#import <UIKit/UIKit.h>

#import <CoreData/CoreData.h>

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (readonly, strong) NSPersistentContainer \*persistentContainer;

@property (strong, nonatomic) UIWindow \*window;

- (void)saveContext;

@end

#import "AppDelegate.h"

#import "PTBaseNavigationController.h"

#import "PTCameraViewController.h"

#import <IQKeyboardManager/IQKeyboardManager.h>

@interface AppDelegate ()

@end

@implementation AppDelegate

- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions {

// Override point for customization after application launch.

[[GADMobileAds sharedInstance] startWithCompletionHandler:nil];

[self loadIQKeyboardManager];

PTCameraViewController \*vc = [[PTCameraViewController alloc]init];

PTBaseNavigationController \*nav = [[PTBaseNavigationController alloc]initWithRootViewController:vc];

self.window.rootViewController = nav;

[self.window makeKeyAndVisible];

return YES;

}

- (void)loadIQKeyboardManager{

// 使用智能键盘

IQKeyboardManager \*manager = [IQKeyboardManager sharedManager];

// 控制整个功能是否启用

manager.enable = YES;

// 控制是否显示键盘上的自动工具条,当需要支持内联编辑(Inline Editing), 这就需要隐藏键盘上的工具条(默认打开)

manager.enableAutoToolbar = NO;

}

#pragma mark - Core Data stack

@synthesize persistentContainer = \_persistentContainer;

- (NSPersistentContainer \*)persistentContainer {

// The persistent container for the application. This implementation creates and returns a container, having loaded the store for the application to it.

@synchronized (self) {

if (\_persistentContainer == nil) {

\_persistentContainer = [[NSPersistentContainer alloc] initWithName:@"PhotoTranslate"];

[\_persistentContainer loadPersistentStoresWithCompletionHandler:^(NSPersistentStoreDescription \*storeDescription, NSError \*error) {

if (error != nil) {

// Replace this implementation with code to handle the error appropriately.

// abort() causes the application to generate a crash log and terminate. You should not use this function in a shipping application, although it may be useful during development.

/\*

Typical reasons for an error here include:

\* The parent directory does not exist, cannot be created, or disallows writing.

\* The persistent store is not accessible, due to permissions or data protection when the device is locked.

\* The device is out of space.

\* The store could not be migrated to the current model version.

Check the error message to determine what the actual problem was.

\*/

NSLog(@"Unresolved error %@, %@", error, error.userInfo);

abort();

}

}];

}

}

return \_persistentContainer;

}

#pragma mark - Core Data Saving support

- (void)saveContext {

NSManagedObjectContext \*context = self.persistentContainer.viewContext;

NSError \*error = nil;

if ([context hasChanges] && ![context save:&error]) {

// Replace this implementation with code to handle the error appropriately.

// abort() causes the application to generate a crash log and terminate. You should not use this function in a shipping application, although it may be useful during development.

NSLog(@"Unresolved error %@, %@", error, error.userInfo);

abort();

}

}

@end

#import "PTBaseViewController.h"

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTWebViewController : PTBaseViewController

@property (nonatomic, strong) NSString \*url;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTWebViewController.h"

#import "PTWKWebView.h"

@interface PTWebViewController ()

@property (weak, nonatomic) IBOutlet PTWKWebView \*webView;

@property (nonatomic, strong) UIProgressView \*myProgressView;// 进度条

@end

@implementation PTWebViewController

- (void)viewDidLoad {

[super viewDidLoad];

// Do any additional setup after loading the view from its nib.

NSURLRequest \*request = [[NSURLRequest alloc] initWithURL:[NSURL URLWithString:self.url]];

[self.webView loadRequest:request];

// 进度条

[self.webView addObserver:self forKeyPath:@"estimatedProgress" options:NSKeyValueObservingOptionNew context:nil];

[self.view addSubview:self.myProgressView];

}

- (void)dealloc{

[self.webView removeObserver:self forKeyPath:@"estimatedProgress"];

}

// 计算wkWebView进度条

- (void)observeValueForKeyPath:(NSString \*)keyPath ofObject:(id)object change:(NSDictionary \*)change context:(void \*)context{

if (object == self.webView && [keyPath isEqualToString:@"estimatedProgress"]) {

CGFloat newprogress = [[change objectForKey:NSKeyValueChangeNewKey] doubleValue];

self.myProgressView.alpha = 1.0f;

[self.myProgressView setProgress:newprogress animated:YES];

if (newprogress >= 1.0f) {

[UIView animateWithDuration:0.3f

delay:0.3f

options:UIViewAnimationOptionCurveEaseOut

animations:^{

self.myProgressView.alpha = 0.0f;

}

completion:^(BOOL finished) {

[self.myProgressView setProgress:0 animated:NO];

}];

}

} else {

[super observeValueForKeyPath:keyPath ofObject:object change:change context:context];

}

}

/\*

#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation

- (void)prepareForSegue:(UIStoryboardSegue \*)segue sender:(id)sender {

// Get the new view controller using [segue destinationViewController].

// Pass the selected object to the new view controller.

}

\*/

#pragma mark - getter and setter

- (UIProgressView \*)myProgressView{

if (\_myProgressView == nil) {

\_myProgressView = [[UIProgressView alloc] initWithFrame:CGRectMake(0, 0, [UIScreen mainScreen].bounds.size.width, 0)];

\_myProgressView.tintColor = [UIColor blueColor];

\_myProgressView.trackTintColor = [UIColor whiteColor];

}

return \_myProgressView;

}

@end

#import <WebKit/WebKit.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTWKWebView : WKWebView

@end

NS\_ASSUME\_NONNULL\_END

#import "PTWKWebView.h"

@implementation PTWKWebView

/\*

// Only override drawRect: if you perform custom drawing.

// An empty implementation adversely affects performance during animation.

- (void)drawRect:(CGRect)rect {

// Drawing code

}

\*/

- (instancetype)initWithCoder:(NSCoder \*)coder{

CGRect frame = [[UIScreen mainScreen] bounds];

WKWebViewConfiguration \*myConfiguration = [WKWebViewConfiguration new];

self = [super initWithFrame:frame configuration:myConfiguration];

self.translatesAutoresizingMaskIntoConstraints = NO;

return self;

}

@end

#import "PTBaseViewController.h"

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTTextComparedViewController : PTBaseViewController

@property (nonatomic, strong) NSString \*fromText;

@property (nonatomic, strong) NSString \*toText;

@property (nonatomic, strong) NSDictionary \*fromLanguageDic;

@property (nonatomic, strong) NSDictionary \*toLanguageDic;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTTextComparedViewController.h"

#import <CommonCrypto/CommonDigest.h>

//typedef void(^PDDelayedBlockHandle)(BOOL cancel);

@interface PTTextComparedViewController ()<GADBannerViewDelegate>

@property (weak, nonatomic) IBOutlet UITextView \*textView;

@property (weak, nonatomic) IBOutlet UISegmentedControl \*segment;

@property(nonatomic, strong) GADBannerView \*bannerView; // 谷歌横幅广告

//@property (nonatomic, assign) PDDelayedBlockHandle delayedBlockHandle;

@property (nonatomic, assign) int postCount;// 请求次数，最多5次

@end

@implementation PTTextComparedViewController

- (void)viewDidLoad {

[super viewDidLoad];

// Do any additional setup after loading the view from its nib.

self.title = NSLocalizedString(@"Text Comparison", nil);

[self.segment setTitle:NSLocalizedString(@"Original", nil) forSegmentAtIndex:0];

[self.segment setTitle:NSLocalizedString(@"Translation", nil) forSegmentAtIndex:1];

UIView \*bottomView = [[UIView alloc] initWithFrame:CGRectMake(0, SCREEN\_HEIGHT - StatusbarAndNavigationbarHeight - (is\_iPhoneX?49+34:64), SCREEN\_WIDTH, is\_iPhoneX?49+34:64)];

bottomView.backgroundColor = [UIColor whiteColor];

[self.view addSubview:bottomView];

// 线

UIView \*lineView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, SCREEN\_WIDTH, 0.5)];

lineView.backgroundColor = UIColor(0x000000, 0.1);

[bottomView addSubview:lineView];

// 编辑

UIButton \*editButton = [[UIButton alloc] initWithFrame:CGRectMake(13, 8, 34, 34)];

[editButton setImage:[UIImage imageNamed:@"text\_edit"] forState:UIControlStateNormal];

[editButton addTarget:self action:@selector(editButtonAction:) forControlEvents:UIControlEventTouchUpInside];

[bottomView addSubview:editButton];

