```swift
//
//  CalculatorViewController.swift
//  Kintematics Calculator
//
//  Created by Luke Deratzou on 6/4/18.
//  Copyright © 2018 Luke Deratzou. All rights reserved.
//
//Idea: do not initially have any labels or textfields; instead just the view. Then, whenever the view loads, it checks which calc is being loaded, and creates the necessary number of labels & textfields based on that.
//Since I am doing the more confusing (at first) method for calculating, find way to add a help function. Whether it be a first time only popup, a popup that comes up indefinitly until turned off manually, or a simple "?" or "Help" button somewhere on the screen. Former two would be better imo, but harder to implement... Could also ask friends/Mr. Yorgey the value of having said helpful message or if its just a nussaince.


// Can have an alert pop up that tells users how many values they have to enter before they can calculate a value... but have it have an option to never show this again.
//could also, instead, have a label that lets user know how many values they need to input that changes based on # of values that they have entered.
//can be at ANSWER location... and it can say "Input x more values..."


import UIKit

class CalculatorViewController: UIViewController, UITextFieldDelegate, UIPickerViewDelegate, UIPickerViewDataSource {

    @IBOutlet weak var varOneLabel: UILabel!
    @IBOutlet weak var varTwoLabel: UILabel!
    @IBOutlet weak var varThreeLabel: UILabel!
    @IBOutlet weak var varFourLabel: UILabel!
    @IBOutlet weak var varFiveLabel: UILabel!
    @IBOutlet weak var answerLabel: UILabel!
    @IBOutlet weak var calculateBtn: UIButton!
    @IBOutlet weak var varOneTextField: UITextField!
    @IBOutlet weak var varTwoTextField: UITextField!
    @IBOutlet weak var varThreeTextField: UITextField!
    @IBOutlet weak var varFourTextField: UITextField!
    @IBOutlet weak var varFiveTextField: UITextField!
    @IBOutlet weak var calculatorTitleLabel: UILabel!
    @IBOutlet weak var varView: UIView!
```

```
@IBOutlet weak var moreBtn: UIButton!
@IBOutlet weak var settingsBtn: UIButton!

@IBOutlet weak var saveQuestionBtn: UIButton!


var toPass: String = ""
var listOfVars: [PhysicsVariable] = [PhysicsVariable]()

var bottomPickerView: UIView!
var listOfButtons: [UIButton] = [UIButton]()
var tempPickerString: String = ""
var equationName: String = "N/A"

var savedName = "Saved Problem
 \(Int(UserDefaults.standard.getSavedProblemCounter()))"

var textField: UITextField?

var exitHelpMode: Bool = false

var viewDidAppearAlready = false

override func viewDidLoad() {
    super.viewDidLoad()
    print("viewdidload")
    formatButtonsAndLabels()
    for i in Helper.GET_LIST_OF_EQS() {
        if toPass.contains(i) {
            equationName = i
            toPass = i
        }
    }
    if Helper.MODE == "Help" {
        answerLabel.isHidden = true
        helpMode()
        adjustLabelsAndFields()
        return
    }
    saveQuestionBtn.isEnabled = false
    setUpCalculator()
    if UserDefaults.standard.getProblemTypePP() == "None" {
        UserDefaults.standard.setCurrentPhysicsEqPP(value: toPass)
    }
    self.hideKeyboardWhenTappedAround()
    varOneTextField.delegate = self
    varTwoTextField.delegate = self
```

```
        varThreeTextField.delegate = self
        varFourTextField.delegate = self
        varFiveTextField.delegate = self

        NotificationCenter.default.addObserver(self, selector:
         #selector(PracticeProblemsViewController
         .keyboardWillShow(sender:)), name:
         NSNotification.Name.UIKeyboardWillShow, object: nil)

        NotificationCenter.default.addObserver(self, selector:
         #selector(PracticeProblemsViewController
         .keyboardWillHide(sender:)), name:
         NSNotification.Name.UIKeyboardWillHide, object: nil)

        adjustLabelsAndFields()
        setUpUnitBtns()

        if calculatorTitleLabel.text?.contains("ravitational") ?? false
         {
            fixTitleLabel()

        }
        updateVarCount()

        // Do any additional setup after loading the view.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if Helper.MODE == "Help" {
            if exitHelpMode {
                Helper.MODE = "Normal"
            }
            return
        }
        switch segue.identifier {
        case "options":
            let svc = segue.destination as! OptionsViewController;
            svc.toPass = "\(toPass) calculator"
        case "settings":
            let svc = segue.destination as! SettingsViewController;
            svc.toPass = "\(toPass) calculator"
            //doubt i need this... svc.listOfVars = listOfVars
        default:
```

```
        print("error w/ preparing for segue")
    }
}

override func viewWillAppear(_ animated: Bool) {
    if viewDidAppearAlready {

    } else {
        fixVarNamesAndFields()
        viewDidAppearAlready = true
    }



}

@objc func keyboardWillShow(sender: NSNotification) {
    if varFiveTextField.isFirstResponder {
        self.view.frame.origin.y = -100 // Move view 150 points
         upward
    }

}

@objc func keyboardWillHide(sender: NSNotification) {
    self.view.frame.origin.y = 0 // Move view to original position
}

func textFieldShouldReturn(_ textField: UITextField) -> Bool {
    self.view.endEditing(true)
    return false
}


func fixTitleLabel() {
    calculatorTitleLabel.text = "Gravitational Force"
