Veculator Debugging Development Journal

Orion

To-Do-List

1. Previous View Bar cannot accurately work ✔
2. The direction result might be inaccurate because of Double type calculation. For example, 7 might be showing 6.9999999999 ✔
3. When pressing "AC" button, it will get 0.0[nah]. Before solving this, I should do a little bit research about the meaning of nah. ✔
4. Answer sometimes can be too long, and the Bar cannot show the full result. ✔
5. User Interface needs further adjustment. I'm thinking red as the theme colour, and Put all the touchDigits() buttons on the bottom, with smaller sized performOperation() buttons on top, and then the display bars. ✔
6. When pressing square root button √, it does not get the right degree. The direction will always appear [0.0]. ✔

Journal #1 March 12, 2017

I am going to make my previous view bar work properly. I am trying to make it appear

2 + … if user enters 2+;

2 + 7 = if user enters 2+7=;

√(4[210]) if user enters 4[210]√.

I will add two more variables in my Model. Var description to store the string, and var isPartialResult to store a boolean value.

Journal #2 March 12, 2017

While I was working on my previous view bar, I found the dot sign isn't working properly either. It adds the dot "." at the end of direction sign "[12]." 👈 like this. It's easy to fix because it's obvious that there's something wrong in my if statement of touchDigit() function. There are too many if-else. I want to make them look neater.

I replaced the original function for solely entering dot and direction with a switch statement in touchDigit(). Codes are neater, and it worked just the way it did before.

I also change the layout of my User Interface. Direction button moved to the bottom, and it makes more sense because all the buttons that are bond with touchDigit() are together in the middle, and all the buttons that are bond with performerOperation() surrounds them.

Journal #3 March 13, 2017

Previous View Bar is working almost correctly, but when I entered 2[27] + 3[32] =, it appeared "2[27] + …" and then when I pressed "=" button, it appeared "3[32] ="

I believe the problem comes from the fact that I reset the description variable every time when accumulator is reassigned. I guess I only have to adjust a little bit about that.

Journal #4 March 14, 2017

Previous View Bar is nearly working right. It shows up equal sign "=" when I pressed AC. The reason for this error is obvious because every time I get the string value for Previous View Bar, I appends it with “ =”, when it is partial result. Therefore, when I pressed AC, it gets a “ =” at the end because of it is also a partial result.

I will set the variable that store whether it is a partial result from boolean to an optional type that binds with boolean value. When the value is nil, then outputs “0”, and I will set the isPartialResult as nil in the Reset enum case.

It worked out perfectly. XD

Journal #5 March 16, 2017

There might be some small errors in the direction result due to floating point calculation. Also, answer sometimes can be too long, and the bar cannot show the full result.

I round the result to six-decimal spaces, and set the font on the bar auto-shrinkable. The minimum font size is now 20.

It worked.

Journal #6 March 16, 2017

When pressing “AC” button, it will get 0.0[nah]. Before solving this, I should do a little bit research about the meaning of nah.

I believe the reason is that I reset the accumulator as (0.0, 0.0) in Model when user presses AC button. When converted to value, there will be a division accumulator.0 / accumulator.1. Denominator should never be zero, or there will be a math error. Therefore, this problem will not only appear when user presses AC, but it will also show up whenever the denominator equals to zero.

Journal #7 March 19, 2017

Luckily that Double type in swift has a getter function itself. `.isNaH` to get the isNaH property. Therefore, every time the direction gets nah, it will turn into 0.0, and then I will let my switch function to determine what angles it should be according to the y value.

Journal #8 March 21, 2017

I was going to do some refinements on my UI layout, but before doing this, I did some random testing. I found the square root function does not work correctly. It cannot get the right direction. It always gets [0.0].

Reason for this is that I thought square root function should return a scalar value instead of a vector value. Therefore, I set the direction as 0 by default.

```
"√": Operation.UnaryOperation({
        (sqrt(sqrt($0.0 * $0.0 + $0.1 * $0.1)), 0.0)
    })
```

Small adjustments will be needed for my square root function.

Because every value will be transformed into a coordinate value, instead of direction and magnitude, and it will be transformed back. I am trying to do something on the coordinate value to get both the correct magnitude and direction.