// 拷贝

UIButton \*copyButton = [[UIButton alloc] initWithFrame:CGRectMake(SCREEN\_WIDTH -34 -13, 8, 34, 34)];

[copyButton setImage:[UIImage imageNamed:@"text\_copy"] forState:UIControlStateNormal];

[copyButton addTarget:self action:@selector(copyButtonAction:) forControlEvents:UIControlEventTouchUpInside];

[bottomView addSubview:copyButton];

// UITextView设置行间距

NSMutableParagraphStyle \*paragraphStyle = [[NSMutableParagraphStyle alloc] init];

paragraphStyle.lineSpacing = 6;// 字体的行间距

NSDictionary \*attributes = @{NSFontAttributeName:[UIFont systemFontOfSize:16],

NSParagraphStyleAttributeName:paragraphStyle

};

self.textView.attributedText = [[NSAttributedString alloc] initWithString:self.textView.text attributes:attributes];

// 默认显示翻译后的

self.textView.text = self.fromText;

// In this case, we instantiate the banner with desired ad size.

self.bannerView = [[GADBannerView alloc]

initWithAdSize:kGADAdSizeBanner];

[self addBannerViewToView:self.bannerView];

self.bannerView.adUnitID = @"ca-app-pub-9104226499574771/3090627881";

self.bannerView.rootViewController = self;

[self.bannerView loadRequest:[GADRequest request]];

self.bannerView.delegate = self;

}

- (void)addBannerViewToView:(UIView \*)bannerView {

bannerView.translatesAutoresizingMaskIntoConstraints = NO;

[self.view addSubview:bannerView];

[self.view addConstraints:@[

[NSLayoutConstraint constraintWithItem:bannerView

attribute:NSLayoutAttributeTop

relatedBy:NSLayoutRelationEqual

toItem:self.segment

attribute:NSLayoutAttributeBottom

multiplier:1

constant:10],

[NSLayoutConstraint constraintWithItem:bannerView

attribute:NSLayoutAttributeCenterX

relatedBy:NSLayoutRelationEqual

toItem:self.view

attribute:NSLayoutAttributeCenterX

multiplier:1

constant:0]

]];

}

/\*

#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation

- (void)prepareForSegue:(UIStoryboardSegue \*)segue sender:(id)sender {

// Get the new view controller using [segue destinationViewController].

// Pass the selected object to the new view controller.

}

\*/

#pragma mark - Action

- (IBAction)segmentedAction:(UISegmentedControl \*)sender {

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:[NSString stringWithFormat:@"btn\_segment\_%ld", (long)sender.selectedSegmentIndex] Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

if (sender.selectedSegmentIndex == 0) {

self.textView.text = self.fromText;

}else if (sender.selectedSegmentIndex == 1){

self.textView.text = self.toText;

}else{

}

}

- (void)editButtonAction:(id)sender{

NSLog(@"编辑");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_editButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

// [[PTEventRecord shareManager] addEventWithType:@"input" Name:@"text" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

self.textView.userInteractionEnabled = YES;

[self.textView becomeFirstResponder];

self.navigationItem.rightBarButtonItem = [[UIBarButtonItem alloc]initWithTitle:NSLocalizedString(@"Done", nil) style:UIBarButtonItemStylePlain target:self action:@selector(rightBarButtonAction:)];

}

- (void)copyButtonAction:(id)sender{

NSLog(@"拷贝");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_copyButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

UIPasteboard \*pastboard = [UIPasteboard generalPasteboard];

[pastboard setString:self.textView.text];

UIAlertController \*alert = [UIAlertController alertControllerWithTitle:NSLocalizedString(@"Done", nil) message:nil preferredStyle:UIAlertControllerStyleAlert];

[alert addAction:[UIAlertAction actionWithTitle:@"OK" style:UIAlertActionStyleDefault handler:^(UIAlertAction \* \_Nonnull action) {

// NSLog(@"用户点击了OK");

}]];

[self presentViewController:alert animated:YES completion:nil];

}

- (void)rightBarButtonAction:(id)sender{

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_DoneButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

self.textView.userInteractionEnabled = NO;

[self.textView resignFirstResponder];

self.navigationItem.rightBarButtonItem = nil;

if (self.segment.selectedSegmentIndex == 0) { // 如果当前编辑在源语言，则翻译方向是 源语言 -> 目标语言

self.fromText = self.textView.text;

// [self loadDataWithText:self.fromText fromLanguageCode:[self.fromLanguageDic objectForKey:@"code"] toLanguageCode:[self.toLanguageDic objectForKey:@"code"]];

[self loadDataWithText:self.fromText fromLanguageCode:[self.toLanguageDic objectForKey:@"code"] toLanguageCode:[self.fromLanguageDic objectForKey:@"code"]];

}else if (self.segment.selectedSegmentIndex == 1) { // 如果当前编辑在目标语言，则翻译方向是 目标语言 -> 源语言

self.toText = self.textView.text;

// [self loadDataWithText:self.toText fromLanguageCode:[self.toLanguageDic objectForKey:@"code"] toLanguageCode:[self.fromLanguageDic objectForKey:@"code"]];

[self loadDataWithText:self.toText fromLanguageCode:[self.fromLanguageDic objectForKey:@"code"] toLanguageCode:[self.toLanguageDic objectForKey:@"code"]];

}else{

}

}

#pragma mark - LoadData

- (void)loadDataWithText:(NSString \*)text fromLanguageCode:(NSString \*)fromLanguageCode toLanguageCode:(NSString \*)toLanguageCode{

WeakSelf(self);

[MBProgressHUD showHUD];

NSString \*ts = [PTTools getNowTimeTimestamp];

NSString \*nonce = [PTTools randomStringWithLength:[PTTools getRandomNumber:10 to:18]];

AFHTTPSessionManager \*manager = [AFHTTPSessionManager manager];

manager.requestSerializer.timeoutInterval = 20;

// request

NSError \*requestError = nil;

NSMutableURLRequest \*request = [manager.requestSerializer requestWithMethod:@"POST" URLString:[NSString stringWithFormat:@"https://gmgmapps.com/text/translate/?t=%@&s=%@",fromLanguageCode, toLanguageCode] parameters:nil error:&requestError];

// body

NSString \*jsonStr = @{@"text": text}.mj\_JSONString;

NSData \*postData = [jsonStr dataUsingEncoding:NSUTF8StringEncoding];

[request setHTTPBody:postData];

// 设置header

NSString \*str = [PTTools sha1:[NSString stringWithFormat:@"%@%@%@%s" , [PTTools md5:jsonStr], ts, nonce, SECRET]];

[request setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];

[request setValue:ts forHTTPHeaderField:@"X-TimeStamp"];

[request setValue:nonce forHTTPHeaderField:@"X-Nonce"];

[request setValue:str forHTTPHeaderField:@"X-Signature"];

NSURLSessionDataTask \*dataTask = [manager.session dataTaskWithRequest:request completionHandler:^(NSData \* \_Nullable data, NSURLResponse \* \_Nullable response, NSError \* \_Nullable error) {

if (error) { // failed

NSLog("\n请求失败：%@", error);

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

}else{ // succese

NSError \* error = nil;

id responseObject = [NSJSONSerialization JSONObjectWithData:data options:NSJSONReadingMutableContainers error:&error];

if (error) {

NSLog("\n解析数据失败：%@", responseObject);

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Json error"];

});

}else {

NSLog("\n请求成功 数据：\n%@", responseObject);

if ([[responseObject objectForKey:@"status"] isEqualToString:@"waiting"]) {// 成功

self->\_postCount = 0;

// 请求成功根据id查询识别与翻译结果-------

[weakSelf getTextCheckWithID:[responseObject objectForKey:@"id"] ts:ts nonce:nonce];

}else{// 错误

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

}

}

}

}];

[dataTask resume];

}

/// 根据id查询识别与翻译结果

/// @param idStr id

/// @param ts 时间戳

/// @param nonce 随机字符串

- (void)getTextCheckWithID:(NSString \*)idStr ts:(NSString \*)ts nonce:(NSString \*)nonce{

if (\_postCount >= 15) {

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

return;

}

WeakSelf(self);

AFHTTPSessionManager \*manager = [AFHTTPSessionManager manager];

manager.requestSerializer.timeoutInterval = 20;

// request

NSError \*requestError = nil;

NSMutableURLRequest \*request = [manager.requestSerializer requestWithMethod:@"POST" URLString:@"https://gmgmapps.com/image/status/query/" parameters:nil error:&requestError];