}

func updateVarCount() {
    let listOfFields: [UITextField] = [varOneTextField,
     varTwoTextField, varThreeTextField, varFourTextField,
     varFiveTextField]
    var counter = 0
    for i in listOfFields {
        if i.hasText {
            counter += 1
        }
    }
```

```
    switch toPass {
    case "kinematics":
        counter = 3 - counter
    case "forces":
        counter = 2 - counter
    case "kinetic energy":
        counter = 2 - counter
    case "gravitational force":
        counter = 3 - counter
    default:
        print("ERROR- toPass invalid")
    }
    answerLabel.text = "Enter \(counter) more values"
    if counter == 1 {
        answerLabel.text?.removeLast()
    }
}

func formatButtonsAndLabels() {

    var cornerRadius: CGFloat = 10
    if self.view.frame.width > 500 {
        cornerRadius = 25
    }
    varOneTextField.layer.masksToBounds = true
    varOneTextField.layer.cornerRadius = cornerRadius
    varTwoTextField.layer.masksToBounds = true
    varTwoTextField.layer.cornerRadius = cornerRadius
    varThreeTextField.layer.masksToBounds = true
    varThreeTextField.layer.cornerRadius = cornerRadius
    varFourTextField.layer.masksToBounds = true
    varFourTextField.layer.cornerRadius = cornerRadius
    varFiveTextField.layer.masksToBounds = true
    varFiveTextField.layer.cornerRadius = cornerRadius
    answerLabel.layer.masksToBounds = true
    answerLabel.layer.cornerRadius = cornerRadius
    varView.layer.cornerRadius = cornerRadius
    calculatorTitleLabel.layer.masksToBounds = true
    calculatorTitleLabel.layer.cornerRadius = cornerRadius


    let isIphoneX = Helper.IS_IPHONE_X()
    let length: CGFloat =
     CGFloat(UserDefaults.standard.getButtonSize())
    if isIphoneX {
        moreBtn.frame = CGRect(x: moreBtn.frame.minX, y: 42, width:
         length, height: length)
```

```
            settingsBtn.frame = CGRect(x: settingsBtn.frame.minX, y:
             42, width: length, height: length)
        } else {
            moreBtn.frame = CGRect(x: moreBtn.frame.minX, y:
             moreBtn.frame.minY, width: length, height: length)
            settingsBtn.frame = CGRect(x: settingsBtn.frame.minX, y:
             settingsBtn.frame.minY, width: length, height: length)
        }
        fixCalculateAndSaveBtn()
    }

    func adjustLabelsAndFields() {
        var labelList: [UILabel] = [varOneLabel, varTwoLabel,
         varThreeLabel, varFourLabel, varFiveLabel]
        var textFieldList: [UITextField] = [varOneTextField,
         varTwoTextField, varThreeTextField, varFourTextField,
         varFiveTextField]
        for i in 0...4 {
            let yPos: CGFloat = labelList[i].frame.midY -
             textFieldList[i].frame.height/2
            textFieldList[i].frame = CGRect(x:
             textFieldList[i].frame.minX, y: yPos, width:
             textFieldList[i].frame.width, height:
             textFieldList[i].frame.height)
        }
    }

    func fixCalculateAndSaveBtn() {
        //calculate w: 120
        //calculate h: 24
        if calculateBtn.frame.width / 120 > calculateBtn.frame.height /
         24 {
            let newWidth: CGFloat = calculateBtn.frame.height * (120/24)
            calculateBtn.frame = CGRect(x: calculateBtn.frame.minX, y:
             calculateBtn.frame.minY, width: newWidth, height:
             calculateBtn.frame.height)
        } else {
            let newHeight: CGFloat = calculateBtn.frame.width * (24/120)
            calculateBtn.frame = CGRect(x: calculateBtn.frame.minX, y:
             calculateBtn.frame.minY, width: calculateBtn.frame.width,
             height: newHeight)
        }

        if saveQuestionBtn.frame.width / 60 >
         saveQuestionBtn.frame.height / 20 {
            let newWidth: CGFloat = saveQuestionBtn.frame.height *
             (60/20)
```

```
            saveQuestionBtn.frame = CGRect(x:
             saveQuestionBtn.frame.minX, y: saveQuestionBtn.frame.minY,
             width: newWidth, height: saveQuestionBtn.frame.height)
        } else {
            let newHeight: CGFloat = saveQuestionBtn.frame.width *
             (20/60)
            saveQuestionBtn.frame = CGRect(x:
             saveQuestionBtn.frame.minX, y: saveQuestionBtn.frame.minY,
             width: saveQuestionBtn.frame.width, height: newHeight)
        }
    }

    func fixVarNamesAndFields() {
        let fontSize1 = self.varOneLabel.getFontSizeForLabel()
        let fontSize2 = self.varTwoLabel.getFontSizeForLabel()
        let fontSize3 = self.varThreeLabel.getFontSizeForLabel()
        let fontSize4 = self.varFourLabel.getFontSizeForLabel()
        //let fontSize5 = self.varFiveLabel.getFontSizeForLabel()

        var smallestFontSize = min(min(fontSize1, fontSize2),
         min(fontSize3, fontSize4)) - 4
        if Helper.IS_IPHONE_X() {
            smallestFontSize = smallestFontSize - 4
        }
        var factor:CGFloat = 1.0
        if toPass == "forces" {
            factor = 0.8
        }
        self.varOneLabel.font =
         self.varOneLabel.font.withSize(smallestFontSize)
        self.varTwoLabel.font =
         self.varTwoLabel.font.withSize(smallestFontSize)
        self.varThreeLabel.font =
         self.varThreeLabel.font.withSize(smallestFontSize)
        self.varFourLabel.font =
         self.varFourLabel.font.withSize(smallestFontSize)
        self.varFiveLabel.font =
         self.varFiveLabel.font.withSize(smallestFontSize)

