|  |  |  |
| --- | --- | --- |
| |  | | --- | | A New Algorithm  The new algorithm goes like this:   1. **Subtract the target value from the slider’s value.** 2. **If the result is a negative number, then multiply it by -1 to make it a positive number.**   Here you no longer avoid the negative number since computers can work just fine with negative numbers. You simply turn it into a positive number.  And now it's time for a challenge! Try converting the above algorithm into source code.  Hint: the English description of the algorithm contains the words “if” and “then”, which is a pretty good indication you’ll have to use an if statement.  **Challenge:** If you haven't already, stop reading this email and try writing the source code for the above algorithm. If you'd like to compare what you wrote with ours, [stop by the discussion thread](https://t.dripemail2.com/c/eyJhY2NvdW50X2lkIjoiMjY0MDEzMCIsImRlbGl2ZXJ5X2lkIjoiNnp2aTA0ejZqcDBzYzN6ZXJtOWciLCJ1cmwiOiJodHRwczovL2ZvcnVtcy5yYXl3ZW5kZXJsaWNoLmNvbS90L3BhcnQtOC1hLW5ldy1hbGdvcml0aG0vMzg5Mjg_X19zPW1nYXpodDlyNDZtMHM2N2Z3aHFvIn0) on our forums. We'd love to see what you came up with! :]  Use the new algorithm  Ready to try out the code you just wrote? Great!  To start, open up the Bull's Eye project where you left it off last time (or download the starter project from the corresponding forum [discussion thread](https://t.dripemail2.com/c/eyJhY2NvdW50X2lkIjoiMjY0MDEzMCIsImRlbGl2ZXJ5X2lkIjoiNnp2aTA0ejZqcDBzYzN6ZXJtOWciLCJ1cmwiOiJodHRwczovL2ZvcnVtcy5yYXl3ZW5kZXJsaWNoLmNvbS90L3BhcnQtOC1hLW5ldy1hbGdvcml0aG0vMzg5Mjg_X19zPW1nYXpodDlyNDZtMHM2N2Z3aHFvIn0)).  Open **ViewController.swift**, and replace the old stuff at the top of showAlert() as follows:  @IBAction func showAlert() {  var difference = currentValue - targetValue  if difference < 0 {  difference = difference \* -1  }  let message = . . .  }  When you run this new version of the app (try it!), it should work exactly the same as before. The result of the computation does not change, only the technique used changed.  Another variation  The final alternative algorithm I want to show you uses a function.  You’ve already seen functions a few times before. For example, when you used lroundf() for rounding off the slider’s decimals.  Similarly, to make sure a number is always positive, you can use the abs() function.  If you took math in school you might remember the term “absolute value”, which is the value of a number without regard to its sign.  That’s exactly what you need here, and the standard library contains a convenient function for it, which allows you to reduce this entire algorithm down to a single line of code:  let difference = abs(targetValue - currentValue)  It really doesn’t matter whether you subtract currentValue from targetValue or the other way around. If the number is negative, abs() turns it positive. It’s a handy function to remember.  ➤ Make the change to showAlert() and try it out:  @IBAction func showAlert() {  let difference = abs(targetValue - currentValue)  let message = . . .  }  It doesn’t get much simpler than that!  What’s the score?  Now that you know how far off the slider is from the target, calculating the player’s score for each round is easy.  ➤ Change showAlert() to:  @IBAction func showAlert() {  let difference = abs(targetValue - currentValue)  let points = 100 - difference  let message = "You scored \(points) points"  . . .  }  The maximum score you can get is 100 points if you put the slider right on the target and the difference is zero. The further away from the target you are, the fewer points you earn.  ➤ Run the app and score some points!    The total score  In this game, you want to show the player’s total score on the screen. After every round, the app should add the newly scored points to the total and then update the score label.  Store the total score  Because the game needs to keep the total score around for a long time, you will need an instance variable.  ➤ Add a new score instance variable to **ViewController.swift**:  class ViewController: UIViewController {  var currentValue: Int = 0  var targetValue = 0  var score = 0 // add this line  Again, we make use of type inference to not specify a type for score.  **Note**: If you are not sure about the inferred type of a variable, there is an easy way to find out. Simply hold down the **Alt** / Option key and hover your cursor over the variable in question. The variable will be highlighted in blue and your cursor will turn into a question mark. Now, click on the variable and you will get a handy pop up which tells you the type of the variable as well as the source file in which the variable was declared.    Now that you're using type inference, you can clean up currentValue as well (and make its initial value 0, if you haven't already):  var currentValue = 0  var targetValue = 0  Thanks to type inference, you only have to specify the data type when you’re not giving the variable an initial value. But most of the time, you can safely make Swift guess at the type.  I think type inference is pretty sweet! It will definitely save you some, uh, typing (in more ways than one!).  Update the total score  Now showAlert() can be amended to update this score variable.  ➤ Make the following changes:  @IBAction func showAlert() {  let difference = abs(targetValue - currentValue)  let points = 100 - difference  score += points // add this line  let message = "You scored \(points) points"  . . .  }  Nothing too shocking here. You just added the following line:  score += points  This adds the points that the user scored in this round to the total score. You could also have written it like this:  score = score + points  Personally, I prefer the shorthand += version, but either one is okay. Both accomplish exactly the same thing.  Display the score  To display your current score, you’re going to do the same thing that you did for the target label: hook up the score label to an outlet and put the score value into the label’s text property.  Add this line to **ViewController.swift**:  @IBOutlet weak var scoreLabel: UILabel!  Then you connect the relevant label on the storyboard (the one that says 999999) to the new scoreLabel outlet.  Unsure how to connect the outlet? There are several ways to make connections from user interface objects to the view controller’s outlets:   * Control-click on the object to get a context-sensitive popup menu. Then drag from New Referencing Outlet to View Controller (you did this with the slider). * Go to the Connections Inspector for the label. Drag from New Referencing Outlet to View Controller (you did this with the target label). * Control-drag **from** View Controller to the label (give this one a try now) - doing it the other way, Control-dragging from the label to the view controller, won’t work.   There is more than one way to skin a cat, or, connect outlets. :]  Great, that gives you a scoreLabel outlet that you can use to display the score. Now where in the code can you do that? In updateLabels(), of course.  ➤ Back in **ViewController.swift**, change updateLabels() to the following:  func updateLabels() {  targetLabel.text = String(targetValue)  scoreLabel.text = String(score) // add this line  }  Nothing new here. You convert the score – which is an Int – into a String and then pass that string to the label’s text property. In response to that, the label will redraw itself with the new score.  ➤ Run the app and verify that the points for this round are added to the total score label whenever you tap the button.    One more round…  Speaking of rounds, you also have to increment the round number each time the player starts a new round.  And that leads me to your next challenge - wow, two in one email! :]  Your challenge is to keep track of the current round number (starting at 1) and increment it when a new round starts. Display the current round number in the corresponding label.  I may be throwing you into the deep end here, but if you’ve been able to follow the instructions so far, then you’ve already seen all the pieces you will need to pull this off. Good luck!  **Challenge:** If you haven't already, stop reading this email and try keeping track of (and displaying) the current round. If you'd like to compare what you wrote with ours, [stop by the discussion thread](https://t.dripemail2.com/c/eyJhY2NvdW50X2lkIjoiMjY0MDEzMCIsImRlbGl2ZXJ5X2lkIjoiNnp2aTA0ejZqcDBzYzN6ZXJtOWciLCJ1cmwiOiJodHRwczovL2ZvcnVtcy5yYXl3ZW5kZXJsaWNoLmNvbS90L3BhcnQtOC1hLW5ldy1hbGdvcml0aG0vMzg5Mjg_X19zPW1nYXpodDlyNDZtMHM2N2Z3aHFvIn0) on our forums. We'd love to see what you came up with! :]  You're making great progress, well done! | |  | |