// body

NSString \*jsonStr = @{@"id":idStr}.mj\_JSONString;

NSData \*postData = [jsonStr dataUsingEncoding:NSUTF8StringEncoding];

[request setHTTPBody:postData];

// 设置header

NSString \*signStr = [PTTools sha1:[NSString stringWithFormat:@"%@%@%@%s" , [PTTools md5:jsonStr], ts, nonce, SECRET]];

[request setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];

[request setValue:ts forHTTPHeaderField:@"X-TimeStamp"];

[request setValue:nonce forHTTPHeaderField:@"X-Nonce"];

[request setValue:signStr forHTTPHeaderField:@"X-Signature"];

NSURLSessionDataTask \*dataTask = [manager.session dataTaskWithRequest:request completionHandler:^(NSData \* \_Nullable data, NSURLResponse \* \_Nullable response, NSError \* \_Nullable error) {

if (error) { // failed

NSLog("\n请求失败：%@", error);

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

}else{ // succese

NSError \* error = nil;

id responseObject = [NSJSONSerialization JSONObjectWithData:data options:NSJSONReadingMutableContainers error:&error];

if (error) {

NSLog("\n解析数据失败：%@", responseObject);

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Json error"];

});

}else {

NSLog("\n请求成功 数据：\n%@", responseObject);

if ([[responseObject objectForKey:@"status"] isEqualToString:@"completed"]) {// 成功

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

});

// 解析字符串

NSArray \*resultArray = [responseObject objectForKey:@"result"];

NSDictionary \*firstDic = resultArray.firstObject;

NSString \*str = [firstDic objectForKey:@"translated\_text"];

dispatch\_async(dispatch\_get\_main\_queue(), ^{

if (self.segment.selectedSegmentIndex == 0) { // 如果当前编辑在源语言，则翻译方向是 源语言 -> 目标语言

self.toText = str;

}else if (self.segment.selectedSegmentIndex == 1) { // 如果当前编辑在目标语言，则翻译方向是 目标语言 -> 源语言

self.fromText = str;

}else{

}

});

}else if ([[responseObject objectForKey:@"status"] isEqualToString:@"waiting"] || [[responseObject objectForKey:@"status"] isEqualToString:@"processing"]){

// 最多请求5次

self->\_postCount += 1;

[weakSelf getTextCheckWithID:idStr ts:ts nonce:nonce];

}else{// 错误

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

}

}

}

}];

[dataTask resume];

}

#pragma mark - GADBannerViewDelegate

/// Tells the delegate an ad request loaded an ad.

///告诉委托人广告请求已加载广告。

- (void)adViewDidReceiveAd:(GADBannerView \*)adView {

NSLog(@"adViewDidReceiveAd");

}

/// Tells the delegate an ad request failed.

///告诉委托人广告请求失败。

- (void)adView:(GADBannerView \*)adView

didFailToReceiveAdWithError:(GADRequestError \*)error {

NSLog(@"adView:didFailToReceiveAdWithError: %@", [error localizedDescription]);

}

/// Tells the delegate that a full-screen view will be presented in response

/// to the user clicking on an ad.

///告诉代表将以全屏视图显示

///给点击广告的用户。

- (void)adViewWillPresentScreen:(GADBannerView \*)adView {

NSLog(@"adViewWillPresentScreen");

}

/// Tells the delegate that the full-screen view will be dismissed.

///告诉代表该全屏视图将被关闭。

- (void)adViewWillDismissScreen:(GADBannerView \*)adView {

NSLog(@"adViewWillDismissScreen");

}

/// Tells the delegate that the full-screen view has been dismissed.

///告诉代表该全屏视图已关闭。

- (void)adViewDidDismissScreen:(GADBannerView \*)adView {

NSLog(@"adViewDidDismissScreen");

}

/// Tells the delegate that a user click will open another app (such as

/// the App Store), backgrounding the current app.

///告诉代表，用户单击将打开另一个应用程序（例如

/// App Store），以当前应用为背景。

- (void)adViewWillLeaveApplication:(GADBannerView \*)adView {

NSLog(@"adViewWillLeaveApplication");

}

@end

#import "PTBaseViewController.h"

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTSelectLanguageViewController : PTBaseViewController

@property (nonatomic, copy) void (^selectLanguageBlock)(NSDictionary \*languageDic);

@property (nonatomic, assign) BOOL isShowAutoLanguage;//判断是否展示自动检测

@end

NS\_ASSUME\_NONNULL\_END

#import "PTSelectLanguageViewController.h"

@interface PTSelectLanguageViewController ()<UITableViewDelegate, UITableViewDataSource>

@property (weak, nonatomic) IBOutlet UITableView \*tableView;

@property (strong,nonatomic) NSMutableArray \*sectionTitles;//头标题数组

@property (strong,nonatomic) NSMutableArray \*contentArray;//内容数组

@end

@implementation PTSelectLanguageViewController

- (void)viewDidLoad {

[super viewDidLoad];

// Do any additional setup after loading the view from its nib.

self.title = NSLocalizedString(@"Select Language", nil);

UIBarButtonItem \*left = [[UIBarButtonItem alloc]initWithTitle:@"Cancel" style:UIBarButtonItemStylePlain target:self action:@selector(cancelButtonAction:)];

self.navigationItem.leftBarButtonItem = left;

// aws支持翻译语言数据

self.sectionTitles = @[@"A",

@"B",

@"C",

@"D",

@"E",

@"F",

@"G",

@"H",

@"I",

@"J",

@"K",

@"L",

@"M",

@"N",

@"O",

@"P",

@"R",

@"S",

@"T",

@"U",

@"V",

@"W",

@"X",

@"Y",

@"Z"].mutableCopy;

self.contentArray = @[

@[@{@"name":@"Afrikaans",@"code":@"af"},@{@"name":@"Albanian",@"code":@"sq"},@{@"name":@"Amharic",@"code":@"am"},@{@"name":@"Arabic",@"code":@"ar"},@{@"name":@"Armenian",@"code":@"hy"},@{@"name":@"Azerbaijani",@"code":@"az"}],

@[@{@"name":@"Basque",@"code":@"eu"},@{@"name":@"Belarusian",@"code":@"be"},@{@"name":@"Bengali",@"code":@"bn"},@{@"name":@"Bosnian",@"code":@"bs"},@{@"name":@"Bulgarian",@"code":@"bg"}],

@[@{@"name":@"Catalan",@"code":@"ca"},@{@"name":@"Cebuano",@"code":@"ceb"},@{@"name":@"Chichewa",@"code":@"ny"},@{@"name":@"Chinese (Simplified)",@"code":@"zh-CN"},@{@"name":@"Chinese (Traditional)",@"code":@"zh-TW"},@{@"name":@"Corsican",@"code":@"co"},@{@"name":@"Croatian",@"code":@"hr"},@{@"name":@"Czech",@"code":@"cs"}], @[@{@"name":@"Danish",@"code":@"da"},@{@"name":@"Dutch",@"code":@"nl"}], @[@{@"name":@"English",@"code":@"en"},@{@"name":@"Esperanto",@"code":@"eo"},@{@"name":@"Estonian",@"code":@"et"}], @[@{@"name":@"Filipino",@"code":@"tl"},@{@"name":@"Finnish",@"code":@"fi"},@{@"name":@"French",@"code":@"fr"},@{@"name":@"Frisian",@"code":@"fy"}], @[@{@"name":@"Galician",@"code":@"gl"},@{@"name":@"Georgian",@"code":@"ka"},@{@"name":@"German",@"code":@"de"},@{@"name":@"Greek",@"code":@"el"},@{@"name":@"Gujarati",@"code":@"gu"}],