        self.varOneTextField.font =
         self.varOneTextField.font?.withSize(smallestFontSize*factor)
        self.varTwoTextField.font =
         self.varTwoTextField.font?.withSize(smallestFontSize*factor)
        self.varThreeTextField.font =
         self.varThreeTextField.font?.withSize(smallestFontSize*factor)
        self.varFourTextField.font =
         self.varFourTextField.font?.withSize(smallestFontSize*factor)
```

```
        self.varFiveTextField.font =
         self.varFiveTextField.font?.withSize(smallestFontSize*factor)

        self.varOneLabel.adjustsFontSizeToFitWidth = false
        self.varTwoLabel.adjustsFontSizeToFitWidth = false
        self.varThreeLabel.adjustsFontSizeToFitWidth = false
        self.varFourLabel.adjustsFontSizeToFitWidth = false
        self.varFiveLabel.adjustsFontSizeToFitWidth = false
    }

    func setUpCalculator() {
        UserDefaults.standard.setCurrentPhysicsEqPP(value: toPass)
        calculateBtn.isEnabled = false
        hideStuff()
        switch true {
        case toPass.contains("kinematics"):
            calculatorTitleLabel.text = "Kinematics Calculator"
            toPass = "kinematics"
        case toPass.contains("forces"):
            calculatorTitleLabel.text = "Forces Calculator"
            toPass = "forces"
        case toPass.contains("kinetic energy"):
            calculatorTitleLabel.text = "Kinetic Energy Calculator"
            toPass = "kinetic energy"
        case toPass.contains("gravitational force"):
            calculatorTitleLabel.text = "Gravitational Force Calculator"
            toPass = "gravitational force"
            if self.view.frame.width < 500 {
                varFourLabel.numberOfLines = 2
                varFourLabel.frame = CGRect(x:
                 varThreeLabel.frame.minX, y:
                 varFourTextField.frame.minY, width:
                 varThreeLabel.frame.width, height:
                 varThreeLabel.frame.height * 2)
            }
        default:
            print("error w/ toPass")
        }
        setUpLabelsAndFields(toPass)
    }

    func setUpLabelsAndFields(_ temp: String) {
        var tempList: [String] = Helper.GET_LIST_OF_VARS(eq: temp)
        var labelList: [UILabel] = [varOneLabel, varTwoLabel,
         varThreeLabel, varFourLabel, varFiveLabel]
        var textFieldList = [varOneTextField, varTwoTextField,
         varThreeTextField, varFourTextField, varFiveTextField]
```

```
    var longestLabel: Int = -1
    var labelSize: Int = 0
    for i in 0...tempList.endIndex - 1 {
        labelList[i].isHidden = false
        labelList[i].text = tempList[i]

        print(tempList[i])
        if tempList[i].count > labelSize {
            labelSize = tempList[i].count
            longestLabel = i
        }
        textFieldList[i]?.isHidden = false
        textFieldList[i]?.placeholder = tempList[i]
    }
    /*var labelX = UILabel(frame: CGRect(x: 0, y:0, width:
     labelList[longestLabel].frame.width,
     height:labelList[longestLabel].frame.height))
    labelX.font = UIFont(name: "Menlo", size: 100)
    labelX.text = labelList[longestLabel].text
    labelX.adjustsFontSizeToFitWidth = true
    labelX.minimumScaleFactor = 0.5
    self.view.addSubview(labelX)
    print(labelX.font)*/
    for j in 0...tempList.endIndex - 1 {
        labelList[j].font = labelList[longestLabel].font
    }
    //FIX THIS!!!
    //e. probably don't need this... (stuff above)
    if self.view.frame.width < 500 {
        for i in 0...tempList.endIndex - 1 {
            textFieldList[i]?.font = UIFont(name: "Menlo", size:12)
        }
    }

}

func setUpUnitBtns() {
    bottomPickerView = UIView(frame: CGRect(x: 0, y:
     self.view.frame.height - Helper.GET_BOTTOM_VIEW_HEIGHT(),
     width: self.view.frame.width, height:
     Helper.GET_BOTTOM_VIEW_HEIGHT()))
    bottomPickerView.backgroundColor = UIColor(displayP3Red:
     93/255, green: 188/255, blue: 210/255, alpha: 1)
    self.view.addSubview(bottomPickerView)
    bottomPickerView.isHidden = true
    //done button was here.....
    let tempList: [String] = Helper.GET_LIST_OF_VARS(eq: toPass)
```

```
let tempFieldList: [UITextField] = [varOneTextField,
 varTwoTextField, varThreeTextField, varFourTextField,
 varFiveTextField]
for i in 1...tempList.endIndex {
    let y:CGFloat = tempFieldList[i-1].frame.minY
    let unitWidth: CGFloat = varView.frame.width -
     varOneTextField.frame.maxX - 10
    let unitHeight: CGFloat = varOneTextField.frame.height
    let unitBtn:UIButton = UIButton(frame: CGRect(x:
     varView.frame.width - unitWidth - 5, y: y, width:
     unitWidth, height: unitHeight))
    unitBtn.setTitle(Helper.GET_SHORTENED_UNIT(unitName:
     Helper.GET_SI_UNIT(vName: tempList[i-1])), for: .normal)
    unitBtn.addTarget(self, action: #selector(showPickerView),
     for: .touchUpInside)
    switch Helper.GET_SHORTENED_UNIT(unitName:
     Helper.GET_SI_UNIT(vName: tempList[i-1])) {
    case "m/s":
        if i == 2 &&
         calculatorTitleLabel.text?.contains("inematics") ??
         false {
            unitBtn.tag = -2 //fv
        } else {
            unitBtn.tag = -1 //iv
        }
    case Helper.exponentize(str: "m/s^2"):
        unitBtn.tag = -3
    case "m":
        unitBtn.tag = -4
    case "s":
        unitBtn.tag = -5
    case "kg":
        if i == 2 &&
         calculatorTitleLabel.text?.contains("ravitational") ??
