**import UIKit**

**import SwiftyStoreKit**

**@UIApplicationMain**

**class AppDelegate: UIResponder, UIApplicationDelegate {**

**var window: UIWindow?**

**func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {**

**//注册内购**

**SwiftyStoreKit.completeTransactions(atomically: true) { purchases in**

**for purchase in purchases {**

**switch purchase.transaction.transactionState {**

**case .purchased, .restored:**

**if purchase.needsFinishTransaction {**

**// Deliver content from server, then:**

**SwiftyStoreKit.finishTransaction(purchase.transaction)**

**}**

**case .failed, .purchasing, .deferred:**

**break // do nothing**

**default:**

**break;**

**}**

**}**

**}**

**//设置默认的音频**

**if UserDefaults.standard.string(forKey: "currentAudio") == nil {**

**UserDefaults.standard.set("Welcome\_to\_He", forKey: "currentAudio")**

**}**

**GADMobileAds.sharedInstance().start(completionHandler: nil)**

**//        UMConfigure.initWithAppkey("5f3a3e6db4b08b653e95e8d6", channel: "APP Store")**

**UMConfigure.initWithAppkey("5f3a3e6db4b08b653e95e8d6", channel: "dev")**

**MobClick.setAutoPageEnabled(true)**

**// Override point for customization after application launch.**

**return true**

**}**

**}**

**import UIKit**

**import CoreMotion**

**import AVFoundation**

**import Kingfisher**

**import SDWebImage**

**import JExtension**

**import Photos**

**import Toast\_Swift**

**import GPUImage**

**import Photos**

**class MainViewController: UIViewController,GPUImageMovieWriterDelegate, SVGAPlayerDelegate {**

**//陀螺仪管理器**

**let motionManager = CMMotionManager()**

**var eluer:Eluer?**

**@IBOutlet weak var bannerView: GADBannerView!**

**@IBOutlet weak var senBgView: UIView!**

**@IBOutlet weak var magBgView: UIView!**

**@IBOutlet var radarBgView: UIView!**

**@IBOutlet weak var recordBtn: UIButton!**

**lazy var senPlayer: SVGAPlayer = {**

**let player = SVGAPlayer(frame: CGRect(x: 0, y: 0, width: 67, height: 37))**

**player.delegate = self**

**player.loops = 0**

**player.clearsAfterStop = true**

**return player**

**}()**

**lazy var radarPlayer: SVGAPlayer = {**

**let player = SVGAPlayer(frame: CGRect(x: 25, y: 25, width: 120, height: 120))**

**player.delegate = self**

**player.loops = 0**

**player.clearsAfterStop = true**

**return player**

**}()**

**lazy var magPlayer: SVGAPlayer = {**

**let player = SVGAPlayer(frame: CGRect(x: 0, y: 0, width: 67, height: 37))**

**player.delegate = self**

**player.loops = 0**

**player.clearsAfterStop = true**

**return player**

**}()**

**//Ghost gif view**

**//    var ghostImageView:SDAnimatedImageView?**

**//Ghost指示点**

**var ghost:Ghost?**

**lazy var ghPointView: UIImageView = {**

**let imageView = UIImageView(frame: CGRect(x: 1000, y: 0, width: 20, height: 20))**

**imageView.image = UIImage(named: "icon\_home\_greenDot")**

**imageView.backgroundColor = .clear**

**imageView.layer.cornerRadius = 10**

**imageView.layer.masksToBounds = true**

**return imageView**

**}()**

**lazy var ghostLab: UILabel = {**

**let lab = UILabel(frame: CGRect(x: 1000, y: 0, width: 25, height: 11))**

**lab.text = "Ghost"**

**lab.font = UIFont.systemFont(ofSize: 8)**

**lab.textColor = UIColor(red: 0.34, green: 1, blue: 0.34, alpha: 1)**

**return lab**

**}()**

**//Ghost 初始化的位置**

**var ghLocation:gLocation?**

**//当前页面是否有Ghost**

**var isHaveGhost = false**

**//当前页面是否播放音频**

**var isPlayAudio = true**

**//是否展示Ghost信息**

**var isShowGhostInfo: Bool = true {**

**didSet{**

**if isShowGhostInfo {**

**}else{**

**}**

**}**

**}**

**@IBOutlet weak var leftImg: UIImageView!**

**@IBOutlet weak var rightImg: UIImageView!**

**@IBOutlet var ghostInfoView: GhostInfoView!**

**@IBOutlet weak var FrequeLab: UILabel!**

**var frequeNumber:Double = 0.0**

**@IBOutlet var ghostTranslateView: TranslateView!**

**@IBOutlet weak var transFrequeLab: UILabel!**

**@IBOutlet var videoTimeView: UIView!**

**@IBOutlet weak var recordTimeLab: UILabel!**

**///sound**

**var bgAudioName = ""{**

**didSet{**

**}**

**}**

**var voicePlayer:AudioManage?**

**var camera:GPUImageVideoCamera?**

**var cameraScreen:GPUImageView?**

**var movieWrite:GPUImageMovieWriter?**

**var gifImages:Array<UIImage> = []**

**var duration:TimeInterval = 0.0**

**var currenIndex = 0**

**var filter:GPUImageDissolveBlendFilter?**

**var ghostImage:UIImageView?**

**var second = 0**

**var timer:Timer?**

**override var preferredStatusBarStyle: UIStatusBarStyle {**

**return UIStatusBarStyle.lightContent**

**}**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**setNeedsStatusBarAppearanceUpdate()**

**let authostatus = AVCaptureDevice.authorizationStatus(for: .video)**

**if (authostatus == .denied) || authostatus == .restricted {**

**let alertC = UIAlertController.init(title: "Setting", message: "Please turn on camera permissions", preferredStyle: UIAlertController.Style.alert)**