@[@{@"name":@"Haitian Creole",@"code":@"ht"},@{@"name":@"Hausa",@"code":@"ha"},@{@"name":@"Hawaiian",@"code":@"haw"},@{@"name":@"Hebrew",@"code":@"iw"},@{@"name":@"Hindi",@"code":@"hi"},@{@"name":@"Hmong",@"code":@"hmn"},@{@"name":@"Hungarian",@"code":@"hu"}], @[@{@"name":@"Icelandic",@"code":@"is"},@{@"name":@"Igbo",@"code":@"ig"},@{@"name":@"Indonesian",@"code":@"id"},@{@"name":@"Irish",@"code":@"ga"},@{@"name":@"Italian",@"code":@"it"}], @[@{@"name":@"Japanese",@"code":@"ja"},@{@"name":@"Javanese",@"code":@"jw"}], @[@{@"name":@"Kannada",@"code":@"kn"},@{@"name":@"Kazakh",@"code":@"kk"},@{@"name":@"Khmer",@"code":@"km"},@{@"name":@"Kinyarwanda",@"code":@"rw"},@{@"name":@"Korean",@"code":@"ko"},@{@"name":@"Kurdish (Kurmanji)",@"code":@"ku"},@{@"name":@"Kyrgyz",@"code":@"ky"}], @[@{@"name":@"Lao",@"code":@"lo"},@{@"name":@"Latin",@"code":@"la"},@{@"name":@"Latvian",@"code":@"lv"},@{@"name":@"Lithuanian",@"code":@"lt"},@{@"name":@"Luxembourgish",@"code":@"lb"}], @[@{@"name":@"Macedonian",@"code":@"mk"},@{@"name":@"Malagasy",@"code":@"mg"},@{@"name":@"Malay",@"code":@"ms"},@{@"name":@"Malayalam",@"code":@"ml"},@{@"name":@"Maltese",@"code":@"mt"},@{@"name":@"Maori",@"code":@"mi"},@{@"name":@"Marathi",@"code":@"mr"},@{@"name":@"Mongolian",@"code":@"mn"},@{@"name":@"Myanmar (Burmese)",@"code":@"my"}],

@[@{@"name":@"Nepali",@"code":@"ne"},@{@"name":@"Norwegian",@"code":@"no"}],

@[@{@"name":@"Odia (Oriya)",@"code":@"or"}],

@[@{@"name":@"Pashto",@"code":@"ps"},@{@"name":@"Persian",@"code":@"fa"},@{@"name":@"Polish",@"code":@"pl"},@{@"name":@"Portuguese",@"code":@"pt"},@{@"name":@"Punjabi",@"code":@"pa"}],

@[@{@"name":@"Romanian",@"code":@"ro"},@{@"name":@"Russian",@"code":@"ru"}], @[@{@"name":@"Samoan",@"code":@"sm"},@{@"name":@"Scots Gaelic",@"code":@"gd"},@{@"name":@"Serbian",@"code":@"sr"},@{@"name":@"Sesotho",@"code":@"st"},@{@"name":@"Shona",@"code":@"sn"},@{@"name":@"Sindhi",@"code":@"sd"},@{@"name":@"Sinhala",@"code":@"si"},@{@"name":@"Slovak",@"code":@"sk"},@{@"name":@"Slovenian",@"code":@"sl"},@{@"name":@"Somali",@"code":@"so"},@{@"name":@"Spanish",@"code":@"es"},@{@"name":@"Sundanese",@"code":@"su"},@{@"name":@"Swahili",@"code":@"sw"},@{@"name":@"Swedish",@"code":@"sv"}], @[@{@"name":@"Tajik",@"code":@"tg"},@{@"name":@"Tamil",@"code":@"ta"},@{@"name":@"Tatar",@"code":@"tt"},@{@"name":@"Telugu",@"code":@"te"},@{@"name":@"Thai",@"code":@"th"},@{@"name":@"Turkish",@"code":@"tr"},@{@"name":@"Turkmen",@"code":@"tk"}], @[@{@"name":@"Ukrainian",@"code":@"uk"},@{@"name":@"Urdu",@"code":@"ur"},@{@"name":@"Uyghur",@"code":@"ug"},@{@"name":@"Uzbek",@"code":@"uz"}],

@[@{@"name":@"Vietnamese",@"code":@"vi"}],

@[@{@"name":@"Welsh",@"code":@"cy"}],

@[@{@"name":@"Xhosa",@"code":@"xh"}],

@[@{@"name":@"Yiddish",@"code":@"yi"},@{@"name":@"Yoruba",@"code":@"yo"}],

@[@{@"name":@"Zulu",@"code":@"zu"}],

].mutableCopy;

// 查看是否有最近使用的数组

NSUserDefaults \*userDefault =[NSUserDefaults standardUserDefaults];

NSArray \*array = [userDefault objectForKey:@"RecentLanguageArray"];

if (array.count > 0) {

// 添加进标题、内容数组

[self.sectionTitles insertObject:@"RecentlyUsed" atIndex:0];

[self.contentArray insertObject:array atIndex:0];

}else{

}

// 判断是否展示自动检测

if (self.isShowAutoLanguage) {

// 自动检测

[self.sectionTitles insertObject:@"Auto" atIndex:0];

[self.contentArray insertObject:@[@{@"name":@"Auto",@"code":@"auto"}] atIndex:0];

}

}

/\*

#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation

- (void)prepareForSegue:(UIStoryboardSegue \*)segue sender:(id)sender {

// Get the new view controller using [segue destinationViewController].

// Pass the selected object to the new view controller.

}

\*/

#pragma mark - Action

- (void)cancelButtonAction:(id)sender{

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_CancelButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

[self dismissViewControllerAnimated:YES completion:^{

}];

}

#pragma mark - UITableViewDelegate, UITableViewDataSource

- (NSInteger)numberOfSectionsInTableView:(UITableView \*)tableView{

return self.contentArray.count;

}

- (NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section{

return [self.contentArray[section] count];

}

- (UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath{

static NSString \*cellID = @"PTSelectLanguageViewControllerCellID";

UITableViewCell \*cell = [tableView dequeueReusableCellWithIdentifier:cellID];

if (cell == nil) {

cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:cellID];

}

NSDictionary \*dic = self.contentArray[indexPath.section][indexPath.row];

cell.textLabel.text = [dic objectForKey:@"name"];

cell.selectionStyle = UITableViewCellSelectionStyleNone;

return cell;

}

- (void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath{

NSDictionary \*dic = self.contentArray[indexPath.section][indexPath.row];

if (self.selectLanguageBlock) {

self.selectLanguageBlock(dic);

}

[self dismissViewControllerAnimated:YES completion:^{

}];

// 自动检测不用存最近使用

if ([[dic objectForKey:@"name"] isEqualToString:@"Auto"]) {

}else{

// 选择后将最近3次的选择记录存在本地

NSUserDefaults \*userDefault =[NSUserDefaults standardUserDefaults];

NSArray \*array = [userDefault objectForKey:@"RecentLanguageArray"];

NSMutableArray \*mutableArray = [NSMutableArray arrayWithArray:array];

// 去重判断添加

if (![mutableArray containsObject:dic]) {

if (mutableArray.count > 2) {// 最多保存3个

[mutableArray removeLastObject];

}else{

}

[mutableArray insertObject:dic atIndex:0];

}else{

// 如果有不添加

}

NSArray \*array2 = [NSArray arrayWithArray:mutableArray];

[userDefault setObject:array2 forKey:@"RecentLanguageArray"];

[userDefault synchronize];

}

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:[NSString stringWithFormat:@"btn\_selectCell\_%@", [dic objectForKey:@"name"]] Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

}

//点击索引跳转到相应位置

-(NSInteger)tableView:(UITableView \*)tableView sectionForSectionIndexTitle:(NSString \*)title atIndex:(NSInteger)index{

NSIndexPath \*selectIndexPath = [NSIndexPath indexPathForRow:0 inSection:index];

[tableView scrollToRowAtIndexPath:selectIndexPath atScrollPosition:UITableViewScrollPositionNone animated:YES];

return index;

}

- (CGFloat)tableView:(UITableView \*)tableView heightForHeaderInSection:(NSInteger)section{

return 28;

}

- (CGFloat)tableView:(UITableView \*)tableView heightForFooterInSection:(NSInteger)section{

if (section == 0) {

return 20;

}else{

return 0.1;

}

}

//分区标题

-(NSString \*)tableView:(UITableView \*)tableView titleForHeaderInSection:(NSInteger)section{

NSString \*str = [self.sectionTitles objectAtIndex:section];

// 自动检测不显示组标题

if ([str isEqualToString:@"Auto"]) {

return @"";

}else{

return str;

}

}

//索引标题

-(NSArray \*)sectionIndexTitlesForTableView:(UITableView \*)tableView{

// 右侧索引去掉自动检测 和 最近使用

NSMutableArray \*array = self.sectionTitles.mutableCopy;

if ([array containsObject:@"Auto"]) {

[array replaceObjectAtIndex:[array indexOfObject:@"Auto"] withObject:@""];

}

if ([array containsObject:@"RecentlyUsed"]) {

[array replaceObjectAtIndex:[array indexOfObject:@"RecentlyUsed"] withObject:@""];

}

return array;

}

// tableView的UITableViewHeaderFooterView修改颜色为透明

- (void)tableView:(UITableView \*)tableView willDisplayHeaderView:(UIView \*)view forSection:(NSInteger)section{

if ([view isKindOfClass:[UITableViewHeaderFooterView class]]) {

view.tintColor = UIColor(0xf7f7f7, 1.0);

((UITableViewHeaderFooterView \*)view).backgroundView.backgroundColor = [UIColor clearColor];

}

}

- (void)tableView:(UITableView \*)tableView willDisplayFooterView:(UIView \*)view forSection:(NSInteger)section{

if ([view isKindOfClass:[UITableViewHeaderFooterView class]]) {

view.tintColor = UIColor(0xf7f7f7, 1.0);

((UITableViewHeaderFooterView \*)view).backgroundView.backgroundColor = [UIColor clearColor];

}

}

@end

#import "PTBaseViewController.h"

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTPrivacyViewController : PTBaseViewController

@property (nonatomic, copy) void(^acceptButtonBlock)(void);

@end

NS\_ASSUME\_NONNULL\_END

#import "PTPrivacyViewController.h"

#import <YYText/YYText.h>

#import "PTWebViewController.h"

@interface PTPrivacyViewController ()

@property (weak, nonatomic) IBOutlet UIView \*alertView;

@end

@implementation PTPrivacyViewController

- (void)viewDidLoad {

[super viewDidLoad];

// Do any additional setup after loading the view from its nib.