         false {
            unitBtn.tag = -6 //m2
        } else {
            unitBtn.tag = -7 //m1
        }
    case "N":
        unitBtn.tag = -8
    case "J":
        unitBtn.tag = -9
    default:
        print("ERRORA")
    }
    unitBtn.titleLabel?.font = UIFont(name: "Menlo", size:
     Helper.GET_FONT_SIZE()-1)
```

```
            self.varView.addSubview(unitBtn)
            listOfButtons.append(unitBtn)
        }
    }

    @objc func showPickerView(_ sender: UIButton) {
        bottomPickerView.tag = 111
        for i in self.view.subviews {
            if i.tag != 111 {
                i.isUserInteractionEnabled = false
            }
        }
        bottomPickerView.isHidden = false
        //delete the for loop... does not do anything
        for i in bottomPickerView.subviews {
            i.isHidden = false
        }
        /*var unitName: String = sender.titleLabel?.text ?? "nul"
        unitName = Helper.GET_FULL_UNIT_NAME(unitName: unitName)*/
        let unitPicker: UIPickerView = UIPickerView(frame: CGRect(x: 0,
         y: 0, width: self.bottomPickerView.frame.width, height:
         self.bottomPickerView.frame.height))
        unitPicker.delegate = self
        unitPicker.dataSource = self
        unitPicker.tag = sender.tag * -1
        self.bottomPickerView.addSubview(unitPicker)
        let doneBtn = DoneButton(frame: CGRect(x: self.view.frame.width
         - Helper.GET_DONE_BTN_WIDTH() - 5, y: 5, width:
         Helper.GET_DONE_BTN_WIDTH(), height:
         Helper.GET_DONE_BTN_HEIGHT()))
        doneBtn.tag = sender.tag
        doneBtn.addTarget(self, action: #selector(hidePickerView), for:
         .touchUpInside)
        self.bottomPickerView.addSubview(doneBtn)
        //have to check if the titleLabel.text of sender is equal to
         one of the Strings inside of one of the lists of units... for
         whichever it is... that is the unit list to display in
         pickerView... So it will be several for loops but also will
         need to convert the titleLabel.text into the long version of
         the units...
        //So... need to create some new lists and a function in
         PhysicsVariable or Helper that converts the shortened to the
         longer unit (unless it already is on my app)...
    }

    @objc func hidePickerView(_ sender: AnyObject) {
        for i in self.view.subviews {
            if i.tag != 111 {
```

```
                i.isUserInteractionEnabled = true
            }
        }
        bottomPickerView.isHidden = true
        var tempTag = -100
        for i in bottomPickerView.subviews {
            if i.tag < 0 {
                tempTag = i.tag
            }
            if let viewWithTag =
             self.bottomPickerView.viewWithTag(i.tag) {
                viewWithTag.removeFromSuperview()
            }

        }
        for i in listOfButtons {
            if i.tag == tempTag && !tempPickerString.isEmpty{
                let titleLabel: String =
                 Helper.GET_SHORTENED_UNIT(unitName: tempPickerString)
                i.setTitle(titleLabel, for: .normal)
                tempPickerString = ""
            }
        }

    }

    func hideStuff() {
        var labelList = [varOneLabel, varTwoLabel, varThreeLabel,
         varFourLabel, varFiveLabel]
        var textFieldList = [varOneTextField, varTwoTextField,
         varThreeTextField, varFourTextField, varFiveTextField]
        for i in 0...4  {
            labelList[i]?.isHidden = true
            textFieldList[i]?.isHidden = true
            textFieldList[i]?.text?.removeAll()
            //listOfButtons[i].isHidden = true
        }
    }
    //These are bad functions... I do not need to have 5 functions for
     5 fields... can instead have one function for all five fields...
     or simply make the fields be added programatically rather than
     storyboard wise...

    @IBAction func var1TFEndEditing(_ sender: UITextField) {
        checkFieldStatus()
    }

    @IBAction func var2TFEndEditing(_ sender: UITextField) {
```

```
    checkFieldStatus()
}

@IBAction func var3TFEndEditing(_ sender: UITextField) {
    checkFieldStatus()
}

@IBAction func var4TFEndEditing(_ sender: UITextField) {
    checkFieldStatus()
}

@IBAction func var5TFEndEditing(_ sender: UITextField) {
    checkFieldStatus()
}
@IBAction func var1TFBeginEditing(_ sender: UITextField) {
    calculateBtn.isEnabled = false
    varOneTextField.backgroundColor = UIColor.white
}
@IBAction func var2TFBeginEditing(_ sender: UITextField) {
    calculateBtn.isEnabled = false
    varTwoTextField.backgroundColor = UIColor.white
}
@IBAction func var3TFBeginEditing(_ sender: UITextField) {
    calculateBtn.isEnabled = false
    varThreeTextField.backgroundColor = UIColor.white
}
@IBAction func var4TFBeginEditing(_ sender: UITextField) {
    calculateBtn.isEnabled = false
    varFourTextField.backgroundColor = UIColor.white
}
@IBAction func var5TFBeginEditing(_ sender: UITextField) {
    calculateBtn.isEnabled = false
    varFiveTextField.backgroundColor = UIColor.white
}


func checkFieldStatus() {
    updateVarCount()
    var textFieldList = [varOneTextField, varTwoTextField,
     varThreeTextField, varFourTextField, varFiveTextField]
    var counter: Int = 0
    var numOfVars: Int!
    switch toPass {
    case "kinematics":
        numOfVars = 3
    case "forces":
        numOfVars = 2
    case "kinetic energy":
```

```
            numOfVars = 2
        case "gravitational force":
            numOfVars = 3
        default:
            print("error w/ toPass in checkFieldStatus")
        }
        for i in 0...4 {
            if (textFieldList[i]?.hasText)! {
                counter += 1
                if counter == numOfVars {
                    calculateBtn.isEnabled = true
                    for j in 0...4 {
                        if (!(textFieldList[j]?.hasText)!) {
                            textFieldList[j]?.isEnabled = false
                            textFieldList[j]?.backgroundColor =
                             UIColor.gray
                        }
                    }
                } else {
                    calculateBtn.isEnabled = false
                    enableOrDisableFields(enable: true)
                }
            }
        }

    }

    func enableOrDisableFields(enable: Bool) {
        var textFieldList = [varOneTextField, varTwoTextField,
         varThreeTextField, varFourTextField, varFiveTextField]
        for i in 0...4 {
            textFieldList[i]?.isEnabled = enable
            textFieldList[i]?.backgroundColor = UIColor.white
        }
    }

    func areInputsValid() -> Bool {
        var textFieldList = [varOneTextField, varTwoTextField,
         varThreeTextField, varFourTextField, varFiveTextField]
        var labelList = [varOneLabel, varTwoLabel, varThreeLabel,
         varFourLabel, varFiveLabel]
        for i in 0...textFieldList.count - 1 {
            let tempVar = PhysicsVariable.init(name:
             (labelList[i]?.text!)!)
            if (textFieldList[i]?.hasText)! {
                if !isItAValidNumber(input: (textFieldList[i]?.text!)!)