**let action1 = UIAlertAction.init(title: "Cancle", style: UIAlertAction.Style.cancel, handler: { action in**

**self.navigationController?.popViewController(animated: true)**

**})**

**let action2 = UIAlertAction.init(title: "Setting", style: UIAlertAction.Style.default, handler: { action in**

**let url  = NSURL(string: UIApplication.openSettingsURLString)**

**UIApplication.shared.open(url! as URL)**

**})**

**alertC.addAction(action1)**

**alertC.addAction(action2)**

**self.present(alertC, animated: true, completion: nil)**

**}**

**bannerView.adUnitID = "ca-app-pub-9104226499574771/4399418667"**

**bannerView.rootViewController = self**

**bannerView.load(GADRequest())**

**DispatchQueue.main.async {**

**self.createCamera()**

**}**

**self.senBgView.addSubview(self.senPlayer)**

**self.radarBgView.addSubview(self.radarPlayer)**

**self.magBgView.addSubview(self.magPlayer)**

**startAnimation()**

**}**

**override func viewWillAppear(\_ animated: Bool) {**

**startMotion()**

**self.navigationController?.isNavigationBarHidden = true**

**Timer.scheduledTimer(withTimeInterval: 2, repeats: false) { \_ in**

**if UserDefaults.standard.string(forKey: "currentAudio") != "off" {**

**self.voicePlayer = AudioManage(name: UserDefaults.standard.string(forKey: "currentAudio")!)**

**self.voicePlayer?.player.prepareToPlay()**

**\_ = self.voicePlayer?.play()**

**}**

**}**

**let time = Int(arc4random()%4 + 4)//两秒到6秒产生一个Ghost**

**self.perform(#selector(creatGhostTime), with: nil, afterDelay: TimeInterval(time))**

**}**

**override func viewWillDisappear(\_ animated: Bool) {**

**self.navigationController?.isNavigationBarHidden = false**

**//停止陀螺仪**

**self.motionManager.stopDeviceMotionUpdates()**

**//停止播放音频**

**self.voicePlayer?.stop()**

**//停止摄像**

**//移除Ghost**

**self.removeGhost()**

**timer?.invalidate()**

**}**

**@objc func creatGhostTime()  {**

**self.creatGhost()**

**timer = Timer.scheduledTimer(withTimeInterval: 80, repeats: false, block: { \_ in**

**self.removeGhost()**

**let time = Int(arc4random()%4 + 4)//两秒到6秒产生一个Ghost**

**self.perform(#selector(self.creatGhostTime), with: nil, afterDelay: TimeInterval(time))**

**})**

**}**

**func creatGhost() {**

**//生成一个ghost**

**let ghost = GhostManager.manager.getOneGhost()**

**self.ghost = ghost**

**//ghost的位置**

**ghLocation = ghost.ghostLocation**

**self.radarBgView.addSubview(ghPointView)**

**self.radarBgView.addSubview(ghostLab)**

**self.isHaveGhost = true**

**isShowGhostInfo = true**

**gifImages = []**

**let path = Bundle.main.path(forResource: ghost.gifName, ofType: "gif")**

**let imageData = NSData(contentsOfFile: path!)**

**gifImages = self.getImagesFromGif(path: path!)**

**duration = GifTool.duration(forGifData: imageData! as Data)**

**currenIndex = 0**

**ghostImage?.image = gifImages[currenIndex]**

**}**

**func removeGhost() {**

**self.isHaveGhost = false**

**self.isShowGhostInfo = false**

**self.ghostImage?.frame = CGRect(x: 1000, y: 0, width:  0, height:  0)**

**self.ghost = nil**

**self.ghostInfoView.removeFromSuperview()**

**self.ghPointView.removeFromSuperview()**

**self.ghostLab.removeFromSuperview()**

**}**

**func startMotion() {**

**//判断设备支持情况**

**guard motionManager.isGyroAvailable else {**

**print("\n当前设备不支持陀螺仪\n")**

**return**

**}**

**guard motionManager.isAccelerometerAvailable else {**

**print("\n当前设备不支持磁场\n")**

**return**

**}**

**//获取陀螺仪数据**

**self.motionManager.startDeviceMotionUpdates(to: OperationQueue()) { (data, error) in**

**var quad = Quad()**

**quad.q0 = self.motionManager.deviceMotion?.attitude.quaternion.w ?? 0.0**

**quad.q1 = self.motionManager.deviceMotion?.attitude.quaternion.x ?? 0.0**

**quad.q2 = self.motionManager.deviceMotion?.attitude.quaternion.y ?? 0.0**

**quad.q3 = self.motionManager.deviceMotion?.attitude.quaternion.z ?? 0.0**

**self.eluer = quadtoeluer(input: quad)**

**// Yaw（偏航）：欧拉角向量的y轴psi**

**// Pitch（俯仰）：欧拉角向量的x轴phi**

**// Roll（翻滚）： 欧拉角向量的z轴theta**

**//            print("yaw --- \(eluer.psi)\n yaw --- \(eluer.phi)\n yaw --- \(eluer.theta)\n ")**

**if self.isHaveGhost {**

**DispatchQueue.main.sync {**

**//85是雷达图frame的width的一半**

**let p = self.eluer!.psi - Double(self.ghLocation!.gx)**

**self.ghPointView.center.x = 85 + CGFloat(sin( p \* Double.pi / 180) \* 40)**

**self.ghPointView.center.y = 85 - CGFloat(cos(p \*  Double.pi / 180) \* 40)**

**self.ghostLab.center.x = self.ghPointView.center.x**

**self.ghostLab.center.y = self.ghPointView.center.y + 15**

**let moveX = CGFloat(Double.pi \* (self.eluer!.psi - Double(self.ghLocation!.gx)))**

**let moveZ = CGFloat(Double.pi \* (self.eluer!.phi - 90))**

**self.leftImg.isHidden = true**

**self.rightImg.isHidden = true**

**if (GHScreenWidth/2 + moveX) < GHScreenWidth && (GHScreenWidth/2 + moveX) > 0 {**

**if self.isShowGhostInfo{**

**if self.view.viewWithTag(999) == nil{**

**let width = self.getTextWidth(string: self.ghost?.ghostName ?? "", font: 16, height: 20, lineSpace: 0) + 20**

**self.ghostInfoView.frame = CGRect(x: 30, y: GH\_Navi\_Height + 10, width: width > 120 ? width : 120, height: 230)**

**self.ghostInfoView.setWith(ghost: self.ghost ?? Ghost(json: ""))**

**self.view.addSubview(self.ghostInfoView)**

**}**

**}**

**}else{**

**self.ghostInfoView.removeFromSuperview()**

**if (GHScreenWidth/2 + moveX) < 0 {**

**self.leftImg.isHidden = false**

**}else{**

**self.rightImg.isHidden = false**

**}**

**}**

**}**

**}**

**}**

**self.motionManager.accelerometerUpdateInterval = 0.3**

**self.motionManager.startAccelerometerUpdates(to: OperationQueue()) { (data, error) in**

**let x = data?.acceleration.x**

**let y = data?.acceleration.y**

**let z = data?.acceleration.z**

**DispatchQueue.main.async {**

**let date = Date()**

**let timeInterval:TimeInterval = TimeInterval(date.timeIntervalSince1970)**

**let millisecond = CLongLong(round(timeInterval))**

**if (millisecond % 2) == 1{**

**self.FrequeLab.text = String(format: "frequency:%.4f", abs(x ?? 0.0) + abs(y ?? 0.0) + abs(z ?? 0.0))**

**self.transFrequeLab.text = String(format: "frequency:%.4f", abs(x ?? 0.0) + abs(y ?? 0.0) + abs(z ?? 0.0))**

**}**

**}**

**}**

**}**

**func getTextWidth(string:String,font:CGFloat,height:CGFloat,lineSpace:CGFloat) ->CGFloat{**

**let font = UIFont.systemFont(ofSize: font)**

**let size = CGSize(width: CGFloat(MAXFLOAT), height: height)**

**let paragraphStyle = NSMutableParagraphStyle()**

**paragraphStyle.lineSpacing = lineSpace**

**let attributes = [NSAttributedString.Key.font:font, NSAttributedString.Key.paragraphStyle:paragraphStyle.copy()]**

**let text = string as NSString**

**let rect = text.boundingRect(with: size, options:.usesLineFragmentOrigin, attributes: attributes, context:nil)**

**return rect.size.width**

**}**

**@IBAction func SoundBtnClick(\_ sender: UIButton) {**

**let vc = SoundTableViewController.loadStoryboard(name: "Main")**

**self.navigationController?.pushViewController(vc, animated: true)**

**}**

**var remainingCount: Int = 0 {**

**didSet{**

**DispatchQueue.main.async {**

**self.recordTimeLab.text = Date.transToHourMinSec(time: Float(self.remainingCount))**

**}**

**}**

**}**

**var codeTimer:DispatchSourceTimer?**

**@IBAction func VideoBtnClick(\_ sender: UIButton) {**

**PhotoTool.photoAlbumPermissions(authorizedBlock: {**

**if !sender.isSelected {**

**DispatchQueue.main.async {**

**sender.isSelected = true**

**// MARK: -添加录制计时器**

**self.videoTimeView.frame = CGRect(x: GHScreenWidth/2 - 50, y: GH\_Navi\_Height, width: 100, height: 30)**

**self.view.addSubview(self.videoTimeView)**

**self.movieWrite?.startRecording()**

**self.codeTimer = DispatchSource.makeTimerSource(queue:DispatchQueue.global())**

**self.codeTimer?.schedule(deadline: .now(), repeating: .seconds(1))**

**self.codeTimer?.setEventHandler(handler: {**

**DispatchQueue.main.async {**

**self.remainingCount += 1**

**}**

**})**

**// 启动时间源**

**self.codeTimer?.resume()**

**}**

**}else{**

**self.codeTimer?.cancel()**

**self.videoTimeView.removeFromSuperview()**

**sender.isSelected = false**

**self.camera?.audioEncodingTarget = nil**

**self.movieWrite?.finishRecording()**

**print(self.moviePath())**

**PHPhotoLibrary.shared().performChanges({**

**PHAssetChangeRequest.creationRequestForAssetFromVideo(atFileURL: (URL(fileURLWithPath: self.moviePath())))**

**}) { (success, error) in**

**if let errorDescription = error?.localizedDescription {**

**print("写入视频错误：\(errorDescription)")**

**} else {**

**DispatchQueue.main.async {**

**self.view.makeToast("save video success", duration: 2, position: .center, style: ToastStyle())**

**}**

**print("写入视频成功")**

**}**

**self.resetWrite()**

**}**

**}**

**}) {**

**let alertC = UIAlertController.init(title: "Setting", message: "Please open album permissions", preferredStyle: UIAlertController.Style.alert)**