The technique that I'm going to use is similar triangles. If I only want the hypotenuse to be square rooted, but the two other sides remain the same, the two triangles are similar triangles. Therefore, their three sides are in ratios.

ri / rf = xi / xf —> ri / √ri = xi / xf —> xf = 1 / √r * xi

Journal #9 March 22, 2017

The square root function is now

```
"√": Operation.UnaryOperation({
    let ratio = 1.0 / sqrt($0.0 * $0.0 + $0.1 * $0.1)
    let result = ($0.0 * ratio, $0.1 * ratio)
    return result
  })
```

However, this does not work well. The final value is always 1.0[0.0]. Reasons are unknown to me, and I am adding more print() function to find out more.

————10 minites later————

Reason was found! I missed one sqrt() function

```
let ratio = 1.0 / sqrt(sqrt($0.0 * $0.0 + $0.1 * $0.1))
```

After changing this, the magnitude and direction of the vector are both working properly.

Journal #10 March 27, 2017

I redesigned the whole UI. Theme colour is now Teal. It looks better to me.

Also, I move all the touchDigits button to the bottom, with all the performOperation buttons in middle and two display bars on the top.

Journal #11 March 28, 2017

When testing, I found that when turn the display from portrait into landscape, the font of the display bar is too big, so it cannot show the full content.

I have several options for solving this:

1. Delete the previous display bar
2. Change the font
3. Make the previous display only shows content when user has a partial result (Has only entered one variable for binary calculations).

Journal #12 March 31, 2017

I decided to make previous view bar only show contents, when there is a partial result. For example, it will appear

"2 + …" if user enters 2+;

"" if user enters 2+7=;

"" if user enters 4[210]√;

"π" if user enters π (Although there is no π).

Therefore, there will be more space for the content.

Due to the silly swift UI design the default font size of previous display is 1, and the current display is 95.

When the previous display is activated, the previous display font size will be 20, and the current display font size will be 65.

Journal #13 March 31, 2017

For I don’t know what reason, the font size switching does not work well. Aside from this, it is still showing “x[a] + y[b] =“ in the previous view bar.

Therefore, there is no need for var isPartialResult. Now, the previous view content getter becomes like this:

```
var previouslyEntered: String {
    get {
        return description == " " ? " " : description + " ..."
    }
```

Initial font size for current display bar is now 70, and font is thin.

No further changes.

Journal #14 April 1, 2017

I added an extension to Double value A accessor to get the scientific value of a Double value.

This applies to every double value in the two display bars, when the double value is larger than 10_000_000.

```
        extension Double {
    struct Number {
        static var formatter = NumberFormatter()
    }
    var scientificStyle: String {
        Number.formatter.numberStyle = NumberFormatter.Style.scientific
        Number.formatter.positiveFormat = "0.###E+0"
        Number.formatter.exponentSymbol = "e"
        return Number.formatter.string(from: NSNumber(value: self))!
            //Number.formatter.stringFromNumber(NSNumber(value: self) ?? description
    }
}
```

Journal #15 April 1, 2017

New app icon added.

Journal #16 April 1, 2017

Subtraction method does not work due to the wrong symbol for subtraction. I used the hyphen-minus instead of minus sign. I changed it back.

I added a new variable, lastPressedButtonName, into the view to keep track of the last pressed button. If a perform operation button is pressed twice, it won't start performing now.

Journal #17 April 1, 2017

Fixed all the noticeable small errors.

Added a new extension variable to Double value to check whether it's an integer.

```
var isInteger: Bool {
    if self == floor(self) {
        return true
    }
    return false
}
```

The view will check whether the number is a integer or not, before updating it to the two display bars. Therefore, there is no "0.0" again.

Changed the isLastPressedButtonAOperation variable to a Bool value. In BinaryOperation, before executingPending(), it will now check if pending is a nil. If it is not, it will update the description. This prevents user keeps pressing “+ 5 + 5 + 5”, but the previous view bar still shows “5[0]” instead of “10[0]”

Journal #18 April 1, 2017

I used the wrong function, when convert the accumulator to value to set the previous view bar. I used valueToCoordinate which is supposed to be coordinateToValue.

Journal #19 April 4, 2017

I made one mistake. I made every number to come out as an integer.

I fixed the problem.

Journal #20 April 5, 2017

Fixed the scientific style which did not show up.