicsBeginImageContextWithOptions(size, YES, 0);
    [original drawInRect:CGRectMake(0, 0, size.width, size.height)];
    UIImage *final = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
   return final;
}

- (IBAction)shootPictureOrVideo:(id)sender
{
   [self pickMediaFromSource:UIImagePickerControllerSourceTypeCamera];
}
- (IBAction)selectExistingPictureOrVideo:(id)sender
{
   [self pickMediaFromSource:UIImagePickerControllerSourceTypePhotoLibrary];
}

#pragma mark - Image Picker Controller delegate methods
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
   self.lastChosenMediaType = info[UIImagePickerControllerMediaType];
    if ([self.lastChosenMediaType isEqual:(NSString *)kUTTypeImage]) {
       UIImage *chosenImage = info[UIImagePickerControllerEditedImage];
        UIImage *shrunkenImage = [self shrinkImage:chosenImage
toSize:self.imageView.bounds.size];
       self.image = shrunkenImage;
    } else if ([self.lastChosenMediaType isEqual:(NSString *)kUTTypeMovie]) {
```

```
        self.movieURL = info[UIImagePickerControllerMediaURL];
    }
   [picker dismissViewControllerAnimated:YES completion:NULL];
}
- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker
{
   [picker dismissViewControllerAnimated:YES completion:NULL];
}
```

Pseudocode: Pictures transition through arrays that correspond with appointment and necessary documents. UISlider controls speed of transition (in seconds) from 1.0 - 5.0. Alternatively, could modify so that slider controls which picture the UIImageView is displaying or implement a scrollView with page control. This is an area where I'd like to test and reiterate upon design. I'm not yet certain which design would serve the app best yet.
Furthermore, there is a button that will take user to photo library if they'd like to view a different picture. One concern is that the UIImageView won't serve the purpose of an outlet for a single picture if it is set to animate (transition). This is another reason why animation may be turned off and the slider used to transition from picture to picture in an array. Testing needs to be done with the pinching gesture which is a top priority to implement as it will allow caseworkers a closer view on some of those documents.

Example with UISlider controlling transition speed for pictures:

```
-(void)viewDidLoad
{
        [super viewDidLoad];
        self.title=NSLocalizedString(@"ImagesTitle", @"");
// set up UIImage with a group or array of images to animate
self.imageView.animationImages=@[
                                                [UIImage imageNamed:@"______.png"];
                                                [UIImage imageNamed:@"______.png"];
                                                [UIImage imageNamed:@"______.png"];
                                                [UIImage imageNamed:@"______.png"];
                                                [UIImage imageNamed:@"______.png"]
                                        ];
self.imageView.animationDuration=5.0;
[self.imageView stopAnimating];

// Set accessibility labels
[self.imageView setIsAccessibilityElement:YES];
[self.imageView setAccessibilityLabel:self.title];
[self.slider setAccessibilityLabel:NSLocalizedString(@"PhotoSlider", @"")];
}

//slow down or speed up the slide show as slider is moved
-(IBAction)sliderAction:(id)sender
{
        UISlider *transitionSpeed = sender;
        self.imageView.animationDuration=[transitionSpeed value];
        if (!self.imageView.isanimating)
                [self.imageView startAnimating];
```

```
}

#pragma mark - UIViewController delegate methods

-(void)viewWillDisappear:(BOOL)animated
{
        [super viewWillDisappear:animated];
        [self.imageView stopAnimating];
}

-(void)viewWillAppear:(BOOL)animated
{
        [super viewWillAppear:animated];
        [self.imageView startAnimating];

}
@end
```

DETECTING PINCH AND ROTATION

Pseudocode: `UIPinchGestureRecognizer`

Description: `UIPinchGestureRecognizer` is a continuous gesture recognizer because it calls an action method over and over again during the pinch when the gesture is underway, recognizer goes through a number of states.

Implementation in UIImageView:

// SecondViewController.h

Let FirstViewController conform to UIGestureRecognizerDelegate after UIViewController. This should allow several gesture recognizers to recognize gestures simultaneously. Will it allow gestures when there's a transition element, like my app? This is something to test and consider the UISlider if not so.

```
@implementation SecondViewController: UIViewController
<UIGestureRecognizerDelegate>
```

In SecondViewController.m, delete:

```
@interface SecondViewController ( )
@end
```

Add beneath `@implementation SecondViewController {`

```
CGFloat scale, previousScale;
CGFloat rotation, previousRotation;
{

Add under [superViewDidLoad];
previousScale=1;
UIImage *image=[UIImageNamed: "__________.png"];
```

// This is where I would need to modify to work with arrays

```
self.imageView=[UIImageView alloc]initWithImage:image];
self.imageView.userInteractionEnabled=YES;
self.imageView.center=CGPointMake(160,160);
[self.view addSubview:self.imageView1;
UIPinchGestureRecognizer *pinchGesture= [[UIPinchGestureRecognizer alloc]initWithTarget:
self action@selector(doPinch:)];
pinchGesture.delegate=self;
[self.imageView addGestureRecognizer:rotationGesture];
```

Add a new method beneath

```
-(BOOL)gestureRecognizer:(UIGestureRecognizer *) gestureRecognizer
shouldRecognizeSimultaneouslyWithGestureRecognizer:(UIGestureRecognizer
*)otherGestureRecognizer {
return YES;
}
```

Add another method

```
-(void)transformImageView
{
       CGAffineTransform t = CGAffineTransformMakeScale(scale *previousScale, scale
*previousScale);
t=CGAffineTransformRotate(t, rotation + previousRotation);
self.imageView.transform=t;
}
```

Add method

```
-(void)doPinch:(UIPinchGestureRecognizer *)gesture
{
scale=gesture.scale;
[self transformImageView];
if(gesture.state==UIGestrureRecognizerStateEnded){
previousScale=scale *previousScale;
scale=1;
}
}
```

Add method

```
-(void)doRotate:(UIRotationGestureRecognizer *)gesture
{
rotation=gesture.rotation;
[self transformImageView];
if(gesture.state==UIGestureRecognizerStateEnded){
previousRotation=rotation +previousRotation;
rotation=0;
}
```

Reference Material:

- “Beginning iOS6 Development”
- UICatalog from Apple Developer tools