**let action1 = UIAlertAction.init(title: "Cancel", style: UIAlertAction.Style.cancel, handler: { action in**

**self.navigationController?.popViewController(animated: true)**

**})**

**let action2 = UIAlertAction.init(title: "Setting", style: UIAlertAction.Style.default, handler: { action in**

**let url  = NSURL(string: UIApplication.openSettingsURLString)**

**UIApplication.shared.open(url! as URL)**

**})**

**alertC.addAction(action1)**

**alertC.addAction(action2)**

**self.present(alertC, animated: true, completion: nil)**

**}**

**}**

**@IBAction func PhotoBtnClick(\_ sender: UIButton) {**

**PhotoTool.photoAlbumPermissions(authorizedBlock: {**

**self.movieWrite?.startRecording()**

**DispatchQueue.main.asyncAfter(deadline: .now() + 0.5) {**

**self.savePhoto()**

**}**

**}) {**

**let alertC = UIAlertController.init(title: "Setting", message: "Please open album permissions", preferredStyle: UIAlertController.Style.alert)**

**let action1 = UIAlertAction.init(title: "Cancel", style: UIAlertAction.Style.cancel, handler: { action in**

**self.navigationController?.popViewController(animated: true)**

**})**

**let action2 = UIAlertAction.init(title: "Setting", style: UIAlertAction.Style.default, handler: { action in**

**let url  = NSURL(string: UIApplication.openSettingsURLString)**

**UIApplication.shared.open(url! as URL)**

**})**

**alertC.addAction(action1)**

**alertC.addAction(action2)**

**self.present(alertC, animated: true, completion: nil)**

**//无权限**

**}**

**}**

**func savePhoto() {**

**self.camera?.audioEncodingTarget = nil**

**self.movieWrite?.finishRecording()**

**let url = NSURL(fileURLWithPath: self.moviePath())**

**let asset = AVURLAsset(url: url as URL)**

**let generate = AVAssetImageGenerator(asset: asset)**

**generate.appliesPreferredTrackTransform = true**

**let time = CMTimeMake(value: 1, timescale: 2)**

**guard let oneRef = try? generate.copyCGImage(at: time, actualTime: nil)else {**

**return**

**}**

**let oneImage = UIImage(cgImage: oneRef)**

**PHPhotoLibrary.shared().performChanges({**

**PHAssetChangeRequest.creationRequestForAsset(from: oneImage)**

**}) { (success, error) in**

**if success{**

**DispatchQueue.main.async {**

**self.view.makeToast("save success", duration: 2, position: .center, style: ToastStyle())**

**}**

**print("save success")**

**}else{**

**print("save fail")**

**}**

**self.resetWrite()**

**DispatchQueue.main.async {**

**//                sender.isEnabled = true**

**}**

**}**

**}**

**@IBAction func SettingBtnClick(\_ sender: UIButton) {**

**let set = SettingTableViewController.loadStoryboard(name: "Main")**

**self.navigationController?.pushViewController(set, animated: true)**

**}**

**@IBAction func infoTranslateClick(\_ sender: UIButton) {**

**if Purchase.shared.getTraVipStatus(){**

**self.isShowGhostInfo = false**

**self.ghostInfoView.removeFromSuperview()**

**self.ghostTranslateView.setView(ghost: self.ghost ?? Ghost(json: ""))**

**self.ghostTranslateView.frame = CGRect(x: 30, y: GH\_Navi\_Height, width: GHScreenWidth - 60, height: self.ghostTranslateView.getHeight(text: ghost?.gTranslate ?? ""))**

**self.view.addSubview(self.ghostTranslateView)**

**}else{**

**let ActionSheet: UIAlertController = UIAlertController(title: "Communicator", message: "Understand what the ghost is saying! Use a universal translator to upgrade ghost detectors and translate ghost voices into text.", preferredStyle: UIAlertController.Style.alert)**

**ActionSheet.addAction(UIAlertAction(title:"Purchase 0.99$", style: UIAlertAction.Style.default, handler: { (UIAlertAction) in**

**Purchase.shared.pay(productID: "com.gm.ghost.translate", productName: .spiritism)**

**}))**

**ActionSheet.addAction(UIAlertAction(title:"Restore(if you buy)", style: UIAlertAction.Style.default, handler: { (UIAlertAction) in**

