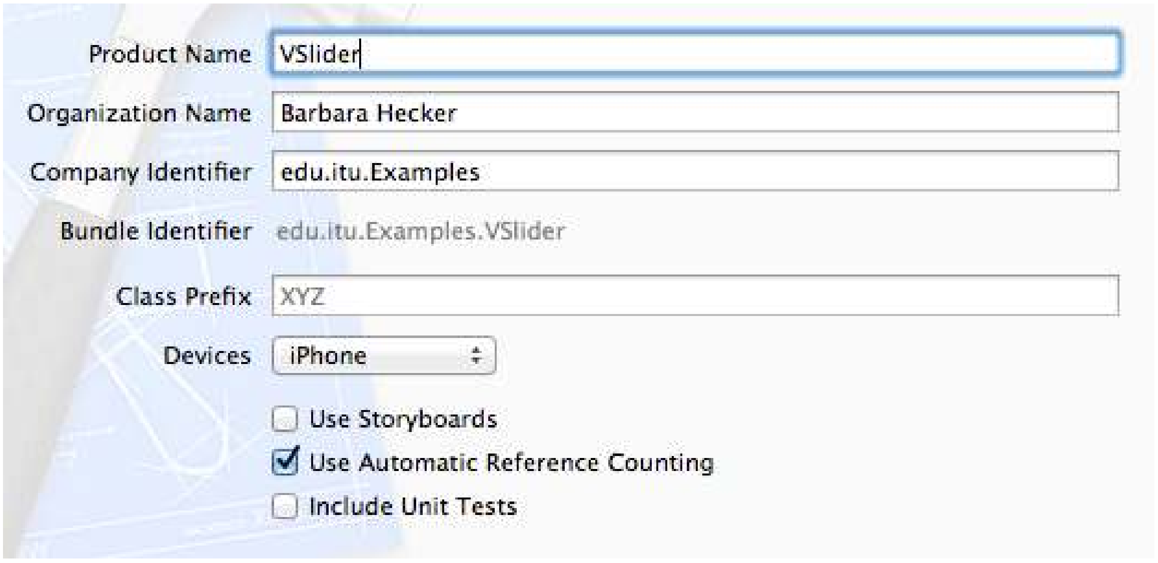
A Vertical Slider for iPhone

The UISlider control offers a way to continuously get values from the user within a range of set values. In the Interface Builder library of controls, there is only a horizontal version of the slider, with no obvious way to change its orientation. In this tutorial, you will learn how to rotate the slider control to a vertical orientation.

Step 1: Start Xcode, choose "Create a new Xcode project," and select the Single View Application template. Click Next, name the project VSlider and choose options as shown:

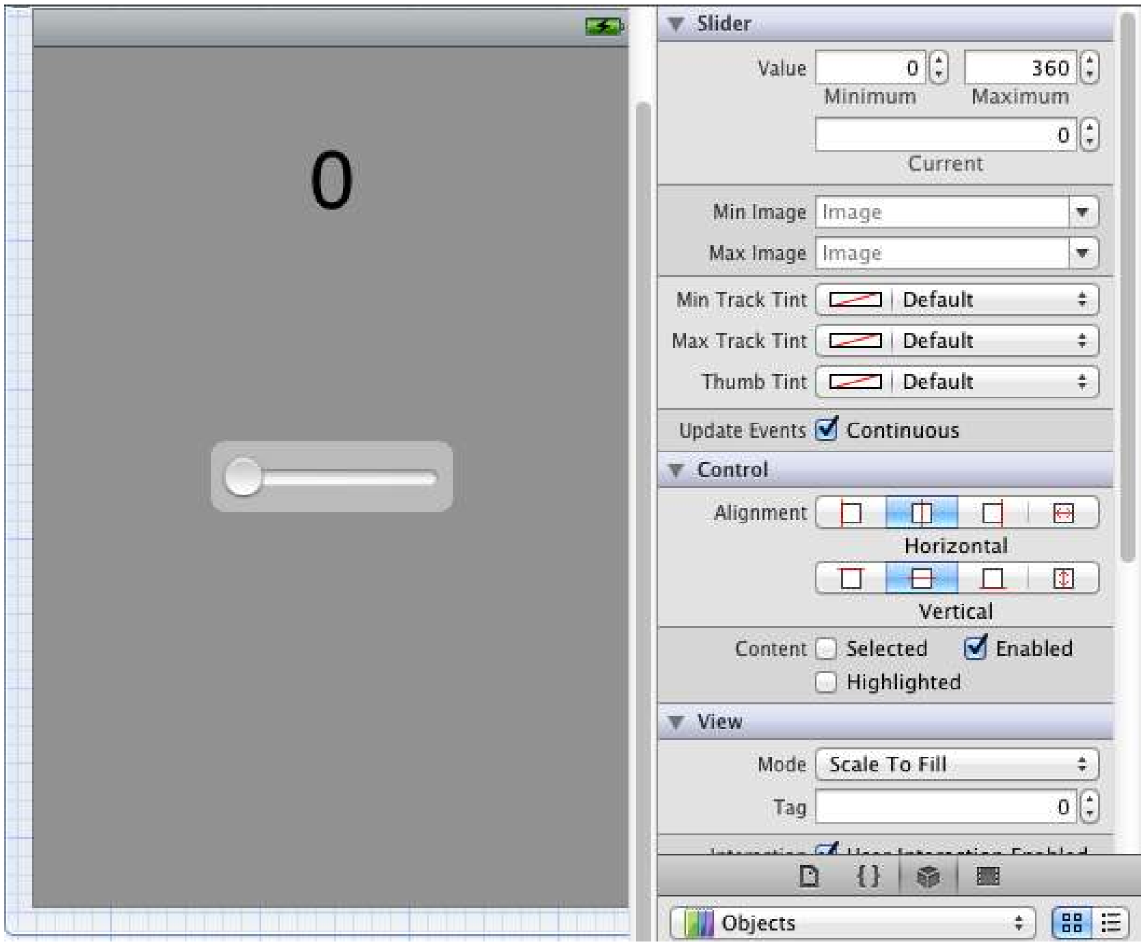


Product Name	VSlider
Organization Name	Barbara Hecker
Company Identifier	edu.itu.Examples
Bundle Identifier	edu.itu.Examples.VSlider
Class Prefix	XYZ
Devices	iPhone

☐ Use Storyboards
☑ Use Automatic Reference Counting
☐ Include Unit Tests

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

Step 2: Open **ViewController.xib**, and place a slider control on the view. Adjust the Minimum, Maximum, and Current values to 0, 360, and 0 as shown below. Also drag a UILabel control to the view, stretch it out and center it, and set the text to 44 points. Change the text in the label to "0", and center the text in the label.



Slider

Value: Minimum 0, Maximum 360, Current 0

Min Image: Image

Max Image: Image

Min Track Tint: Default

Max Track Tint: Default

Thumb Tint: Default

Update Events: [✓ Continuous]

