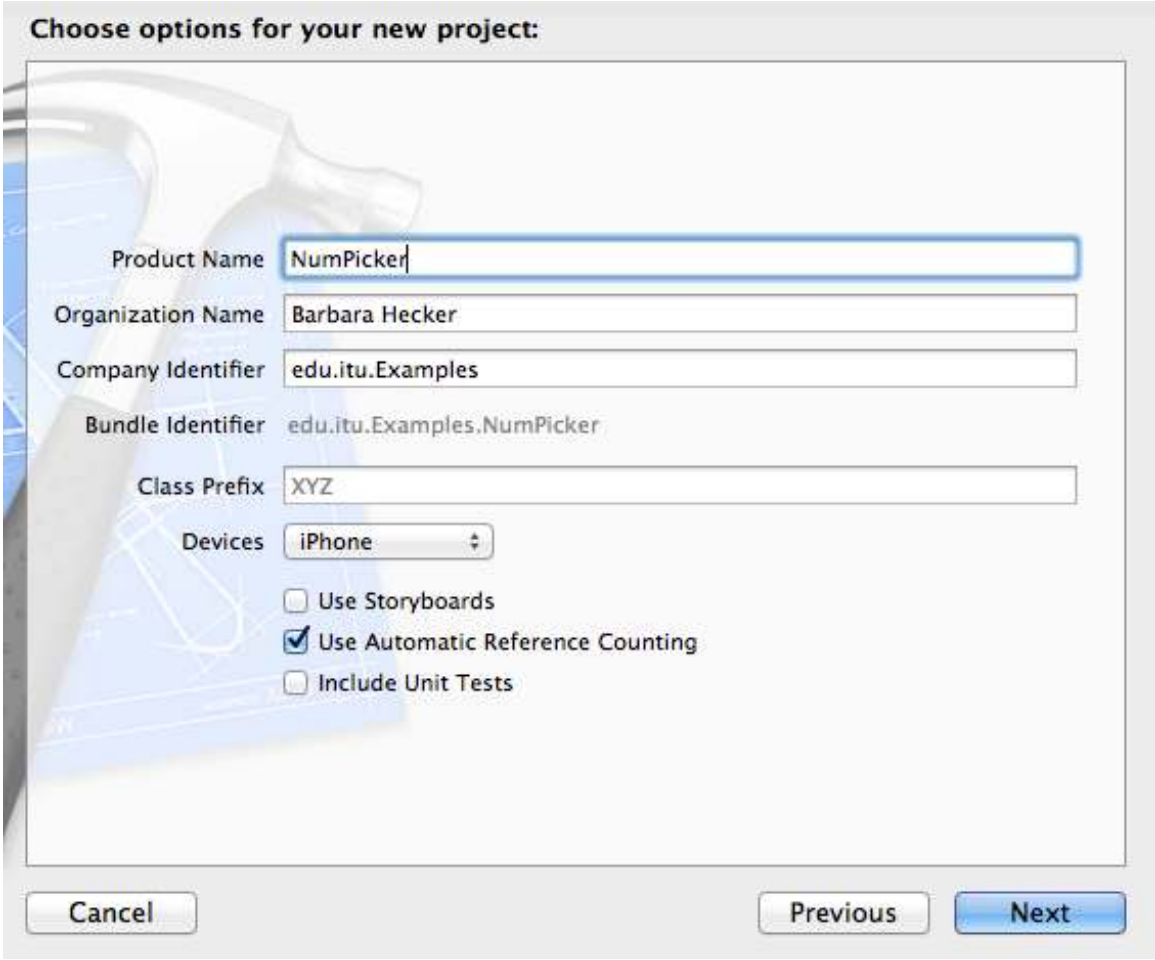
Picking and Comparing Numbers

In this tutorial we will use a UIPicker control to implement a simple lottery application. The user will choose a two digit number using a picker control then press a button to see if their number matches a random number generated by the app.

Step 1: Start Xcode, choose "Create a new Xcode project," then select the Single View Application template. Name the project **NumPicker**, and choose options as shown:



Click Next, choose a location to save the project, and Click Create.