                 {
                    showAlert(alertType: "invalid input")
```

```
                listOfVars.removeAll()
                textFieldList[i]?.backgroundColor = UIColor.red
                return false
            } else if tempVar.isScalar &&
             (textFieldList[i]?.text?.contains("-"))! {
                var reachedE = false
                var negativeAtFront = false
                for i in (textFieldList[i]?.text)! {
                    if i == "-" && reachedE && !negativeAtFront {

                    } else if i == "-" {
                        negativeAtFront = true
                    } else if i == "e" {
                        reachedE = true
                    }
                }
                if negativeAtFront {
                    showAlert(alertType: "negative scalar \(i)")
                    listOfVars.removeAll()
                    textFieldList[i]?.backgroundColor = UIColor.red
                    return false
                }
            }
        }

    }
    return true
}

func isItAValidNumber(input: String) -> Bool {
    return Double(input) != nil
}

@IBAction func calculate(_ sender: UIButton) {
    if calculateBtn.backgroundImage(for: .normal) == UIImage(named:
     "button_another-calculation.gif") {
        reset()
    } else {
        if !areInputsValid() {
            return
        }
        saveQuestionBtn.isEnabled = true
        let list: [PhysicsVariable] = setListOfVars()
        let eq: Equation = Equation.init(listOfVars: list,
         equationName: toPass, isUnitsEnabled: true)
        eq.doEquation()
        listOfVars = list
        var tempString: String = ""
```

```
        for i in eq.getAnswer() {
            listOfVars.append(i)
            if ans(answer:i).contains("nan") || ans(answer:
             i).contains("inf") {
                showAlert(alertType: "nan")
                return
            }
            tempString += "\(i.getRealName()): \(ans(answer: i))
             \(i.unit)" + "\n"
        }

        answerLabel.text = tempString
        calculateBtn.setBackgroundImage(UIImage(named:
         "button_another-calculation.gif"), for: .normal)
        enableOrDisableFields(enable: false)
        for i in listOfVars {
            if i.isScalar && i.value < 0 {
                let negativeScalarAlert = UIAlertController(title:
                 "Error!", message: "It is not possible to have a
                 negative value for \(i.getRealName()), as it is a
                 scalar. Double check that you correctly inputed
                 values.", preferredStyle: .alert)
                let negativeScalarAlertAction =
                 UIAlertAction(title: "Got it!", style: .cancel,
                 handler: { (ACTION: UIAlertAction) in
                })

                 negativeScalarAlert
                 .addAction(negativeScalarAlertAction)
                present(negativeScalarAlert, animated: true)
            }
        }
    }
}
//depricated?
func ans(answer: PhysicsVariable) -> String {
    if UserDefaults.standard.getEnableSciNot() {
        let temp = Helper.CONVERT_TO_SCI_NOTATION(value:
         answer.getRoundedAns())
        return temp
    } else {
        return answer.getRoundedAns()
    }
}

func setListOfVars() -> [PhysicsVariable] {
    var list: [PhysicsVariable] = [PhysicsVariable]()
```

```
    var labelList = [varOneLabel, varTwoLabel, varThreeLabel,
     varFourLabel, varFiveLabel]
    var textFieldList = [varOneTextField, varTwoTextField,
     varThreeTextField, varFourTextField, varFiveTextField]
    for i in 0...4 {
        if (textFieldList[i]?.hasText)! {
            let x: Double = Double(textFieldList[i]!.text!)!
            let a: PhysicsVariable = PhysicsVariable.init(name:
             labelList[i]!.text!, value: x)
            a.unConvertedValue = x
            let unit: String = listOfButtons[i].titleLabel?.text ??
             "ERROR"
            a.unit = Helper.GET_FULL_UNIT_NAME(unitName: unit)
            list.append(a)
        }
    }
    return list
    //for loop to go through listOfLabels and listOfBtns (unit) DONE
    //for each i, check if field is empty DONE
    //if it is not empty, then set the unitbtn titleLabel.text
     equal to the unit of the PhysicsVariable that is associated
     with the label name...
    //also, still let user click btns even after fields are
     disabled... as the answers will be in the same units as what
     they are shown to be in the field!

}

func showAlert(alertType: String) {
    if alertType.contains("negative scalar") {
        let labelList: [UILabel] = [varOneLabel, varTwoLabel,
         varThreeLabel, varFourLabel, varFiveLabel]
        let i: Int = Int(String(alertType.last!))!
        let v: String = labelList[i].text!
        let errorAlert = UIAlertController(title: "Error!",
         message: "\(v) is a scalar, so it cannot be negative.