**Purchase.shared.restore(productName: .translate) { bool in**

**}**

**}))**

**ActionSheet.addAction(UIAlertAction(title:"Cancel", style: UIAlertAction.Style.cancel, handler:nil))**

**self.present(ActionSheet, animated: true, completion: nil)**

**}**

**}**

**@IBAction func translateOKClick(\_ sender: UIButton) {**

**isShowGhostInfo = true**

**self.ghostTranslateView.removeFromSuperview()**

**}**

**@IBAction func askQuestionTap(\_ sender: UITapGestureRecognizer) {**

**let sp = SpiritshViewController.loadStoryboard(name: "Main")**

**self.navigationController?.pushViewController(sp, animated: true)**

**}**

**}**

**// MARK: -svgaPlayer**

**extension MainViewController{**

**func startAnimation() {**

**self.senPlayer.stopAnimation()**

**self.radarPlayer.stopAnimation()**

**self.magPlayer.stopAnimation()**

**let senParse = SVGAParser()**

**senParse.parse(withNamed: "sen", in: Bundle.main, completionBlock: { item in**

**self.senPlayer.videoItem = item**

**self.senPlayer.startAnimation()**

**}) { error in**

**}**

**let radarParse = SVGAParser()**

**radarParse.parse(withNamed: "radar", in: Bundle.main, completionBlock: { item in**

**self.radarPlayer.videoItem = item**

**self.radarPlayer.startAnimation()**

**}) { error in**

**}**

**let magParse = SVGAParser()**

**magParse.parse(withNamed: "mag", in: Bundle.main, completionBlock: { item in**

**self.magPlayer.videoItem = item**

**self.magPlayer.startAnimation()**

**}) { error in**

**}**

**}**

**}**

**extension MainViewController{**

**func createCamera() {**

**self.camera = GPUImageVideoCamera(sessionPreset: AVCaptureSession.Preset.vga640x480.rawValue, cameraPosition: .back)**

**self.camera?.outputImageOrientation = .portrait**

**self.camera?.horizontallyMirrorRearFacingCamera = false**

**self.camera?.horizontallyMirrorFrontFacingCamera = true**

**self.camera?.addAudioInputsAndOutputs()**

**self.cameraScreen = GPUImageView(frame: CGRect(x: 0, y: 0, width: GHScreenWidth, height: GHScreenHeight))**

**self.view.insertSubview(self.cameraScreen!, at: 0)**

**self.cameraScreen?.fillMode = kGPUImageFillModePreserveAspectRatioAndFill**

**self.cameraScreen?.clipsToBounds = true**

**//        self.cameraScreen?.layer.masksToBounds = true**

**let contentView = UIView(frame: CGRect(x: 0, y: 0, width: GHScreenWidth, height: GHScreenHeight))**

**//        creatGhost()**

**let width = ghost?.gSize.width ?? 0**

**let height = ghost?.gSize.height ?? 0**

**let rect = CGRect(x: 1000, y: 0, width: width, height: height)**

**ghostImage = UIImageView(frame: rect)**

**contentView.addSubview(ghostImage!)**

**//UIView 转成纹理对象**

**let uiElement = GPUImageUIElement(view: contentView)**

**//图像混合滤镜**

**let filter = GPUImageDissolveBlendFilter()**

**filter.mix = 0.5**

**let filterGroup = GPUImageFilterGroup()**

**//反色滤镜**

**let colorInvertFilter = GPUImageColorInvertFilter()**

**//黑白**

**let photoEffectNoir = GPUImageGrayscaleFilter()**

**filterGroup.addFilter(colorInvertFilter)**

**filterGroup.addFilter(photoEffectNoir)**

**colorInvertFilter.addTarget(photoEffectNoir)**

**filterGroup.initialFilters = [colorInvertFilter]**

**filterGroup.terminalFilter = photoEffectNoir**

**camera?.addTarget(filterGroup)**

**filterGroup.addTarget(filter)**

**uiElement?.addTarget(filter)**

**filter.addTarget(cameraScreen)**

**self.filter = filter**

**self.camera?.startCapture()**

**filterGroup.frameProcessingCompletionBlock = { (output,time) in**

**if self.isHaveGhost {**

**DispatchQueue.main.async {**

**self.currenIndex += 1**

**self.ghostImage?.image = self.gifImages[self.currenIndex]**

**let moveX = CGFloat(Double.pi \* (self.eluer!.psi - Double(self.ghLocation!.gx)))**

**let moveZ = CGFloat(Double.pi \* (self.eluer!.phi - 90))**

**let cgPoint = CGPoint(x: GHScreenWidth/2 + moveX, y: GHScreenHeight/2 + moveZ)**

**self.ghostImage?.frame = CGRect(x: 1000, y: 0, width: self.ghost?.gSize.width ?? 0, height: self.ghost?.gSize.height ?? 0)**

**self.ghostImage?.center = cgPoint**

**if self.currenIndex == self.gifImages.count - 1 {**

**self.currenIndex = 0**

**}**

**}**

**}**

**uiElement?.update()**

**}**

**addMoveWrite()**

**}**

**func addMoveWrite(){**

**unlink( (moviePath() as NSString).utf8String)**

**let willSaveUrl = URL(fileURLWithPath: moviePath())**

**self.movieWrite = GPUImageMovieWriter(movieURL: willSaveUrl, size: CGSize(width: 480, height: 640))**

**self.movieWrite?.encodingLiveVideo = true**

**self.movieWrite?.shouldPassthroughAudio = true**

**self.movieWrite?.hasAudioTrack = true**

**self.filter?.addTarget(self.movieWrite)**

**self.camera?.audioEncodingTarget = self.movieWrite**

**}**

**func moviePath() -> String {**

**let tempDir = NSSearchPathForDirectoriesInDomains(FileManager.SearchPathDirectory.cachesDirectory, FileManager.SearchPathDomainMask.userDomainMask, true).first as NSString?**

**return tempDir?.appendingPathComponent("Movie.m4v") ?? ""**

**}**

**func getImagesFromGif(path:String) -> Array<UIImage> {**

**var imageArr:Array<UIImage> = []**

**let gifImageSourceRef = CGImageSourceCreateWithURL(URL(fileURLWithPath: path) as CFURL, nil)**

**let framesCount = CGImageSourceGetCount(gifImageSourceRef!)**

**for i in 0 ..< framesCount {**

**let imageRef = CGImageSourceCreateImageAtIndex(gifImageSourceRef!, i, nil)**

**let image = UIImage(cgImage: imageRef!)**

**imageArr.append(image)**

**imageArr.append(image)**

**}**

**return imageArr**

**}**

**//移除原有的input，重新添加一个新的**

**func resetWrite() {**

**self.camera?.removeTarget(self.movieWrite)**

**checkForAndDeleteFile()**

**addMoveWrite()**

**}**

**func checkForAndDeleteFile() {**

**let fm = FileManager.default**

**let url = moviePath()**

**let exist = fm.fileExists(atPath: url)**

**if exist {**

**print("删除之前的临时文件")**

**do {**

**try fm.removeItem(at: URL(fileURLWithPath: self.moviePath()))**

**} catch let error as NSError {**

**print(error.localizedDescription)**

**}**

**}**

**}**

**}**

**import UIKit**

**import AVFoundation**

**class AudioManage: NSObject {**

**var player:AVAudioPlayer!**

**init(name:String,success:((Bool)->())? = nil) {**

**let path = Bundle.main.path(forResource: name, ofType: "mp3")**

**let pathUrl = path?.addingPercentEncoding(withAllowedCharacters: .urlQueryAllowed)**

**player = try? AVAudioPlayer(contentsOf: URL(string: pathUrl!)!)**

**}**

**///停止**

**func stop(){**

**player?.stop()**

**}**

**func pause() {**

**player?.pause()**

**}**

**///播放**

**func play()->Bool{**

**if (player?.isPlaying)! {**

**player?.pause()**

**return false**

**}else{**

**player?.play()**

**return true**

**}**

**}**

**///进度条相关**

**func progress()->Double{**

**return (player?.currentTime)!/(player?.duration)!**

**}**

**func currentTime()->Double{**

**return (player?.currentTime)!**

**}**

**///是否在播放音乐**

**func isPlaying()->Bool{**

**return (player?.isPlaying ?? false)!**

**}**

**}**

**public struct Quad {**

**var q0:Double = 0.0**

**var q1:Double = 0.0**

**var q2:Double = 0.0**

**var q3:Double = 0.0**

**}**

**public struct Eluer {**

**var phi:Double = 0.0**

**var theta:Double = 0.0**

**var psi:Double = 0.0**

**}**

**import Foundation**

**func quadtoeluer(input:Quad) -> Eluer {**

**let radtodeg = 180.0 / Double.pi**

**var output = Eluer()**

**output.phi = radtodeg \* atan2(2\*(input.q0\*input.q1+input.q2\*input.q3),1-2\*(pow(input.q1,2)+pow(input.q2,2)))**

**output.theta=radtodeg\*asin(2\*(input.q0\*input.q2-input.q1\*input.q3));**

**output.psi=radtodeg\*atan2(2\*(input.q0\*input.q3+input.q1\*input.q2),1-2\*(pow(input.q2,2)+pow(input.q3,2)))**

**return output**

**}**

**import UIKit**

**class GhostInfoView: UIView ,SVGAPlayerDelegate{**

**@IBOutlet weak var nameLab: UILabel!**

**@IBOutlet weak var ageLab: UILabel!**

**@IBOutlet weak var svgaBgView: UIView!**

**lazy var magPlayer: SVGAPlayer = {**

**let player = SVGAPlayer(frame: CGRect(x: 0, y: 0, width: svgaBgView.frame.size.width, height: svgaBgView.frame.size.height))**

**player.delegate = self**

**player.loops = 0**

**player.clearsAfterStop = true**

**return player**

**}()**

**override func awakeFromNib() {**

**super.awakeFromNib()**

**self.svgaBgView.addSubview(magPlayer)**

**}**

**func setWith(ghost:Ghost) {**

**nameLab.text = ghost.ghostName**

**ageLab.text = "Age:\(ghost.ghostAge)"**

**self.magPlayer.stopAnimation()**

**let senParse = SVGAParser()**

**senParse.parse(withNamed: "sound", in: Bundle.main, completionBlock: { item in**

**self.magPlayer.videoItem = item**

**self.magPlayer.startAnimation()**

**}) { error in**

**}**

**}**

**//    override func removeFromSuperview() {**

**//        self.magPlayer.stopAnimation()**

**//    }**

**}**

**import UIKit**

**class TranslateView: UIView {**

**@IBOutlet weak var nameLab: UILabel!**

**@IBOutlet weak var ageLab: UILabel!**

**@IBOutlet weak var translateLab: UILabel!**

**override func awakeFromNib() {**

**self.layer.cornerRadius = 6**

**self.layer.masksToBounds = true**

**}**

**func setView(ghost:Ghost) {**

**nameLab.text = ghost.ghostName**

**ageLab.text = "Age:\(ghost.ghostAge)"**

**translateLab.text = ghost.gTranslate**

**}**

**func getHeight(text:String) -> CGFloat {**

**return getNormalStrH(str: text) + 210**

**}**

**func getNormalStrH(str: String) -> CGFloat {**

**return getNormalStrSize(str: str, font: 16, w:  GHScreenWidth - 60, h: CGFloat.greatestFiniteMagnitude).height**

**}**

**private func getNormalStrSize(str: String? = nil, attriStr: NSMutableAttributedString? = nil, font: CGFloat, w: CGFloat, h: CGFloat) -> CGSize {**