Step 2: Open ViewController.h and make the following changes:

Change the @interface line to add the use of the UIPickerViewDataSource and UIPikcerViewDelegate protocols. The finished declaration looks like this:

@interface ViewController : UIViewController
<UIPickerViewDataSource, UIPickerViewDelegate>

Step 3: Create the User Interface. Open ViewController.xib, and drag controls to the main view as shown below:

Step 4: Wire the components to ViewController.h. Wire the label to "prompt" and the picker to "picker." Wire the button IBAction as checkEntry. The finished items look like this:

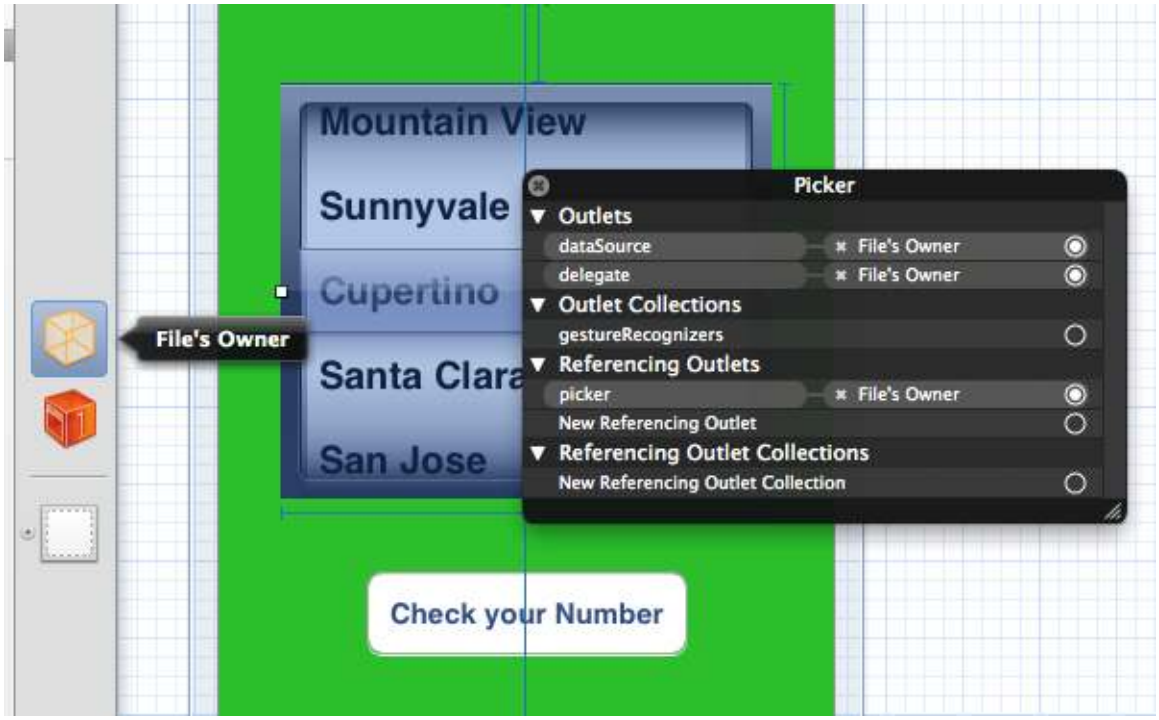
@property (weak, nonatomic) IBOutlet UILabel *prompt;

@property (weak, nonatomic) IBOutlet UIPickerView *picker;

- (IBAction)checkEntry:(UIButton *)sender;

As you can see, we have added two properties: a **UILabe**l control named prompt, and a **UIPickerView** control called picker. We have also added a single action method called **checkEntry**, which will check the value in the picker view against a random number. The ViewController class has also been made to adopt two protocols: **UIPickerViewDataSource** and **UIPickerViewDelegate.**

Step 5: Right click the picker view and drag from both the delegate and datasource to File's Owner. **This is a very important step:**



Step 6: We now need to write the implementation for the app. Open **ViewController.m**, and make the changes shown in the listing below:

Synthesize the two properties as follows:

```
@synthesize prompt, picker;
```

Implement the required protocol methods as follows:

```
- (NSInteger)numberOfComponentsInPickerView:(UIPickerView
*)pickerView
{
    return 2;
}
- (NSInteger)pickerView:(UIPickerView *)pickerView
numberOfRowsInComponent:(NSInteger)component
{
    return 10;
}
- (NSString *)pickerView:(UIPickerView *)pickerView
titleForRow:(NSInteger)row forComponent:(NSInteger)component
```

```
{
    return [NSString stringWithFormat:@"%d", row];
}
```

Add the following line to the ViewDidLoad method to set the prompt on the label:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    self.prompt.text = @"Choose a number please.";

}
```

Implement the IBAction for the button behavior:

```
- (IBAction)checkEntry:(UIButton *)sender {
    int winningNumber = (arc4random() % 100) ;
    int chosenNumber;
    int tens = [self.picker selectedRowInComponent:0] * 10;
    int ones = [self.picker selectedRowInComponent:1];
    chosenNumber = tens + ones;
    if (chosenNumber == winningNumber) {
        self.prompt.text = @"You Win!";
    } else {
        self.prompt.text = [NSString stringWithFormat:@"Sorry, the
winning number was: %d", winningNumber];
    }
}
```

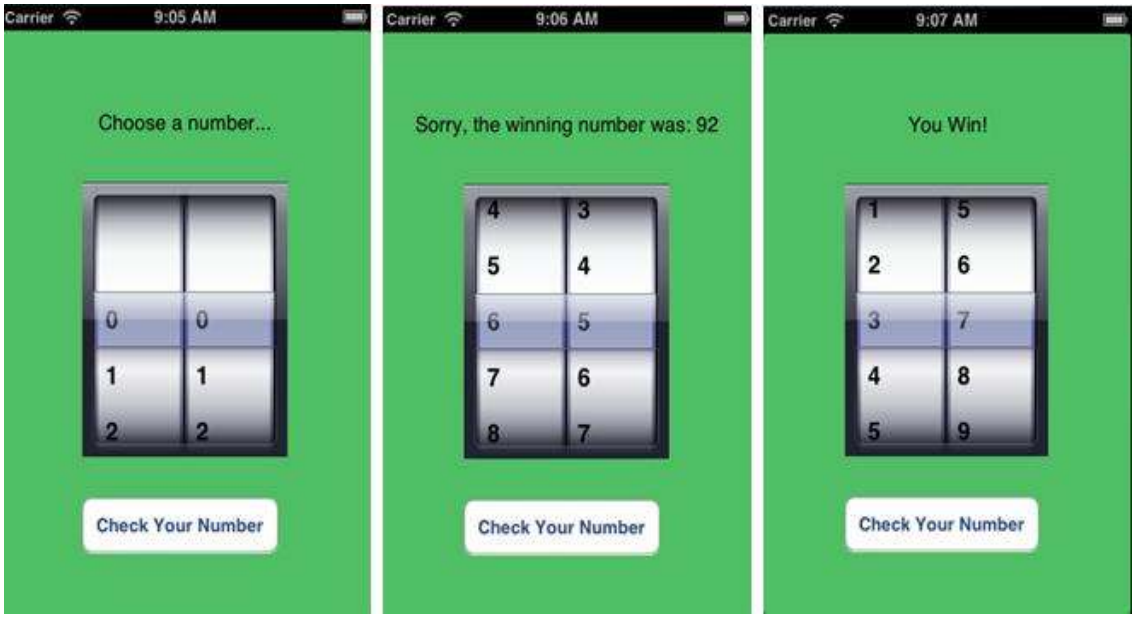
After synthesizing the properties, we proceed to implement three delegate methods of the UIPickerView control: numberOfComponentsInPickerView:, numberOfRowsInComponent:, and pickerView: titleForRow: forComponent:. Since each component of the picker view only needs to contain the digits 0 through 9, these implementations are every straightforward.

NumberOfComponentsInPickerView returns 2 (because there are two "dials" in the picker, one for each digit), numberOfRowsInComponent returns 10 (for all components) because each component will list 10 digits, and pickerView: titleForRow: forComponent: simply formats each row number in each component as a string before returning it.

In the checkEntry: method, we first generate the random number by calling arc4random(), which generates a random int. We mod this with 100, which will return an int in the range 0..99. Next, we get the value of the selected row in each component of the picker into two int variables named tens and ones. We must multiply the tens row by 10 before adding the two together and placing the result in chosenNumber. Finally, we compare chosenNumber with winningNumber and display the result in the prompt control.

We could have set up a property to hold the random number in the ViewController class, and just done the comparison with the picker values in checkEntry:. But we would have had to initialize the random number property in ViewDidLoad:, and reset it somewhere after the comparison was made.

In this implementation, we take advantage of the fact that it doesn't matter when the random number is generated, as long as the code is "honest" (doesn't influence the choice of random number depending on the values in the picker). Implementing the random number generation this way also has another advantage: since winningNumber is entirely within the scope of checkEntry, we are certain to get a new winning number each time the button is touched.



Step 7: Run your program and guess some numbers. You are done!