         Please input a valid value.", preferredStyle: .alert)
        let errorAlertAction = UIAlertAction(title: "Got it!",
         style: .cancel, handler: { (ACTION: UIAlertAction) in
        })
        errorAlert.addAction(errorAlertAction)
        present(errorAlert, animated: true)
    } else if alertType == "invalid input" {
        let errorAlert = UIAlertController(title: "Error!",
         message: "Input a valid value.", preferredStyle: .alert)
        let errorAlertAction = UIAlertAction(title: "Got it!",
         style: .cancel, handler: { (ACTION: UIAlertAction) in
        })
```

```
            errorAlert.addAction(errorAlertAction)
            present(errorAlert, animated: true)
        } else if alertType == "nan" {
            let errorAlert = UIAlertController(title: "Error!",
             message: "One or more of the number(s) inputed caused the
             answer to be invalid. Please double check your numbers and
             reinput them in.", preferredStyle: .alert)
            let errorAlertAction = UIAlertAction(title: "Got it!",
             style: .cancel, handler: { (ACTION: UIAlertAction) in
            })
            errorAlert.addAction(errorAlertAction)
            present(errorAlert, animated: true)
        }
    }

    func reset() {
        saveQuestionBtn.isEnabled = false
        calculateBtn.setBackgroundImage(UIImage(named:
         "button_calculate.gif"), for: .normal)
        listOfVars.removeAll()
        answerLabel.text?.removeAll()
        enableOrDisableFields(enable: true)
        setUpCalculator()
        updateVarCount()
    }

    @IBAction func saveTheQuestion(_ sender: Any) {
        //have an alert popup that prompts user to enter in a name for
         their saved
        //problem...

        let saveQuestionAlert = UIAlertController(title: "Save
         Problem", message: "Enter name for saved problem...",
         preferredStyle: .alert)

        saveQuestionAlert.addTextField { (field) in
        }

        let saveQuestionAction = UIAlertAction(title: "Got it!", style:
         .cancel, handler: { (ACTION: UIAlertAction) in
            self.savedName = saveQuestionAlert.textFields?[0].text ??
             "Saved Problem
             \(Int(UserDefaults.standard.getSavedProblemCounter()))"
            self.saveTheProblem()
        })
        saveQuestionAlert.addAction(saveQuestionAction)

        present(saveQuestionAlert, animated: true)
```

```
}

func saveTheProblem() {
    if savedName == "" {
        savedName = "Saved Problem
         \(Int(UserDefaults.standard.getSavedProblemCounter()))"
    }
    var listOfAnswers: [PhysicsVariable] = [PhysicsVariable]()
    var listOfKnowns: [PhysicsVariable] = [PhysicsVariable]()
    if equationName == "kinematics" {
        listOfAnswers.append(listOfVars[listOfVars.count - 2])
    }
    listOfAnswers.append(listOfVars[listOfVars.count - 1])
    for i in 0...listOfVars.count - listOfAnswers.count - 1 {
        listOfKnowns.append(listOfVars[i])
    }
    let savedProblem = SavedProblem.init(answers: listOfAnswers,
     knownValues: listOfKnowns, equation: equationName,
     savedProblemName: savedName, prompt: "na")

    let userDefaults = UserDefaults.standard

    var savedProblems = [SavedProblem]()
    if userDefaults.getSavedProblemCounter() > 1 {
        let decoded  = userDefaults.object(forKey: "savedProblems")
         as! Data
        savedProblems = NSKeyedUnarchiver.unarchiveObject(with:
         decoded) as! [SavedProblem]
    }
    savedProblems.append(savedProblem)

    let encodedData: Data =
     NSKeyedArchiver.archivedData(withRootObject: savedProblems)
    userDefaults.set(encodedData, forKey: "savedProblems")
    userDefaults.synchronize()

    UserDefaults.standard.setSavedProblemCounter(value:
     UserDefaults.standard.getSavedProblemCounter() + 1)
    savedName = "Saved Problem
     \(Int(UserDefaults.standard.getSavedProblemCounter()))"
    let saveAlert = UIAlertController(title: "Problem was saved.",
     message: "", preferredStyle: .alert)
    present(saveAlert, animated: true)
    let when = DispatchTime.now() + 1
    DispatchQueue.main.asyncAfter(deadline: when){
        // your code with delay
        saveAlert.dismiss(animated: true, completion: nil)
```

```
    }
}

func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 1
}

func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent
 component: Int) -> Int {
    switch pickerView.tag {
    case 1:
        return LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS.count
    case 2:
        return LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS.count
    case 3:
        return LIST_OF_UNITS_FOR_PICKER.ACCELERATION_UNITS.count
    case 4:
        return LIST_OF_UNITS_FOR_PICKER.DISTANCE_UNITS.count
    case 5:
        return LIST_OF_UNITS_FOR_PICKER.TIME_UNITS.count
    case 6:
        return LIST_OF_UNITS_FOR_PICKER.MASS_UNITS.count
    case 7:
        return LIST_OF_UNITS_FOR_PICKER.MASS_UNITS.count
    case 8:
        return LIST_OF_UNITS_FOR_PICKER.FORCE_UNITS.count
    case 9:
        return LIST_OF_UNITS_FOR_PICKER.ENERGY_UNITS.count
    default:
        print("error!")
        return LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS.count
    }
}

func pickerView(_ pickerView: UIPickerView, titleForRow row: Int,
 forComponent component: Int) -> String? {
    switch pickerView.tag {
    case 1:
        return LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS[row]
    case 2:
        return LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS[row]
    case 3:
        return LIST_OF_UNITS_FOR_PICKER.ACCELERATION_UNITS[row]
    case 4:
        return LIST_OF_UNITS_FOR_PICKER.DISTANCE_UNITS[row]
    case 5:
        return LIST_OF_UNITS_FOR_PICKER.TIME_UNITS[row]
    case 6:
```

```
            return LIST_OF_UNITS_FOR_PICKER.MASS_UNITS[row]
        case 7:
            return LIST_OF_UNITS_FOR_PICKER.MASS_UNITS[row]
        case 8:
            return LIST_OF_UNITS_FOR_PICKER.FORCE_UNITS[row]
        case 9:
            return LIST_OF_UNITS_FOR_PICKER.ENERGY_UNITS[row]
        default:
            print("error!")
