```swift
//
//  ENButton.swift
//  RGB Controller
//
//  Created by Erik Nordlund on 6/13/18.
//  Copyright © 2018 Erik Nordlund. All rights reserved.
//
//  Arm Controller includes the following open-source components:
//      • swiftBluetoothSerial: https://github.com/hoiberg/SwiftBluetoothSerial
//      • peertalk-simple: https://github.com/kirankunigiri/peertalk-simple

import UIKit

class ENButton: UIButton {


    var isVanishing = false
    var vanishDelay = 0.2
    var vanishDuration = 0.2

    let springAnimationDuration: Double = 0.4


    // corner radius
    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        self.layer.cornerRadius = 6
        //fatalError("init(coder:) has not been implemented")

    }


    // adding depth response
    override var isHighlighted: Bool {
        get {
            return super.isHighlighted
        } set {
            disableDiscoverability()
            if newValue {
                // when tapped



                let animationDuration: Double = 0.1

                UIView.animate(withDuration: animationDuration, animations: {
                    self.transform = CGAffineTransform(scaleX: 0.9, y: 0.9)
                })
            } else {
                UIView.animate(withDuration: springAnimationDuration,
                               delay: 0,
```

```
                            usingSpringWithDamping: 0.5,
                            initialSpringVelocity: 24.0,
                            options:
                             UIViewAnimationOptions.allowUserInteraction,
                            animations: { () -> Void in
                             self.transform = CGAffineTransform.identity
            })

            if isVanishing && isTouchInside {
                Timer.scheduledTimer(withTimeInterval:
                 springAnimationDuration + vanishDelay, repeats: false,
                 block: {_ in
                    UIView.animate(withDuration: self.vanishDuration,
                     animations: {
                        self.transform = CGAffineTransform(scaleX: 0.001,
                         y: 0.001)
                    }, completion: {_ in
                        UIView.transition(with: self, duration: 0.01,
                         options:
                         UIViewAnimationOptions.transitionCrossDissolve,
                         animations: {
                            self.isHidden = true
                        }, completion: {_ in
                            self.transform = CGAffineTransform(scaleX: 1,
                             y: 1)
                        })
                    })
                })
            }

        }
        super.isHighlighted = newValue
    }
}



// discoverability jiggle
private var discoverabilityTimer: Timer?

func enableDiscoverability(delay: Double, animationDuration: Double,
 repeatInterval: Double?) {
    let flickDuration = 0.07
    let duration = animationDuration - flickDuration

    var repeats = false
    var prePeriod = delay
    var interval = 0.0

    if (repeatInterval != nil) {
        repeats = true
```

```
    interval = repeatInterval!
    if delay > repeatInterval! {
        prePeriod = delay - repeatInterval!

    } else {
        prePeriod = repeatInterval! - delay
    }

}



/// calculating rotationAngle given button's dimensions
let height = self.bounds.height
let width = self.bounds.width

let rotationAngle: CGFloat =
 asin((height+8.115030356)/sqrt(pow(height, 2)+pow(width,
 2)))-atan(height/width)



Timer.scheduledTimer(withTimeInterval: prePeriod, repeats: false,
 block: {_ in

    self.isEnabled = true

    if self.discoverabilityTimer == nil {
        self.discoverabilityTimer =
         Timer.scheduledTimer(withTimeInterval: interval, repeats:
         repeats, block: {_ in
            UIView.animate(withDuration: flickDuration,
                           delay: 0,
                           usingSpringWithDamping: 0.4,
                           initialSpringVelocity: 20.0,
                           options:
                            UIViewAnimationOptions
                            .allowUserInteraction,
                           animations: { () -> Void in
                            self.transform =
                             CGAffineTransform(rotationAngle:
                             rotationAngle)//self.transform =
                             CGAffineTransform.identity
            }) {_ in
                UIView.animate(withDuration: duration,
                               delay: 0,
                               usingSpringWithDamping: 0.3,
                               initialSpringVelocity: 24.0,
                               options:
                                UIViewAnimationOptions
                                .allowUserInteraction,
```

```
                                        animations: { () -> Void in
                                         self.transform =
                                          CGAffineTransform.identity
                        })
                    }
                })
            }


        })
    }


    func disableDiscoverability() {
        self.discoverabilityTimer?.invalidate()
        self.discoverabilityTimer = nil
    }

    func apparate() {
        self.transform = CGAffineTransform(scaleX: 0.001, y: 0.001)
        self.isHidden = false

        UIView.animate(withDuration: springAnimationDuration,
                       delay: 0,
                       usingSpringWithDamping: 0.5,
                       initialSpringVelocity: 24.0,
                       options: UIViewAnimationOptions.allowUserInteraction,
                       animations: { () -> Void in
                        self.transform = CGAffineTransform(scaleX: 1, y: 1)
        })
    }

    func vanish() {
        UIView.animate(withDuration: self.vanishDuration, animations: {
            self.transform = CGAffineTransform(scaleX: 0.001, y: 0.001)
        }, completion: {_ in
            UIView.transition(with: self, duration: 0.01, options:
             UIViewAnimationOptions.transitionCrossDissolve, animations: {
                self.isHidden = true
            }, completion: {_ in
                self.transform = CGAffineTransform(scaleX: 1, y: 1)
            })
        })
    }

}
```