Control

Alignment: Horizontal, Vertical

Content: Selected [✓ Enabled] Highlighted

View

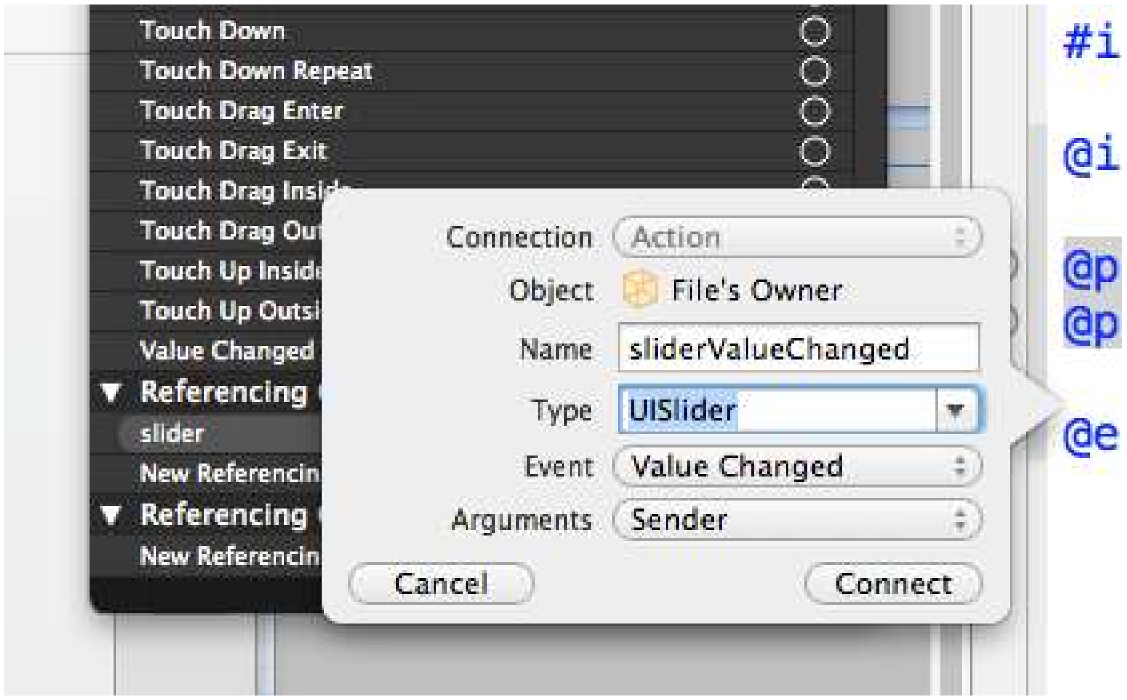
Mode: Scale To Fill

Tag: 0

Step 3: Open **ViewController.h** and wire the label and the slider as follows:

```
@property (weak, nonatomic) IBOutlet UISlider *slider;
@property (weak, nonatomic) IBOutlet UILabel *display;
```

Also wire the “ValueChanged” of the slider to an IBAction as follows:



Connection: Action
Object: File's Owner
Name: sliderValueChanged
Type: UISlider
Event: Value Changed
Arguments: Sender

```
- (IBAction)sliderValueChanged:(UISlider *)sender;
```

We added outlet properties for both the slider and the label. Normally, we would not add an outlet for the slider but in this application we want to change some properties of the slider object (make it vertical) so we need an outlet to do that. We also added a method to detect a change in the value of the slider control.

Step 4: Open **ViewController.m** and make these changes to the file:

Synthesize the two properties as follows:

```
@synthesize slider, display;
```

Add the following method:

```
    - (float)degrees2radians:(float)d
    {
        float r = d * (M_PI / 180);
        return r;
    }
```

Implement the **sliderValueChanged** method as follows:

```
- (IBAction)sliderValueChanged:(UISlider *)sender {
   self.display.text = [NSString stringWithFormat:@"%d",
(int)sender.value];
   CGAffineTransform labelRotation = CGAffineTransformIdentity;
   labelRotation = CGAffineTransformRotate(labelRotation, [self
degrees2radians:sender.value]);
   self.display.transform = labelRotation;
}
```

Add the following lines to the viewDidLoad method to make the slider vertical in appearance:

```
- (void)viewDidLoad
{
   [super viewDidLoad];
        CGAffineTransform sliderRotation =
CGAffineTransformIdentity;
   sliderRotation = CGAffineTransformRotate(sliderRotation, -(M_PI /
2));
   self.slider.transform = sliderRotation;
}
```

After synthesizing the properties, we define a helper method, degrees2radians:, that converts a floating point number of degrees to radian measure. As humans, we tend to think of angles in degrees, but radian measure is used exclusively in most (if not all) graphics libraries. The conversion is simple: 360 degrees == 2π radians, so 1 radian == 2π / 360 degrees. Factoring out the 2 gives us π / 180. We can therefore convert degrees to radians by multiplying the number of degrees with π / 180.

Look next at viewDidLoad: (we'll come back to sliderValueChanged: in a moment). After calling through to super, we apply an affine transform to the slider's underlying view. This is possible because a UISlider (like all controls) is a descendent of UIView: UISlider ← UIControl ← UIView. As we already know, views can be transformed in a number of ways: they can be moved, scaled, and rotated.

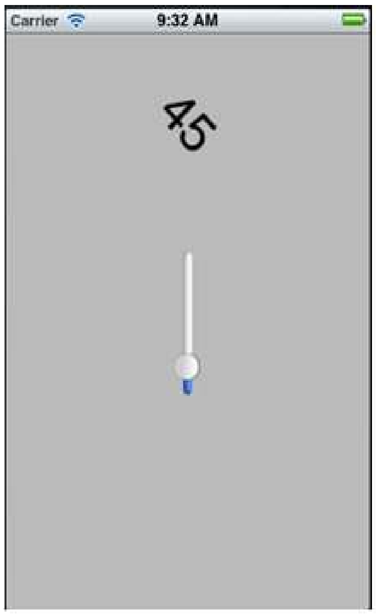
First, we get a transform matrix by setting a local variable to the identity matrix. We then alter this matrix by applying a rotation transform (CGAffineTransformRotate) to it through -(π / 2) radians (-