            return LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS[row]
        }
    }

    func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int,
     inComponent component: Int) {
        var newRow = row
        if newRow == 0 {
            newRow += 1
        }
        switch pickerView.tag {
        case 1:
            tempPickerString =
             LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS[newRow]
        case 2:
            tempPickerString =
             LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS[newRow]
        case 3:
            tempPickerString =
             LIST_OF_UNITS_FOR_PICKER.ACCELERATION_UNITS[newRow]
        case 4:
            tempPickerString =
             LIST_OF_UNITS_FOR_PICKER.DISTANCE_UNITS[newRow]
        case 5:
            tempPickerString =
             LIST_OF_UNITS_FOR_PICKER.TIME_UNITS[newRow]
        case 6:
            tempPickerString =
             LIST_OF_UNITS_FOR_PICKER.MASS_UNITS[newRow]
        case 7:
            tempPickerString =
             LIST_OF_UNITS_FOR_PICKER.MASS_UNITS[newRow]
        case 8:
            tempPickerString =
             LIST_OF_UNITS_FOR_PICKER.FORCE_UNITS[newRow]
        case 9:
            tempPickerString =
             LIST_OF_UNITS_FOR_PICKER.ENERGY_UNITS[newRow]
        default:
```

```
            print("E R R O R")
        }
    }
    //font size, family, and color for each pickerview item
    func pickerView(_ pickerView: UIPickerView, viewForRow row: Int,
     forComponent component: Int, reusing view: UIView?) -> UIView {
        var pickerLabel: UILabel? = (view as? UILabel)
        var fontSize: CGFloat = 15
        switch true {
        case self.view.frame.height > 600 && self.view.frame.width <
         500:
            fontSize = 15
        case self.view.frame.width > 500:
            fontSize = 40
        default:
            fontSize = 13
        }
        if pickerLabel == nil {
            pickerLabel = UILabel()
            pickerLabel?.font = UIFont(name: "Menlo", size: fontSize)
            pickerLabel?.textAlignment = .center
        }
        switch pickerView.tag {
        case 1:
            pickerLabel?.text =
             LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS[row]
        case 2:
            pickerLabel?.text =
             LIST_OF_UNITS_FOR_PICKER.VELOCITY_UNITS[row]
        case 3:
            pickerLabel?.text =
             LIST_OF_UNITS_FOR_PICKER.ACCELERATION_UNITS[row]
        case 4:
            pickerLabel?.text =
             LIST_OF_UNITS_FOR_PICKER.DISTANCE_UNITS[row]
        case 5:
            pickerLabel?.text = LIST_OF_UNITS_FOR_PICKER.TIME_UNITS[row]
        case 6:
            pickerLabel?.text = LIST_OF_UNITS_FOR_PICKER.MASS_UNITS[row]
        case 7:
            pickerLabel?.text = LIST_OF_UNITS_FOR_PICKER.MASS_UNITS[row]
        case 8:
            pickerLabel?.text =
             LIST_OF_UNITS_FOR_PICKER.FORCE_UNITS[row]
        case 9:
            pickerLabel?.text =
             LIST_OF_UNITS_FOR_PICKER.ENERGY_UNITS[row]
        default:
```

```
            print("FATAL ERROR!")
        }
        pickerLabel?.textColor = UIColor.white
        return pickerLabel!
    }
    //spacing between pickerview items
    func pickerView(_ pickerView: UIPickerView, rowHeightForComponent
     component: Int) -> CGFloat {
        switch true {
        case self.view.frame.height > 600 && self.view.frame.width <
         500:
            return 22.0
        case self.view.frame.width > 500:
            return 48.0
        default:
            return 22.0
        }

    }

    //text field restrictions:
    func textField(_ textField: UITextField, shouldChangeCharactersIn
     range: NSRange, replacementString string: String) -> Bool {
        let inverseSet =
         NSCharacterSet(charactersIn:"0123456789").inverted
        let components = string.components(separatedBy: inverseSet)
        let filtered = components.joined(separator: "")

        if filtered == string {
            return true
        } else {
            if string == "." {
                let countdots =
                 textField.text!.components(separatedBy:".").count - 1
                if countdots == 0 {
                    return true
                } else {
                    if countdots > 0 && string == "." {
                        return false
                    } else {
                        return true
                    }
                }
            } else {
                if string == "-" {
                    let countNegs =
                     textField.text!.components(separatedBy:"-").count
                     - 1
```

```
                if countNegs <= 1 {
                    return true
                } else {
                    if countNegs > 1 && string == "-" {
                        return false
                    } else {
                        return true
                    }
                }
            } else {
                if string == "e" {
                    let countdots =
                     textField.text!.components(separatedBy:"e")
                     .count - 1
                    if countdots == 0 {
                        return true
                    } else {
                        if countdots > 0 && string == "e" {
                            return false
                        } else {
                            return true
                        }
                    }
            } else {
                return false
            }
        }
    }
}
}


//new stuff
func helpMode() {
    //could use something to tell users that they are in Help
     Mode...