**if str != nil {**

**let strSize = (str! as NSString).boundingRect(with: CGSize(width: w, height: h), options: .usesLineFragmentOrigin, attributes: [NSAttributedString.Key.font: UIFont.systemFont(ofSize: font)], context: nil).size**

**return strSize**

**}**

**if attriStr != nil {**

**let strSize = attriStr!.boundingRect(with: CGSize(width: w, height: h), options: .usesLineFragmentOrigin, context: nil).size**

**return strSize**

**}**

**return CGSize.zero**

**}**

**}**

**import UIKit**

**class SoundTableViewController: UITableViewController ,StoryboardLoadable,UIGestureRecognizerDelegate{**

**var soundArr = ["Welcome\_to\_He","Creepy\_willow","ES\_Drone\_Dark\_Low"]**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**self.navigationController?.navigationBar.isTranslucent = false**

**self.navigationController?.navigationBar.setBackgroundImage(UIImage.Create(size: CGSize(width: 1, height: 1), color: .black), for: UIBarMetrics.default)**

**self.navigationController?.navigationBar.shadowImage = UIImage()**

**self.navigationController?.navigationBar.barStyle = .black**

**self.navigationController?.navigationBar.tintColor = .white**

**setNeedsStatusBarAppearanceUpdate()**

**self.title = "Sound"**

**// Uncomment the following line to preserve selection between presentations**

**// self.clearsSelectionOnViewWillAppear = false**

**// Uncomment the following line to display an Edit button in the navigation bar for this view controller.**

**// self.navigationItem.rightBarButtonItem = self.editButtonItem**

**}**

**override var preferredStatusBarStyle: UIStatusBarStyle {**

**return UIStatusBarStyle.lightContent**

**}**

**override func viewDidAppear(\_ animated: Bool) {**

**self.navigationController?.interactivePopGestureRecognizer?.delegate = self;**

**}**

**func gestureRecognizerShouldBegin(\_ gestureRecognizer: UIGestureRecognizer) -> Bool {**

**return false**

**}**

**// MARK: - Table view data source**

**override func numberOfSections(in tableView: UITableView) -> Int {**

**// #warning Incomplete implementation, return the number of sections**

**return 2**

**}**

**override func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {**

**// #warning Incomplete implementation, return the number of rows**

**if section == 0 {**

**return 1**

**}else{**

**return soundArr.count**

**}**

**}**

**override func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {**

**let cell = tableView.dequeueReusableCell(withIdentifier: "scell", for: indexPath)**

**if indexPath.section == 0 {**

**let lab = cell.viewWithTag(101) as? UILabel**

**lab?.text = "Turn off the sound"**

**}else{**

**let lab = cell.viewWithTag(101) as? UILabel**

**lab?.text = soundArr[indexPath.row]**

**}**

**return cell**

**}**

**override func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {**

**if indexPath.section == 0 {**

**UserDefaults.standard.set("off", forKey: "currentAudio")**

**}else{**

**UserDefaults.standard.set(soundArr[indexPath.row], forKey: "currentAudio")**

**}**

**self.navigationController?.popViewController(animated: true)**

**}**

**override func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {**

**return 10**

**}**

**override func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {**

**let view = UIView()**

**view.backgroundColor = UIColor(red: 250/255.0, green: 250/255.0, blue: 250/255.0, alpha: 1)**

**return view**

**}**

**}**

**protocol NibLoadable {**

**}**

**extension NibLoadable where Self : UIView {**

**static func loadFromNib(\_ nibname : String? = nil) -> Self {**

**let loadName = nibname == nil ? "\(self)" : nibname!**

**return Bundle.main.loadNibNamed(loadName, owner: nil, options: nil)?.first as! Self**

**}**

**}**

**@objc protocol StoryboardLoadable {}**

**extension StoryboardLoadable where Self: UIViewController {**

**/// 提供 加载方法**

**static func loadStoryboard() -> Self {**

**return UIStoryboard(name: "\(self)", bundle: nil).instantiateViewController(withIdentifier: "\(self)") as! Self**

**}**

**static func loadStoryboard(name:String) -> Self {**

**print(self)**

**return UIStoryboard(name: name, bundle: nil).instantiateViewController(withIdentifier: "\(self)") as! Self**

**}**

**}**

**extension UIImage{**

**class func Create(size:CGSize,color : UIColor,alpha:CGFloat? = 1) -> UIImage{**

**UIGraphicsBeginImageContext(size)**

**color.setFill()**

**let bounds = CGRect.init(x: 0, y: 0, width: size.width, height: size.height)**

**UIRectFill(bounds)**

**UIImage().draw(in: bounds, blendMode: CGBlendMode.destinationIn, alpha: alpha ?? 1)**

**let tintedImage = UIGraphicsGetImageFromCurrentImageContext()**

**UIGraphicsEndImageContext()**

**return tintedImage!**

**}**

**}**

**import UIKit**

**class SpiritshViewController: UIViewController,StoryboardLoadable,SVGAPlayerDelegate,UIGestureRecognizerDelegate {**

**@IBOutlet var lineChart: LineChartView!**

**@IBOutlet weak var chartBgView: UIImageView!**

**@IBOutlet weak var answerBtn: UIButton!**

**var recorder:AVAudioRecorder? //录音器**

**var recorderSeetingsDic:[String : Any]? //录音器设置参数数组**

**var volumeTimer:Timer! //定时器线程，循环监测录音的音量大小**

**var aacPath:String? //录音存储路径**

**var startPoint:CGPoint = CGPoint(x: 47, y: 128)**

**var endPoint:CGPoint = CGPoint(x: 47, y: 128)**

**lazy var answerPlayer: SVGAPlayer = {**

**let player = SVGAPlayer(frame: CGRect(x: 0, y: 29, width: 292, height: 150))**

**player.backgroundColor = .black**

**player.delegate = self**

**player.loops = 1**

**player.clearsAfterStop = true**

**return player**

**}()**

**lazy var answerBgView: UIView = {**

**let view = UIView()**

**view.tag = 888**

**view.frame = CGRect(x: 4, y: 4, width: 292, height: 200)**

**view.backgroundColor = UIColor(red: 0/255.0, green: 0/255.0, blue: 0/255.0, alpha: 1)**

**return view**

**}()**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**self.navigationController?.navigationBar.isTranslucent = false**

**self.navigationController?.navigationBar.setBackgroundImage(UIImage.Create(size: CGSize(width: 1, height: 1), color: UIColor(red: 21/255.0, green: 21/255.0, blue: 21/255.0, alpha: 1)), for: UIBarMetrics.default)**