UIColor \*color = [UIColor blackColor];

self.view.backgroundColor = [color colorWithAlphaComponent:0.5];

YYLabel \*\_yyLabel = [[YYLabel alloc]initWithFrame:CGRectMake(21, 79, 214, 220)];

\_yyLabel.textVerticalAlignment = YYTextVerticalAlignmentTop;

\_yyLabel.textAlignment = NSTextAlignmentJustified;

\_yyLabel.numberOfLines = 0;

[self.alertView addSubview:\_yyLabel];

//设置整段字符串的颜色

NSDictionary \*attributes = @{NSFontAttributeName:[UIFont systemFontOfSize:14], NSForegroundColorAttributeName: UIColor(0x000000, 0.7)};

NSMutableAttributedString \*text = [[NSMutableAttributedString alloc] initWithString:@"Welcome, We are committed to protecting your privacy. Our Terms of Service and Privacy Policy provide more details on how we collect and use data.\nBy clicking “Accept” below, you are indicating that you agree to our Terms of Service and have read and understand the Privacy Policy." attributes:attributes];

text.yy\_lineSpacing = 5;

//设置高亮色和点击事件

[text yy\_setTextHighlightRange:[[text string] rangeOfString:@"Terms of Service"] color:UIColor(0x288fff, 1) backgroundColor:[UIColor clearColor] tapAction:^(UIView \* \_Nonnull containerView, NSAttributedString \* \_Nonnull text, NSRange range, CGRect rect) {

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_TermsOfService" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

PTWebViewController \*vc = [[PTWebViewController alloc] init];

vc.url = @"https://www.gmgmapps.com/ptas/terms.html";

vc.title = @"Terms of service";

PTBaseNavigationController \*nav = [[PTBaseNavigationController alloc] initWithRootViewController:vc];

[self presentViewController:nav animated:YES completion:^{

}];

}];

//设置高亮色和点击事件

[text yy\_setTextHighlightRange:[[text string] rangeOfString:@"Privacy Policy"] color:UIColor(0x288fff, 1) backgroundColor:[UIColor clearColor] tapAction:^(UIView \* \_Nonnull containerView, NSAttributedString \* \_Nonnull text, NSRange range, CGRect rect) {

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_PrivacyPolicy" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

PTWebViewController \*vc = [[PTWebViewController alloc] init];

vc.url = @"https://www.gmgmapps.com/ptas/privacy.html";

vc.title = @"Privacy Policy";

PTBaseNavigationController \*nav = [[PTBaseNavigationController alloc] initWithRootViewController:vc];

[self presentViewController:nav animated:YES completion:^{

}];

}];

\_yyLabel.attributedText = text;

}

/\*

#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation

- (void)prepareForSegue:(UIStoryboardSegue \*)segue sender:(id)sender {

// Get the new view controller using [segue destinationViewController].

// Pass the selected object to the new view controller.

}

\*/

#pragma mark - Action

- (IBAction)acceptButtonAction:(id)sender {

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_AcceptButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

if (self.acceptButtonBlock) {

self.acceptButtonBlock();

}

}

- (void)didReceiveMemoryWarning {

[super didReceiveMemoryWarning];

}

@end

#import <UIKit/UIKit.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTAboutTableViewCell : UITableViewCell

@property (weak, nonatomic) IBOutlet UILabel \*nameLabel;

@property (weak, nonatomic) IBOutlet UIImageView \*rightArrow;

@property (weak, nonatomic) IBOutlet UILabel \*valueLabel;

@property (weak, nonatomic) IBOutlet UIView \*lineView;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTAboutTableViewCell.h"

@implementation PTAboutTableViewCell

- (void)awakeFromNib {

[super awakeFromNib];

// Initialization code

}

- (void)setSelected:(BOOL)selected animated:(BOOL)animated {

[super setSelected:selected animated:animated];

// Configure the view for the selected state

}

@end

#import "PTBaseViewController.h"

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTAboutViewController : PTBaseViewController

@end

NS\_ASSUME\_NONNULL\_END

#import "PTAboutViewController.h"

#import "PTAboutTableViewCell.h"

#import "PTWebViewController.h"

@interface PTAboutViewController ()<UITableViewDelegate, UITableViewDataSource>

@property (weak, nonatomic) IBOutlet UITableView \*tableView;

@property (nonatomic, strong) NSArray \*dataArray;

@end

@implementation PTAboutViewController

- (void)viewDidLoad {

[super viewDidLoad];

// Do any additional setup after loading the view from its nib.

self.title = NSLocalizedString(@"About", nil);

self.dataArray = @[@[@{@"name":NSLocalizedString(@"Share App", nil),@"value":@""},

@{@"name":NSLocalizedString(@"Ratings", nil),@"value":@""}],

@[@{@"name":NSLocalizedString(@"Privacy Policy", nil),@"value":@""},

@{@"name":NSLocalizedString(@"Terms of Service", nil),@"value":@""},

@{@"name":NSLocalizedString(@"Version Number", nil),@"value":@"1.0"}]];

[self.tableView registerNib:[UINib nibWithNibName:@"PTAboutTableViewCell" bundle:nil] forCellReuseIdentifier:@"PTAboutTableViewCell"];

}

/\*

#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation

- (void)prepareForSegue:(UIStoryboardSegue \*)segue sender:(id)sender {

// Get the new view controller using [segue destinationViewController].

// Pass the selected object to the new view controller.

}

\*/

#pragma mark - UITableViewDelegate, UITableViewDataSource

- (NSInteger)numberOfSectionsInTableView:(UITableView \*)tableView{

return self.dataArray.count;

}

- (NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section{

NSArray \*array = self.dataArray[section];

return array.count;

}

- (CGFloat)tableView:(UITableView \*)tableView heightForRowAtIndexPath:(NSIndexPath \*)indexPath{

return 44;

}

- (CGFloat)tableView:(UITableView \*)tableView heightForHeaderInSection:(NSInteger)section{

return 9.9;

}

- (CGFloat)tableView:(UITableView \*)tableView heightForFooterInSection:(NSInteger)section{

return 0.1;

}

- (UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath{

NSDictionary \*dic = self.dataArray[indexPath.section][indexPath.row];

PTAboutTableViewCell \*cell = [tableView dequeueReusableCellWithIdentifier:@"PTAboutTableViewCell" forIndexPath:indexPath];

cell.nameLabel.text = [dic objectForKey:@"name"];

NSString \*str = [dic objectForKey:@"value"];

if (str.length > 0) {

cell.rightArrow.hidden = YES;

cell.valueLabel.hidden = NO;

cell.valueLabel.text = str;

}else{

cell.rightArrow.hidden = NO;

cell.valueLabel.hidden = YES;

}

NSArray \*array = self.dataArray[indexPath.section];

if (indexPath.row == array.count-1) {

cell.lineView.hidden = YES;

}else{

cell.lineView.hidden = NO;

}

return cell;

}

- (void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath{

NSDictionary \*dic = self.dataArray[indexPath.section][indexPath.row];

NSString \*str = [dic objectForKey:@"name"];

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:[NSString stringWithFormat:@"btn\_selectCell\_%@", str] Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

if ([str isEqualToString:NSLocalizedString(@"Share App", nil)]) {

// 调用系统分享

// 分享内容

NSString \*shareTitle = @"Photo Translate & Image to text";

UIImage \*shareImage = [UIImage imageNamed:@"applogo"];