    //like a label at the top or whereever it would fit that says
     help mode
    //or it can be a banner at top that can just be dismissed with
     an x... idk

    ///will display kinematics stuff for this
    setUpKinematics()
    disableEverything()
    addHelpModeBtns()
    setUpInvisibleBtns()
}
```

```
func setUpKinematics() {
    toPass = "kinematics"
    equationName = "kinematics"
    setUpLabelsAndFields(equationName)
    setUpUnitBtns()
    calculatorTitleLabel.text = "Kinematics Calculator"
}

func addHelpModeBtns() {

    var factor: CGFloat = 1
    if self.view.frame.width > 500 {
        factor = 2
    }

    let helpView = UIView(frame: CGRect(x: 0, y:
     self.view.frame.maxY - 50*factor, width:
     self.view.frame.width*factor, height: 50*factor))
    helpView.backgroundColor = UIColor.gray
    self.view.addSubview(helpView)
    let leftArrow: UIButton = UIButton(frame: CGRect(x: 50*factor,
     y: self.view.frame.maxY - 50*factor, width: 50*factor, height:
     50*factor))
    leftArrow.setBackgroundImage(UIImage.init(named:
     "left_arrow.png"), for: .normal)
    leftArrow.addTarget(self, action: #selector(prevView), for:
     .touchUpInside)
    self.view.addSubview(leftArrow)

    let rightArrow: UIButton = UIButton(frame: CGRect(x:
     self.view.frame.maxX - 100*factor, y: self.view.frame.maxY -
     50*factor, width: 50*factor, height: 50*factor))
    rightArrow.setBackgroundImage(UIImage.init(named:
     "right_arrow.png"), for: .normal)
    rightArrow.addTarget(self, action: #selector(nextView), for:
     .touchUpInside)
    self.view.addSubview(rightArrow)

    let exitBtn: UIButton = UIButton(frame: CGRect(x:
     self.view.frame.midX - factor*75/2, y: self.view.frame.maxY -
     40*factor, width: 75*factor, height: 25*factor))
    exitBtn.setBackgroundImage(UIImage(named:
     "button_exit-help.gif"), for: .normal)
    //later add a nice picture for this (or just copy the one from
     quiz)
    exitBtn.addTarget(self, action: #selector(exitHelp), for:
     .touchUpInside)
    self.view.addSubview(exitBtn)
```

```
}

func disableEverything() {
    for i in self.view.subviews {
        i.isUserInteractionEnabled = false
    }
    varView.isUserInteractionEnabled = true
    for i in self.varView.subviews {
        i.isUserInteractionEnabled = false
    }
}

func setUpInvisibleBtns() {
    var listOfBtns: [UIButton] = [UIButton]()


    listOfBtns.append(UIButton(frame: calculatorTitleLabel.frame))
    listOfBtns.append(UIButton(frame: moreBtn.frame))
    listOfBtns.append(UIButton(frame: settingsBtn.frame))

    listOfBtns.append(UIButton(frame: CGRect(x:
     varOneLabel.frame.minX, y: varOneLabel.frame.minY, width:
     varView.frame.width, height: varOneLabel.frame.height)))
    listOfBtns.append(UIButton(frame: CGRect(x:
     varTwoLabel.frame.minX, y: varTwoLabel.frame.minY, width:
     varView.frame.width, height: varTwoLabel.frame.height)))
    listOfBtns.append(UIButton(frame: CGRect(x:
     varThreeLabel.frame.minX, y: varThreeLabel.frame.minY, width:
     varView.frame.width, height: varThreeLabel.frame.height)))
    listOfBtns.append(UIButton(frame: CGRect(x:
     varFourLabel.frame.minX, y: varFourLabel.frame.minY, width:
     varView.frame.width, height: varFourLabel.frame.height)))
    listOfBtns.append(UIButton(frame: CGRect(x:
     varFiveLabel.frame.minX, y: varFiveLabel.frame.minY, width:
     varView.frame.width, height: varFiveLabel.frame.height)))

    listOfBtns.append(UIButton(frame: listOfButtons[0].frame))
    listOfBtns.append(UIButton(frame: listOfButtons[1].frame))
    listOfBtns.append(UIButton(frame: listOfButtons[2].frame))
    listOfBtns.append(UIButton(frame: listOfButtons[3].frame))
    listOfBtns.append(UIButton(frame: listOfButtons[4].frame))
    listOfBtns.append(UIButton(frame: calculateBtn.frame))
    listOfBtns.append(UIButton(frame: saveQuestionBtn.frame))


    for i in 0...listOfBtns.count-1 {
        listOfBtns[i].tag = i
        listOfBtns[i].backgroundColor = UIColor.clear
```

```
        listOfBtns[i].addTarget(self, action: #selector(openPopup),
         for: .touchUpInside)
        if i > 2 && i < 13 {
            self.varView.addSubview(listOfBtns[i])
        } else {
            self.view.addSubview(listOfBtns[i])
        }

    }
}

@objc func openPopup(_ sender: UIButton) {
    if popUpAlreadyExists() {
        closePopup(self)
        return
    }

    var factor:CGFloat = 1
    if self.view.frame.width > 500 {
        factor = 2.5
    }
    let popUp: UITextView = UITextView(frame: CGRect(x:
     self.view.frame.midX-120*factor, y: self.view.frame.midY -
     90*factor, width: 240*factor, height: 180*factor))
    popUp.text = HelpPopups.CALC[sender.tag]
    popUp.tag = -64
    popUp.isEditable = false
    popUp.backgroundColor = UIColor(displayP3Red: 93/255, green:
     188/255, blue: 210/255, alpha: 1)
    popUp.font = UIFont(name: "Menlo", size: Helper.GET_FONT_SIZE()
     + 1*factor)
    self.view.addSubview(popUp)
    let exitGesture = UITapGestureRecognizer(target: self, action:
     #selector(closePopup))

    self.view.addGestureRecognizer(exitGesture)
    /*tag:
     0: titleLabel (select calculator)
     1: settingsBtn
     2: kinematics
     3: force
     4: kinetic
     5: grav
     */
}

func popUpAlreadyExists() -> Bool {
    for i in self.view.subviews {
```

```
            if i.tag == -64 {
                return true
            }
        }
        return false
    }

    @objc func closePopup(_ sender: Any) {
        for i in self.view.subviews {
            if i.tag == -64 {
                if let viewWithTag = self.view.viewWithTag(i.tag) {
                    viewWithTag.removeFromSuperview()
                }
            }
        }

    }

    @objc func exitHelp(_ sender: UIButton) {
        exitHelpMode = true
        performSegue(withIdentifier: "settings", sender: self)
    }

    @objc func nextView(_ sender: UIButton) {
        performSegue(withIdentifier: "next", sender: self)
        //move to next view
    }
    @objc func prevView(_ sender: UIButton) {
        performSegue(withIdentifier: "home", sender: self)
    }

}
```