**self.navigationController?.navigationBar.shadowImage = UIImage()**

**self.navigationController?.navigationBar.barStyle = .black**

**self.navigationController?.navigationBar.tintColor = .white**

**setNeedsStatusBarAppearanceUpdate()**

**self.title = "Spiritism"**

**lineChart.frame = CGRect(x: 4, y: 4, width: 292, height: 200)**

**self.chartBgView.addSubview(lineChart)**

**initAudioSession()**

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

**}**

**override func viewDidAppear(\_ animated: Bool) {**

**self.navigationController?.interactivePopGestureRecognizer?.delegate = self;**

**}**

**func gestureRecognizerShouldBegin(\_ gestureRecognizer: UIGestureRecognizer) -> Bool {**

**return false**

**}**

**func initAudioSession() {**

**let session:AVAudioSession = AVAudioSession.sharedInstance()**

**try! session.setCategory(AVAudioSession.Category.playAndRecord)**

**try! session.setActive(true)**

**//获取Document目录**

**let docDir = NSSearchPathForDirectoriesInDomains(.documentDirectory,**

**.userDomainMask, true)[0]**

**//组合录音文件路径**

**aacPath = docDir + "/play.aac"**

**//初始化字典并添加设置参数**

**recorderSeetingsDic =**

**[**

**AVFormatIDKey: NSNumber(value: kAudioFormatMPEG4AAC),**

**AVNumberOfChannelsKey: 2, //录音的声道数，立体声为双声道**

**AVEncoderAudioQualityKey : AVAudioQuality.max.rawValue,**

**AVEncoderBitRateKey : 320000,**

**AVSampleRateKey : 44100.0 //录音器每秒采集的录音样本数**

**]**

**}**

**@IBAction func AswerTap(\_ sender: UIButton) {**

**sender.isSelected = !sender.isSelected**

**if sender.isSelected {**

**self.lineChart.clearLayer()**

**startPoint = CGPoint(x: 47, y: 128)**

**endPoint = CGPoint(x: 47, y: 128)**

**if self.chartBgView.viewWithTag(888) != nil {**

**self.answerPlayer.stopAnimation()**

**self.answerPlayer.removeFromSuperview()**

**self.answerBgView.removeFromSuperview()**

**}**

**recorder = try! AVAudioRecorder(url: URL(string: aacPath!)!,**

**settings: recorderSeetingsDic!)**

**if recorder != nil {**

**recorder!.isMeteringEnabled = true**

**recorder!.prepareToRecord()**

**recorder!.record()**

**volumeTimer = Timer.scheduledTimer(timeInterval: 0.1, target:self,selector:#selector(changeValue),userInfo: nil, repeats: true)**

**}**

**}else{**

**//停止录音**

**recorder?.stop()**

**//录音器释放**

**recorder = nil**

**//暂停定时器**

**if (volumeTimer != nil) {**

**volumeTimer.invalidate()**

**volumeTimer = nil**

**}**

**}**

**}**

**@objc func changeValue() {**

**if endPoint.x > 200 + 47 {**

**AswerTap(self.answerBtn)**

**return**

**}**

**recorder!.updateMeters() // 刷新音量数据**

**let averageV:Float = recorder!.averagePower(forChannel: 0) //获取音量的平均值**

**let maxV:Float = recorder!.peakPower(forChannel: 0) //获取音量最大值**

**let lowPassResult:Double = pow(Double(10), Double(0.05\*maxV))**

**print(averageV)**

**let x = CGFloat(65.0 / 40.0)**

**let y:CGFloat = CGFloat(148.0 - (averageV + 40) \* 2)**

**endPoint = CGPoint(x: startPoint.x + x, y: y )**

**print(startPoint)**

**self.lineChart.drawLine(startPoint: startPoint, endPoint: endPoint) {**

**self.startPoint = self.endPoint**

**}**

**}**

**@IBAction func getAnswer(\_ sender: UIButton) {**

**if self.answerBtn.isSelected {**

**AswerTap(self.answerBtn)**

**}**

**if Purchase.shared.getSpiVipStatus(){**

**showAnswer()**

**self.lineChart.clearLayer()**

**startPoint = CGPoint(x: 47, y: 128)**

**endPoint = CGPoint(x: 47, y: 128)**

**}else{**

**let ActionSheet: UIAlertController = UIAlertController(title: "Communicator", message: "Ask a yes or no question and the spirits will aswer, waiting for the signal to be analyzed", preferredStyle: UIAlertController.Style.alert)**

**ActionSheet.addAction(UIAlertAction(title:"Purchase 1.99$", style: UIAlertAction.Style.default, handler: { (UIAlertAction) in**

**Purchase.shared.pay(productID: "com.gm.ghost.spiritism", productName: .spiritism)**

**}))**

**ActionSheet.addAction(UIAlertAction(title:"Restore(if you buy)", style: UIAlertAction.Style.default, handler: { (UIAlertAction) in**

**Purchase.shared.restore(productName: .spiritism) { bool in**

**}**

**}))**

**ActionSheet.addAction(UIAlertAction(title:"Cancel", style: UIAlertAction.Style.cancel, handler:nil))**

**self.present(ActionSheet, animated: true, completion: nil)**

**}**

**}**

**func showAnswer() {**

**let answer = arc4random()%2**

**let name = answer == 0 ? "YES" : "NO"**

**self.answerBgView.addSubview(self.answerPlayer)**

**self.chartBgView.addSubview(self.answerBgView)**

**self.answerPlayer.stopAnimation()**

**let senParse = SVGAParser()**

**senParse.parse(withNamed: name, in: Bundle.main, completionBlock: { item in**

**self.answerPlayer.videoItem = item**

**self.answerPlayer.startAnimation()**

**}) { error in**

**}**

**}**

**func svgaPlayerDidFinishedAnimation(\_ player: SVGAPlayer!) {**

**self.answerPlayer.step(toPercentage: 1, andPlay: false)**

**}**

**}**

**import UIKit**

**class SettingTableViewController: UITableViewController,StoryboardLoadable,UIGestureRecognizerDelegate{**

**@IBOutlet weak var versionLab: UILabel!**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**self.navigationController?.navigationBar.isTranslucent = false**

**self.navigationController?.navigationBar.setBackgroundImage(UIImage.Create(size: CGSize(width: 1, height: 1), color: UIColor(red: 21/255.0, green: 21/255.0, blue: 21/255.0, alpha: 1)), for: UIBarMetrics.default)**

**self.navigationController?.navigationBar.shadowImage = UIImage()**

**self.navigationController?.navigationBar.barStyle = .black**

**self.navigationController?.navigationBar.tintColor = .white**

**setNeedsStatusBarAppearanceUpdate()**

**self.title = "Setting"**

**versionLab.text = "\(Bundle.main.infoDictionary!["CFBundleShortVersionString"] as! String)"**

**// Uncomment the following line to preserve selection between presentations**

**// self.clearsSelectionOnViewWillAppear = false**

**// Uncomment the following line to display an Edit button in the navigation bar for this view controller.**

**// self.navigationItem.rightBarButtonItem = self.editButtonItem**

**}**

**// MARK: - Table view data source**

**override func viewDidAppear(\_ animated: Bool) {**

**self.navigationController?.interactivePopGestureRecognizer?.delegate = self;**

**}**

**func gestureRecognizerShouldBegin(\_ gestureRecognizer: UIGestureRecognizer) -> Bool {**

**return false**

**}**

**override func numberOfSections(in tableView: UITableView) -> Int {**

**// #warning Incomplete implementation, return the number of sections**

**return 2**

**}**

**override func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {**

**if section == 0 {**

**return 2**

**}else{**

**return 3**

**}**

**}**

**override func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {**

**return 10**

**}**

**override func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {**

**let view = UIView()**

**view.backgroundColor = UIColor(red: 0.98, green: 0.98, blue: 0.98, alpha: 1)**

**return view**

**}**

**override func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {**

**if indexPath.section == 0 {**

**if indexPath.row == 0 {**

**let image = UIImage(named: "shareApp.png")**

**let url = URL(string: "https://apps.apple.com/cn/app/id1527981450")!**

**///  applicationActivities 可以没有元素，系统会自动选择合适的平台**

**let activityController = UIActivityViewController(activityItems: ["Ghost Detection Machine", image!,url], applicationActivities: [])**