NSURL \*shareUrl = [NSURL URLWithString:@"https://apps.apple.com/us/app/id1532695188"];

NSArray \*activityItemsArray = @[shareTitle, shareImage, shareUrl];

// 调用分享

UIActivityViewController \*activityVC = [[UIActivityViewController alloc] initWithActivityItems:activityItemsArray applicationActivities:nil];

activityVC.modalInPopover = YES;

// ios8.0 之后用此方法回调

UIActivityViewControllerCompletionWithItemsHandler itemsBlock = ^(UIActivityType \_\_nullable activityType, BOOL completed, NSArray \* \_\_nullable returnedItems, NSError \* \_\_nullable activityError){

if (completed) {

NSLog(@"completed");

}else{

NSLog(@"cancel");

}

};

[self presentViewController:activityVC animated:YES completion:nil];

}else if ([str isEqualToString:NSLocalizedString(@"Ratings", nil)]){

NSString \*urlStr = [NSString stringWithFormat:@"itms-apps://itunes.apple.com/WebObjects/MZStore.woa/wa/viewContentsUserReviews?type=Purple+Software&id=%@&pageNumber=0&sortOrdering=2&mt=8", @"1532695188"];

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:urlStr] options:@{} completionHandler:^(BOOL success) {

}];

}else if ([str isEqualToString:NSLocalizedString(@"Privacy Policy", nil)]){

PTWebViewController \*vc = [[PTWebViewController alloc] init];

vc.url = @"https://www.gmgmapps.com/ptas/privacy.html";

vc.title = NSLocalizedString(@"Privacy Policy", nil);

PTBaseNavigationController \*nav = [[PTBaseNavigationController alloc] initWithRootViewController:vc];

[self presentViewController:nav animated:YES completion:^{

}];

}else if ([str isEqualToString:NSLocalizedString(@"Terms of Service", nil)]){

PTWebViewController \*vc = [[PTWebViewController alloc] init];

vc.url = @"https://www.gmgmapps.com/ptas/terms.html";

vc.title = NSLocalizedString(@"Terms of Service", nil);

PTBaseNavigationController \*nav = [[PTBaseNavigationController alloc] initWithRootViewController:vc];

[self presentViewController:nav animated:YES completion:^{

}];

}else if ([str isEqualToString:NSLocalizedString(@"Version Number", nil)]){

}else{

}

}

// tableView的UITableViewHeaderFooterView修改颜色为透明

- (void)tableView:(UITableView \*)tableView willDisplayHeaderView:(UIView \*)view forSection:(NSInteger)section{

if ([view isKindOfClass:[UITableViewHeaderFooterView class]]) {

view.tintColor = [UIColor clearColor];

((UITableViewHeaderFooterView \*)view).backgroundView.backgroundColor = [UIColor clearColor];

}

}

@end

#import <UIKit/UIKit.h>

#import "PTVideoPreview.h"

NS\_ASSUME\_NONNULL\_BEGIN

@class PTCameraView;

@protocol PTCameraViewDelegate <NSObject>

@optional;

/// 闪光灯

-(void)flashLightAction:(PTCameraView \*)cameraView handle:(void(^)(NSError \*error))handle;

/// 补光

-(void)torchLightAction:(PTCameraView \*)cameraView handle:(void(^)(NSError \*error))handle;

/// 转换摄像头

-(void)swicthCameraAction:(PTCameraView \*)cameraView handle:(void(^)(NSError \*error))handle;

/// 自动聚焦曝光

-(void)autoFocusAndExposureAction:(PTCameraView \*)cameraView handle:(void(^)(NSError \*error))handle;

/// 聚焦

-(void)focusAction:(PTCameraView \*)cameraView point:(CGPoint)point handle:(void(^)(NSError \*error))handle;

/// 曝光

-(void)exposAction:(PTCameraView \*)cameraView point:(CGPoint)point handle:(void(^)(NSError \*error))handle;

/// 缩放

-(void)zoomAction:(PTCameraView \*)cameraView factor:(CGFloat)factor;

/// 取消

-(void)cancelAction:(PTCameraView \*)cameraView;

/// 拍照

-(void)takePhotoAction:(PTCameraView \*)cameraView;

/// 停止录制视频

-(void)stopRecordVideoAction:(PTCameraView \*)cameraView;

/// 开始录制视频

-(void)startRecordVideoAction:(PTCameraView \*)cameraView;

/// 改变拍摄类型 1：拍照 2：视频

-(void)didChangeTypeAction:(PTCameraView \*)cameraView type:(NSInteger)type;

// 左边翻译源语言

- (void)leftLanguageAction;

// 右边翻译目标语言

- (void)rightLanguageAction;

// 中间切换button

- (void)convertAction;

// 右侧三个点更多

- (void)moreAction;

// 相册

- (void)photosAction;

@end

@interface PTCameraView : UIView

@property(nonatomic, weak) id <PTCameraViewDelegate> delegate;

@property(nonatomic, strong, readonly) PTVideoPreview \*previewView;

@property(nonatomic, assign, readonly) NSInteger type; // 1：拍照 2：视频

@property (nonatomic, strong) NSString \*fromLanguageButtonStr;// 左边源语言button展示文字

@property (nonatomic, strong) NSString \*toLanguageButtonStr;// 右边目标语言button展示文字

-(void)changeTorch:(BOOL)on;

-(void)changeFlash:(BOOL)on;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTCameraView.h"

#import "UIView+PTHUD.h"

@interface PTCameraView ()<GADBannerViewDelegate>

@property(nonatomic, assign) NSInteger type; // 1：拍照 2：视频

@property(nonatomic, strong) PTVideoPreview \*previewView;

@property(nonatomic, strong) UIView \*topView; // 上面的bar

@property(nonatomic, strong) UIView \*bottomView; // 下面的bar

@property(nonatomic, strong) UIView \*focusView; // 聚焦动画view

@property(nonatomic, strong) UIView \*exposureView; // 曝光动画view

@property(nonatomic, strong) UISlider \*slider;

@property(nonatomic, strong) UIButton \*torchBtn;

@property(nonatomic, strong) UIButton \*flashBtn;

@property(nonatomic, strong) UIButton \*photoBtn;

@property (nonatomic, strong) UIButton \*convertButton;// 中间转换button

@property (nonatomic, strong) UIButton \*leftLanguageButton;// 左边源语言button

@property (nonatomic, strong) UIButton \*rightLanguageButton;// 右边目标语言button

@property (nonatomic, strong) UIImageView \*rightArrowImageView;// 右边目标语言后的小箭头

@property(nonatomic, strong) GADBannerView \*bannerView; // 谷歌横幅广告

@end

@implementation PTCameraView

-(instancetype)initWithFrame:(CGRect)frame

{

NSAssert(frame.size.height>164 || frame.size.width>374, @"相机视图的高不小于164，宽不小于375");

self = [super initWithFrame:frame];

if (self) {

\_type = 1;

[self setupUI];

}

return self;

}

-(UIView \*)topView{

if (\_topView == nil) {

\_topView = [[UIView alloc]initWithFrame:CGRectMake(0, 0, self.width, StatusbarAndNavigationbarHeight)];

\_topView.backgroundColor = [UIColor clearColor];

// 加渐变

CAGradientLayer \*gradient = [CAGradientLayer layer];

//设置开始和结束位置(设置渐变的方向)

gradient.startPoint = CGPointMake(0, 0);

gradient.endPoint = CGPointMake(0, 1);

gradient.frame = CGRectMake(0, 0, self.width, StatusbarAndNavigationbarHeight);

gradient.colors = [NSArray arrayWithObjects:(id)[UIColor colorWithRed:0/255.0 green:0/255.0 blue:0/255.0 alpha:0.4].CGColor,(id)[UIColor colorWithRed:0/255.0 green:0/255.0 blue:0/255.0 alpha:0].CGColor,nil];

[\_topView.layer insertSublayer:gradient atIndex:0];

}

return \_topView;

}

-(UIView \*)bottomView{

if (\_bottomView == nil) {

if (is\_iPhoneX) {

\_bottomView = [[UIView alloc]initWithFrame:CGRectMake(0, self.height-171, self.width, 171)];

}else{

\_bottomView = [[UIView alloc]initWithFrame:CGRectMake(0, self.height-128, self.width, 128)];

}

\_bottomView.backgroundColor = [UIColor blackColor];

}

return \_bottomView;

}

-(UIView \*)focusView{

if (\_focusView == nil) {

\_focusView = [[UIView alloc] initWithFrame:CGRectMake(0.0f, 0.0f, 150, 150.0f)];