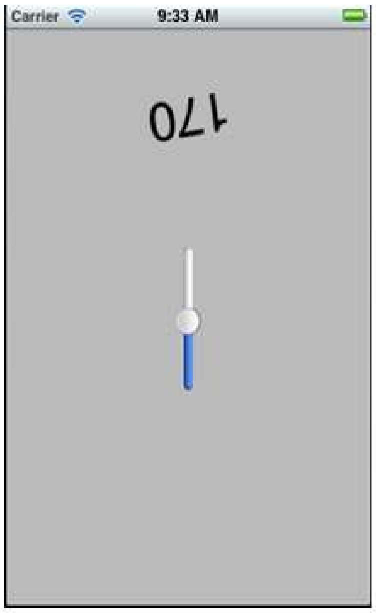
90 degrees). This value is then assigned to the transform property of the slider, which rotates the slider by 90 degrees in the counterclockwise direction. Higher values on the slider will therefore be "up."

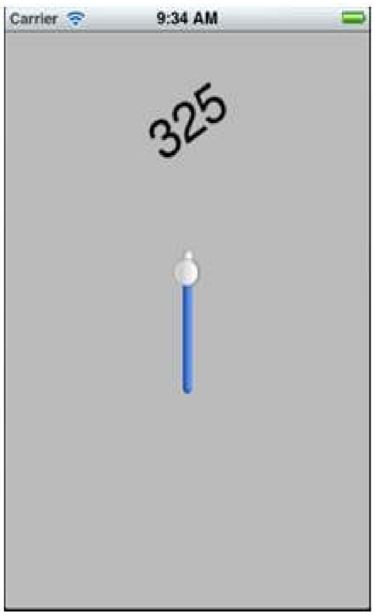
The default center of rotation for a UIView object is the center of the view, calculated as the width of the view / 2 and the height of the view / 2. This transform therefore performs rotation about the center of the control. (If we want to rotate about another point, we must set the rotation point on the view's layer.)

In sliderValueChanged:, we set the text of the label to the value of the slider as an NSString, formatted with the slider's value cast to the integer type. We then apply a rotational transform to the label, converting the slider's value (degrees) into radian measure.

Step 5: Run the application, and note that both the text and angle of the label change as the slider thumb is moved up and down:

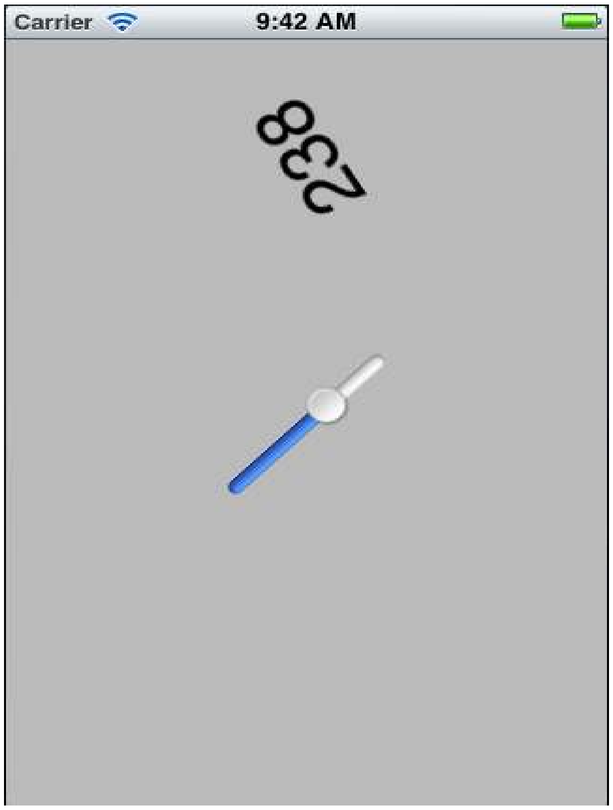






(Of course, the slider itself is oriented vertically, which is the whole point of the tutorial.)

There is nothing that says the orientation of any control can't be at an arbitrary angle, for example, we could have oriented the slider at 45 degrees:



But such orientations are confusing to the user: changing in two directions at once (both horizontally and vertically) implies a change in two independent variables. Unless there is a very good reason to do this, keep the slider in either the default horizontal orientation, or the user will become quite confused or disoriented.