**activityController.completionWithItemsHandler = {**

**(type, flag, array, error) -> Swift.Void in**

**print(type ?? "")**

**}**

**present(activityController, animated: true) { }**

**}**

**if indexPath.row == 1 {**

**let appUrl = "https://itunes.apple.com/cn/app/id1527981450?action=write-review"**

**if UIApplication.shared.canOpenURL(URL(string: appUrl)!){**

**UIApplication.shared.open(URL(string: appUrl)!, options: [:], completionHandler: nil)**

**}**

**}**

**}else if indexPath.section == 1{**

**if indexPath.row == 0 {**

**let vc = PureWebViewController()**

**vc.urlStr = "https://www.gmgmapps.com/foundaghost/privacy.html"**

**vc.titleStr = "Privacy Policy";**

**vc.modalPresentationStyle = .fullScreen**

**self.present(vc, animated: true, completion: nil)**

**}else if indexPath.row == 1{**

**let vc = PureWebViewController()**

**vc.urlStr = "https://www.gmgmapps.com/foundaghost/terms.html"**

**vc.titleStr = "Terms of Use";**

**vc.modalPresentationStyle = .fullScreen**

**self.present(vc, animated: true, completion: nil)**

**}**

**}**

**}**

**}**

**import UIKit**

**import WebKit**

**import SnapKit**

**class PureWebViewController: UIViewController,WKNavigationDelegate {**

**var webView:WKWebView?**

**var urlStr:String = ""**

**var titleStr:String = ""**

**lazy var progressView:UIProgressView = {**

**let progressView = UIProgressView()**

**progressView.trackTintColor = UIColor.white**

**progressView.progressTintColor = UIColor(red: 66/255.0, green: 192/255.0, blue: 46/255.0, alpha: 1)**

**return progressView**

**}()**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**initToolbar()**

**initWebView()**

**loadWebView()**

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

**}**

**func initToolbar(){**

**let toolBar = UIView(frame: CGRect(x: 0, y: 0, width: GHScreenWidth, height: GH\_Navi\_Height))**

**toolBar.backgroundColor = UIColor(red: 21/255.0, green: 21/255.0, blue: 21/255.0, alpha: 1)**

**let lineView = UIView(frame: CGRect(x: 0, y: GH\_Navi\_Height - 1, width: GHScreenWidth, height: 1))**

**let goBackButton = UIButton(type: .custom)**

**goBackButton.setImage(UIImage(named: "navback"), for: .normal)**

**goBackButton.addTarget(self, action: #selector(backClicked), for: .touchUpInside)**

**let titleLab = UILabel()**

**if self.titleStr != "" {**

**titleLab.text = self.titleStr**

**}**

**titleLab.textColor = UIColor.white**

**titleLab.font = UIFont.boldSystemFont(ofSize: 17)**

**toolBar.addSubview(goBackButton)**

**toolBar.addSubview(titleLab)**

**toolBar.addSubview(lineView)**

**self.view.addSubview(toolBar)**

**goBackButton.snp.makeConstraints { make in**

**make.left.equalTo(toolBar.snp\_leftMargin).offset(6)**

**make.bottom.equalTo(toolBar.snp\_bottomMargin).offset(-4)**

**make.width.height.equalTo(30)**

**}**

**titleLab.snp.makeConstraints { make in**

**make.centerX.equalTo(toolBar.snp\_centerXWithinMargins)**

**make.bottom.equalTo(toolBar.snp\_bottomMargin).offset(-7)**

**make.height.equalTo(22)**

**}**

**}**

**func initWebView() {**

**self.progressView.frame = CGRect(x: 0, y: GH\_Navi\_Height, width: GHScreenWidth, height: 2)**

**self.view.addSubview(progressView)**

**self.view.backgroundColor = UIColor.white**

**self.webView = WKWebView(frame: CGRect(x: 16, y: GH\_Navi\_Height + 12, width: GHScreenWidth - 32, height: GHScreenHeight - GH\_Navi\_Height - 12))**

**self.webView?.backgroundColor = UIColor.clear**

**self.webView?.navigationDelegate = self**

**self.webView?.addObserver(self, forKeyPath: "estimatedProgress", options: .new, context: nil)**

**self.view.addSubview(self.webView!);**

**}**

**func loadWebView() {**

**if self.urlStr != "" {**

**let request = URLRequest(url: URL(string: self.urlStr)!)**

**self.webView?.load(request)**

**}**

**}**

**@objc func backClicked(){**

**self.dismiss(animated: true, completion: nil)**

**}**

**func webView(\_ webView: WKWebView, decidePolicyFor navigationAction: WKNavigationAction, decisionHandler: @escaping (WKNavigationActionPolicy) -> Void) {**

**progressView.setProgress(0.0, animated: false)**

**if navigationAction.targetFrame == nil {**

**webView .load(navigationAction.request)**

**}**

**decisionHandler(WKNavigationActionPolicy.allow)**

**}**

**func webView(\_ webView: WKWebView, didFinish navigation: WKNavigation!) {**

**progressView.setProgress(0.0, animated: false)**

**}**

**override func observeValue(forKeyPath keyPath: String?, of object: Any?, change: [NSKeyValueChangeKey : Any]?, context: UnsafeMutableRawPointer?) {**