\_focusView.backgroundColor = [UIColor clearColor];

\_focusView.layer.borderColor = [UIColor blueColor].CGColor;

\_focusView.layer.borderWidth = 5.0f;

\_focusView.hidden = YES;

}

return \_focusView;

}

-(UIView \*)exposureView{

if (\_exposureView == nil) {

\_exposureView = [[UIView alloc] initWithFrame:CGRectMake(0.0f, 0.0f, 150, 150.0f)];

\_exposureView.backgroundColor = [UIColor clearColor];

\_exposureView.layer.borderColor = [UIColor purpleColor].CGColor;

\_exposureView.layer.borderWidth = 5.0f;

\_exposureView.hidden = YES;

}

return \_exposureView;

}

-(UISlider \*)slider{

if (\_slider == nil) {

\_slider = [[UISlider alloc] init];

\_slider.minimumValue = 0;

\_slider.maximumValue = 1;

\_slider.maximumTrackTintColor = [UIColor whiteColor];

\_slider.minimumTrackTintColor = [UIColor whiteColor];

\_slider.alpha = 0.0;

}

return \_slider;

}

-(void)setupUI{

if (is\_iPhoneX) {

self.previewView = [[PTVideoPreview alloc]initWithFrame:CGRectMake(0, 0, self.width, self.height-171)];

}else{

self.previewView = [[PTVideoPreview alloc]initWithFrame:CGRectMake(0, 0, self.width, self.height-128)];

}

[self addSubview:self.previewView];

[self.previewView addSubview:self.topView];

[self addSubview:self.bottomView];

[self.previewView addSubview:self.focusView];

[self.previewView addSubview:self.exposureView];

[self.previewView addSubview:self.slider];

// ----------------------- 手势

// 点击-->聚焦 双击-->曝光

UITapGestureRecognizer \*tap = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector(tapAction:)];

UITapGestureRecognizer \*doubleTap = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector(doubleTapAction:)];

doubleTap.numberOfTapsRequired = 2;

[self.previewView addGestureRecognizer:tap];

[self.previewView addGestureRecognizer:doubleTap];

[tap requireGestureRecognizerToFail:doubleTap];

// 捏合-->缩放

UIPinchGestureRecognizer \*pinch = [[UIPinchGestureRecognizer alloc] initWithTarget:self action: @selector(pinchAction:)];

[self.previewView addGestureRecognizer:pinch];

// ----------------------- UI

// 缩放

self.slider.transform = CGAffineTransformMakeRotation(M\_PI\_2);

self.slider.frame = CGRectMake(SCREEN\_WIDTH-50, 50, 15, 200);

// 拍照

UIButton \*photoButton = [UIButton buttonWithType:UIButtonTypeCustom];

// photoButton.backgroundColor = [UIColor redColor];

[photoButton setImage:[UIImage imageNamed:@"tack\_photo"] forState:UIControlStateNormal];

if (is\_iPhoneX) {

photoButton.frame = CGRectMake((self.bottomView.width-64)/2, 12, 64, 64);

}else{

photoButton.frame = CGRectMake((self.bottomView.width-64)/2, 5, 64, 64);

}

[photoButton addTarget:self action:@selector(takePicture:) forControlEvents:UIControlEventTouchUpInside];

[self.bottomView addSubview:photoButton];

\_photoBtn = photoButton;

// 取消

UIButton \*cancelButton = [UIButton buttonWithType:UIButtonTypeCustom];

[cancelButton setTitle:@"取消" forState:UIControlStateNormal];

[cancelButton setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];

[cancelButton addTarget:self action:@selector(cancel:) forControlEvents:UIControlEventTouchUpInside];

[cancelButton sizeToFit];

cancelButton.center = CGPointMake(40, \_bottomView.height/2);

// [self.bottomView addSubview:cancelButton];

// 照片类型

UIButton \*typeButton = [UIButton buttonWithType:UIButtonTypeCustom];

[typeButton setTitle:@"照片" forState:UIControlStateNormal];

[typeButton setTitle:@"视频" forState:UIControlStateSelected];

[typeButton setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];

[typeButton addTarget:self action:@selector(changeType:) forControlEvents:UIControlEventTouchUpInside];

[typeButton sizeToFit];

typeButton.center = CGPointMake(\_bottomView.width-60, \_bottomView.height/2);

// [self.bottomView addSubview:typeButton];

// 相册

UIButton \*photosButton = [[UIButton alloc] init];

// photosButton.backgroundColor = [UIColor redColor];

if (is\_iPhoneX) {

photosButton.frame = CGRectMake(36, 26, 36, 36);

}else{

photosButton.frame = CGRectMake(36, 19, 36, 36);

}

[photosButton setImage:[UIImage imageNamed:@"photos"] forState:UIControlStateNormal];

[photosButton addTarget:self action:@selector(photosButtonAction:) forControlEvents:UIControlEventTouchUpInside];

[self.bottomView addSubview:photosButton];

// 闪光灯

UIButton \*newFlashButton = [[UIButton alloc] init];

// newFlashButton.backgroundColor = [UIColor redColor];

if (is\_iPhoneX) {

newFlashButton.frame = CGRectMake(self.bottomView.width -36 -36, 26, 36, 36);

}else{

newFlashButton.frame = CGRectMake(self.bottomView.width -36 -36, 19, 36, 36);

}

[newFlashButton setImage:[UIImage imageNamed:@"flash\_no"] forState:UIControlStateNormal];

[newFlashButton setImage:[UIImage imageNamed:@"flash\_yes"] forState:UIControlStateSelected];

[newFlashButton addTarget:self action:@selector(flashClick:) forControlEvents:UIControlEventTouchUpInside];

[self.bottomView addSubview:newFlashButton];

\_flashBtn = newFlashButton;

// 转换前后摄像头

UIButton \*switchCameraButton = [UIButton buttonWithType:UIButtonTypeCustom];

[switchCameraButton setTitle:@"转换摄像头" forState:UIControlStateNormal];

[switchCameraButton setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];

[switchCameraButton addTarget:self action:@selector(switchCameraClick:) forControlEvents:UIControlEventTouchUpInside];

[switchCameraButton sizeToFit];

switchCameraButton.center = CGPointMake(switchCameraButton.width/2+10, \_topView.height/2);

// [self.topView addSubview:switchCameraButton];

// 补光

UIButton \*lightButton = [UIButton buttonWithType:UIButtonTypeCustom];

[lightButton setTitle:@"补光" forState:UIControlStateNormal];

[lightButton setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];

[lightButton setTitleColor:[UIColor blueColor] forState:UIControlStateSelected];

[lightButton addTarget:self action:@selector(torchClick:) forControlEvents:UIControlEventTouchUpInside];

[lightButton sizeToFit];

lightButton.center = CGPointMake(lightButton.width/2 + switchCameraButton.right+10, \_topView.height/2);

// [self.topView addSubview:lightButton];

\_torchBtn = lightButton;

// 重置对焦、曝光

UIButton \*focusAndExposureButton = [UIButton buttonWithType:UIButtonTypeCustom];

[focusAndExposureButton setTitle:@"自动聚焦/曝光" forState:UIControlStateNormal];

[focusAndExposureButton setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];

[focusAndExposureButton addTarget:self action:@selector(focusAndExposureClick:) forControlEvents:UIControlEventTouchUpInside];

[focusAndExposureButton sizeToFit];

focusAndExposureButton.center = CGPointMake(focusAndExposureButton.width/2 + lightButton.right+10, \_topView.height/2);

// [self.topView addSubview:focusAndExposureButton];

// 中间转换button

self.convertButton = [[UIButton alloc] initWithFrame:CGRectMake((self.topView.width - 56)/2, self.topView.height -4 -36, 56, 36)];

[self.convertButton setImage:[UIImage imageNamed:@"convert"] forState:UIControlStateNormal];

[self.convertButton addTarget:self action:@selector(convertButtonAction:) forControlEvents:UIControlEventTouchUpInside];

[self.topView addSubview:self.convertButton];

// 左边源语言button

self.leftLanguageButton= [[UIButton alloc] init];

// self.leftLanguageButton.backgroundColor = [UIColor redColor];

// [self.leftLanguageButton setTitle:@"English" forState:UIControlStateNormal];

[self.leftLanguageButton setContentHorizontalAlignment:UIControlContentHorizontalAlignmentRight];

self.leftLanguageButton.titleLabel.lineBreakMode = NSLineBreakByTruncatingTail;//省略号靠右侧