**//        SVProgressHUD.dismiss()**

**if (keyPath == "estimatedProgress") {**

**progressView.isHidden = self.webView?.estimatedProgress == 1**

**progressView.setProgress(Float(self.webView!.estimatedProgress), animated: true)**

**}**

**}**

**}**

**import UIKit**

**class LineChartView: UIView {**

**var layerArr:Array<CAShapeLayer> = []**

**override func awakeFromNib() {**

**//        addLayer()**

**}**

**func drawLine(startPoint:CGPoint,endPoint:CGPoint,result:@escaping()->()) {**

**// 线的路径**

**let linePath = UIBezierPath.init()**

**//MARK: 动画**

**// 起点**

**linePath.move(to: startPoint)**

**// 其他点**

**linePath.addLine(to: endPoint)**

**let lineLayer = CAShapeLayer.init()**

**lineLayer.lineWidth = 1**

**lineLayer.strokeColor = UIColor(red: 255/255.0, green: 255/255.0, blue: 255/255.0, alpha: 0.7).cgColor**

**lineLayer.path = linePath.cgPath**

**lineLayer.fillColor = UIColor.clear.cgColor**

**//动画1**

**let animation = CABasicAnimation(keyPath: "strokeEnd")**

**animation.fromValue = 0**

**animation.toValue = 1**

**animation.duration = 0.05**

**lineLayer.add(animation, forKey: "")**

**self.layer.addSublayer(lineLayer)**

**self.layerArr.append(lineLayer)**

**Timer.scheduledTimer(withTimeInterval: 0.05, repeats: false) { \_ in**

**result()**

**}**

**}**

**func clearLayer() {**

**self.layerArr.forEach { layer in**

**layer.removeFromSuperlayer()**

**}**

**self.layerArr = []**

**}**

**}**

**import UIKit**

**import Photos**

**typealias OperationBlock = ()->()**

**class PhotoTool: NSObject {**

**class func cameraPermissions(authorizedBlock: OperationBlock?,deniedBlock: OperationBlock?) {**

**let authStatus = AVCaptureDevice.authorizationStatus(for: AVMediaType.video)**

**// .notDetermined .authorized .restricted .denied**

**if authStatus == .notDetermined {**

**// 第一次触发授权 alert**

**AVCaptureDevice.requestAccess(for: .video,completionHandler: { (granted: Bool) in**

**self.cameraPermissions(authorizedBlock: authorizedBlock,deniedBlock: deniedBlock)**

**})**

**} else if authStatus == .authorized {**

**if authorizedBlock != nil {**

**authorizedBlock!()**

**}**

**} else {**

**if deniedBlock != nil {**

**deniedBlock!()**

**}**

**}**

**}**

**// 相册权限**

**class func photoAlbumPermissions(authorizedBlock: OperationBlock?, deniedBlock: OperationBlock?) {**

**let authStatus = PHPhotoLibrary.authorizationStatus()**

**// .notDetermined  .authorized  .restricted  .denied**

**if authStatus == .notDetermined {**

**// 第一次触发授权 alert**

**PHPhotoLibrary.requestAuthorization { (status:PHAuthorizationStatus) -> Void in**

**self.photoAlbumPermissions(authorizedBlock: authorizedBlock, deniedBlock: deniedBlock)**

**}**

**} else if authStatus == .authorized  {**

**if authorizedBlock != nil {**

**authorizedBlock!()**

**}**

**} else {**

**if deniedBlock != nil {**

**deniedBlock!()**

**}**

**}**

**}}**

**import UIKit**

**import SVProgressHUD**

**import SwiftyStoreKit**

**enum ProductName:String {**

**case translate = "translate"**

**case spiritism = "spiritism"**

**}**

**class Purchase: NSObject {**

**static let shared: Purchase = {**

**let instance = Purchase()**

**return instance**

**}()**

**func pay(productID: String,productName:ProductName,payResult: ((Bool)->())? = nil) {**

**SVProgressHUD.showWithNativeType()**

**SwiftyStoreKit.purchaseProduct(productID) { result in**

**switch result {**

**case .success(let product):**

**if productName == .translate {**

**UserDefaults.standard.set(true, forKey: ProductName.translate.rawValue)**

**}else{**

**UserDefaults.standard.set(true, forKey: ProductName.spiritism.rawValue)**

**}**

**if product.needsFinishTransaction {**

**SwiftyStoreKit.finishTransaction(product.transaction)**

**}**

**print("Purchase Success: \(product.productId)")**

**case .error(let error):**

**SVProgressHUD.dismiss()**

**switch error.code {**

**case .unknown:**

**print("Unknown error. Please contact support")**

**case .clientInvalid:**

**print("Not allowed to make the payment")**

**case .paymentCancelled:**

**break**

**case .paymentInvalid:**

**print("The purchase identifier was invalid")**

**case .paymentNotAllowed:**

**print("The device is not allowed to make the payment")**

**case .storeProductNotAvailable:**

**print("The product is not available in the current storefront")**

**case .cloudServicePermissionDenied:**

**print("Access to cloud service information is not allowed")**

**case .cloudServiceNetworkConnectionFailed:**

**print("Could not connect to the network")**

**case .cloudServiceRevoked:**

**print("User has revoked permission to use this cloud service")**

**default:**

**print((error as NSError).localizedDescription)**

**}**

**SVProgressHUD.showError(withStatus: (error as NSError).localizedDescription)**

**}**

**}**

**}**

**func restore(productName:ProductName,success:((\_ resulet:Bool)->())?) {**

**SVProgressHUD.showStatus(string: "restoring...")**

**let signal = DispatchSemaphore(value: 0)**

**var isSuccess = false**

**DispatchQueue.global().async {**

**SwiftyStoreKit.restorePurchases(atomically: true) { results in**

**SVProgressHUD.dismiss()**

**if results.restoreFailedPurchases.count > 0 {**

**print("Restore Failed: \(results.restoreFailedPurchases)")**

**isSuccess = false**

**signal.signal()**

**}**

**else if results.restoredPurchases.count > 0 {**

**print("Restore Success: \(results.restoredPurchases)")**

**if results.restoredPurchases.last?.productId == "com.gm.ghost.translate" {**

**UserDefaults.standard.set(true, forKey: ProductName.translate.rawValue)**

**}**

**if results.restoredPurchases.last?.productId == "com.gm.ghost.spiritism" {**

**UserDefaults.standard.set(true, forKey: ProductName.spiritism.rawValue)**

**}**

**signal.signal()**

**}**

**else {**

**print("Nothing to Restore")**

**isSuccess = false**

**signal.signal()**

**}**

**}**

**signal.wait()**

**if success != nil{**

**success!(isSuccess)**

**}**

**}**

**}**

**func getTraVipStatus() -> Bool{**

**if UserDefaults.standard.object(forKey: ProductName.translate.rawValue) == nil {**

**return false**

**}else{**

**return UserDefaults.standard.bool(forKey: ProductName.translate.rawValue)**

**}**

**}**

**func getSpiVipStatus() -> Bool{**

**if UserDefaults.standard.object(forKey: ProductName.spiritism.rawValue) == nil {**

**return false**

**}else{**

**return UserDefaults.standard.bool(forKey: ProductName.spiritism.rawValue)**

**}**

**}**

**}**

**extension SVProgressHUD{**

**/// 显示文字**

**static func showInfo(string:String?) {**

**SVProgressHUD.showInfo(withStatus: string)**

**SVProgressHUD.setBackgroundLayerColor(UIColor.black)**

**}**

**///状态显示 加载动画**

**static func showStatus(string:String?){**

**SVProgressHUD.setDefaultStyle(.custom)**

**SVProgressHUD.setDefaultMaskType(.black)**

**SVProgressHUD.setDefaultAnimationType(.native)**

**SVProgressHUD.show(withStatus: string)**

**}**

**static func showWithNativeType(){**

**SVProgressHUD.setDefaultStyle(.custom)**

**SVProgressHUD.setDefaultMaskType(.black)**

**SVProgressHUD.setDefaultAnimationType(.native)**

**SVProgressHUD.show()**

**}**

**}**

# import "GifTool.h"

# import <UIKit/UIKit.h>

**@implementation GifTool**

**//获取gif图片的总时长和循环次数**

* **(NSTimeInterval)durationForGifData:(NSData \*)data{**

**//将GIF图片转换成对应的图片源**

**CGImageSourceRef gifSource = CGImageSourceCreateWithData((\_\_bridge CFDataRef)data, NULL);**

**//获取其中图片源个数，即由多少帧图片组成**

**size\_t frameCout = CGImageSourceGetCount(gifSource);**

**//定义数组存储拆分出来的图片**

**NSMutableArray\* frames = [[NSMutableArray alloc] init];**

**NSTimeInterval totalDuration = 0;**

**for (size\_t i=0; i<frameCout; i++) {**

**//从GIF图片中取出源图片**

**CGImageRef imageRef = CGImageSourceCreateImageAtIndex(gifSource, i, NULL);**

**//将图片源转换成UIimageView能使用的图片源**

**UIImage\* imageName = [UIImage imageWithCGImage:imageRef];**

**//将图片加入数组中**

**[frames addObject:imageName];**

**NSTimeInterval duration = [self gifImageDeleyTime:gifSource index:i];**

**totalDuration += duration;**

**CGImageRelease(imageRef);**

**}**

**//获取循环次数**

**NSInteger loopCount;//循环次数**

**CFDictionaryRef properties = CGImageSourceCopyProperties(gifSource, NULL);**

**if (properties) {**

**CFDictionaryRef gif = CFDictionaryGetValue(properties, kCGImagePropertyGIFDictionary);**

**if (gif) {**

**CFTypeRef loop = CFDictionaryGetValue(gif, kCGImagePropertyGIFLoopCount);**

**if (loop) {**

**//如果loop == NULL，表示不循环播放，当loopCount  == 0时，表示无限循环；**

**CFNumberGetValue(loop, kCFNumberNSIntegerType, &loopCount);**

**};**

**}**

**}**

**CFRelease(gifSource);**

**return totalDuration;**

**}**

**//获取GIF图片每帧的时长**

* **(NSTimeInterval)gifImageDeleyTime:(CGImageSourceRef)imageSource index:(NSInteger)index {**

**NSTimeInterval duration = 0;**

**CFDictionaryRef imageProperties = CGImageSourceCopyPropertiesAtIndex(imageSource, index, NULL);**

**if (imageProperties) {**

**CFDictionaryRef gifProperties;**

**BOOL result = CFDictionaryGetValueIfPresent(imageProperties, kCGImagePropertyGIFDictionary, (const void \*\*)&gifProperties);**

**if (result) {**

**const void \*durationValue;**

**if (CFDictionaryGetValueIfPresent(gifProperties, kCGImagePropertyGIFUnclampedDelayTime, &durationValue)) {**

**duration = [(\_\_bridge NSNumber \*)durationValue doubleValue];**

**if (duration < 0) {**

**if (CFDictionaryGetValueIfPresent(gifProperties, kCGImagePropertyGIFDelayTime, &durationValue)) {**

**duration = [(\_\_bridge NSNumber \*)durationValue doubleValue];**

**}**

**}**

**}**

**}**

**}**

**return duration;**

**}**

**@end**