[self.leftLanguageButton setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];

self.leftLanguageButton.titleLabel.font = [UIFont systemFontOfSize:16];

[self.leftLanguageButton addTarget:self action:@selector(leftLanguageButtonAction:) forControlEvents:UIControlEventTouchUpInside];

// 根据button文字内容计算宽度

CGSize buttonSize0 = [self.leftLanguageButton.titleLabel.text boundingRectWithSize:CGSizeMake(MAXFLOAT, 36)

options:NSStringDrawingTruncatesLastVisibleLine | NSStringDrawingUsesLineFragmentOrigin | NSStringDrawingUsesFontLeading

attributes:@{ NSFontAttributeName:self.leftLanguageButton.titleLabel.font}

context:nil].size;

self.leftLanguageButton.frame = CGRectMake(self.convertButton.frame.origin.x -16 -(buttonSize0.width>90?90:buttonSize0.width), self.convertButton.frame.origin.y, buttonSize0.width>90?90:buttonSize0.width, 36);

[self.topView addSubview:self.leftLanguageButton];

UIImageView \*leftArrowImageView = [[UIImageView alloc] initWithFrame:CGRectMake(self.leftLanguageButton.right + 4, self.leftLanguageButton.top + 14, 8, 8)];

leftArrowImageView.image = [UIImage imageNamed:@"down\_arrow"];

[self.topView addSubview:leftArrowImageView];

// 右边目标语言button

self.rightLanguageButton = [[UIButton alloc] init];

// self.rightLanguageButton.backgroundColor = [UIColor greenColor];

// [self.rightLanguageButton setTitle:@"Chinese (Simplified)" forState:UIControlStateNormal];

[self.rightLanguageButton setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];

self.rightLanguageButton.titleLabel.lineBreakMode = NSLineBreakByTruncatingTail;//省略号靠右侧

[self.rightLanguageButton setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];

self.rightLanguageButton.titleLabel.font = [UIFont systemFontOfSize:16];

[self.rightLanguageButton addTarget:self action:@selector(rightLanguageButtonAction:) forControlEvents:UIControlEventTouchUpInside];

// 根据button文字内容计算宽度

CGSize buttonSize = [self.rightLanguageButton.titleLabel.text boundingRectWithSize:CGSizeMake(MAXFLOAT, 36)

options:NSStringDrawingTruncatesLastVisibleLine | NSStringDrawingUsesLineFragmentOrigin | NSStringDrawingUsesFontLeading

attributes:@{ NSFontAttributeName:self.rightLanguageButton.titleLabel.font}

context:nil].size;

self.rightLanguageButton.frame = CGRectMake(self.convertButton.right +4, self.convertButton.frame.origin.y, buttonSize.width>90?90:buttonSize.width, 36);

[self.topView addSubview:self.rightLanguageButton];

self.rightArrowImageView = [[UIImageView alloc] initWithFrame:CGRectMake(self.rightLanguageButton.right + 4, self.rightLanguageButton.top + 14, 8, 8)];

self.rightArrowImageView.image = [UIImage imageNamed:@"down\_arrow"];

[self.topView addSubview:self.rightArrowImageView];

// 右侧三个点

UIButton \*moreButton = [[UIButton alloc] initWithFrame:CGRectMake(self.topView.width - 12 - 36, self.convertButton.frame.origin.y, 36, 36)];

[moreButton setImage:[UIImage imageNamed:@"more"] forState:UIControlStateNormal];

[moreButton addTarget:self action:@selector(moreButtonAction:) forControlEvents:UIControlEventTouchUpInside];

[self.topView addSubview:moreButton];

self.bannerView = [[GADBannerView alloc]

initWithAdSize:kGADAdSizeBanner];

[self addBannerViewToView:self.bannerView];

self.bannerView.adUnitID = @"ca-app-pub-9104226499574771/3090627881";

self.bannerView.rootViewController = self;

[self.bannerView loadRequest:[GADRequest request]];

self.bannerView.delegate = self;

}

- (void)addBannerViewToView:(UIView \*)bannerView {

bannerView.translatesAutoresizingMaskIntoConstraints = NO;

[self.bottomView addSubview:bannerView];

if (is\_iPhoneX) {

[self.bottomView addConstraints:@[

[NSLayoutConstraint constraintWithItem:bannerView

attribute:NSLayoutAttributeBottom

relatedBy:NSLayoutRelationEqual

toItem:self.bottomView

attribute:NSLayoutAttributeBottom

multiplier:1

constant:-34],

[NSLayoutConstraint constraintWithItem:bannerView

attribute:NSLayoutAttributeCenterX

relatedBy:NSLayoutRelationEqual

toItem:self.photoBtn

attribute:NSLayoutAttributeCenterX

multiplier:1

constant:0]

]];

}else{

[self.bottomView addConstraints:@[

[NSLayoutConstraint constraintWithItem:bannerView

attribute:NSLayoutAttributeBottom

relatedBy:NSLayoutRelationEqual

toItem:self.bottomView

attribute:NSLayoutAttributeBottom

multiplier:1

constant:-4],

[NSLayoutConstraint constraintWithItem:bannerView

attribute:NSLayoutAttributeCenterX

relatedBy:NSLayoutRelationEqual

toItem:self.photoBtn

attribute:NSLayoutAttributeCenterX

multiplier:1

constant:0]

]];

}

}

-(void)changeTorch:(BOOL)on{

\_torchBtn.selected = on;

}

-(void)changeFlash:(BOOL)on{

\_flashBtn.selected = on;

}

-(void)pinchAction:(UIPinchGestureRecognizer \*)pinch {

if ([\_delegate respondsToSelector:@selector(zoomAction:factor:)]) {

if (pinch.state == UIGestureRecognizerStateBegan) {

[UIView animateWithDuration:0.1 animations:^{

self->\_slider.alpha = 1;

}];

} else if (pinch.state == UIGestureRecognizerStateChanged) {

if (pinch.velocity > 0) {

\_slider.value += pinch.velocity/100;

} else {

\_slider.value += pinch.velocity/20;

}

[\_delegate zoomAction:self factor: powf(5, \_slider.value)];

} else {

[UIView animateWithDuration:0.1 animations:^{

self->\_slider.alpha = 0.0;

}];

}

}

}

// 聚焦

-(void)tapAction:(UIGestureRecognizer \*)tap {

if ([\_delegate respondsToSelector:@selector(focusAction:point:handle:)]) {

CGPoint point = [tap locationInView:self.previewView];

[self runFocusAnimation:self.focusView point:point];

[\_delegate focusAction:self point:[self.previewView captureDevicePointForPoint:point] handle:^(NSError \*error) {

if (error) [self showError:error];

}];

}

}

// 曝光

-(void)doubleTapAction:(UIGestureRecognizer \*)tap {

if ([\_delegate respondsToSelector:@selector(exposAction:point:handle:)]) {

CGPoint point = [tap locationInView:self.previewView];

[self runFocusAnimation:self.exposureView point:point];

[\_delegate exposAction:self point:[self.previewView captureDevicePointForPoint:point] handle:^(NSError \*error) {

if (error) [self showError:error];

}];

}

}

// 自动聚焦和曝光

-(void)focusAndExposureClick:(UIButton \*)button {

if ([\_delegate respondsToSelector:@selector(autoFocusAndExposureAction:handle:)]) {

[self runResetAnimation];

[\_delegate autoFocusAndExposureAction:self handle:^(NSError \*error) {

if (error) [self showError:error];

}];

}

}

// 拍照、视频

-(void)takePicture:(UIButton \*)btn {

if (self.type == 1) {

if ([\_delegate respondsToSelector:@selector(takePhotoAction:)]) {

[\_delegate takePhotoAction:self];

}

} else {

if (btn.selected == YES) {

// 结束

btn.selected = NO;

[\_photoBtn setTitle:@"开始" forState:UIControlStateNormal];

if ([\_delegate respondsToSelector:@selector(stopRecordVideoAction:)]) {

[\_delegate stopRecordVideoAction:self];

}

} else {

// 开始

btn.selected = YES;

[\_photoBtn setTitle:@"结束" forState:UIControlStateNormal];

if ([\_delegate respondsToSelector:@selector(startRecordVideoAction:)]) {

[\_delegate startRecordVideoAction:self];

}

}

}

}

// 取消

-(void)cancel:(UIButton \*)btn {

if ([\_delegate respondsToSelector:@selector(cancelAction:)]) {

[\_delegate cancelAction:self];

}

}

// 转换拍摄类型

-(void)changeType:(UIButton \*)btn {

btn.selected = !btn.selected;

self.type = self.type == 1?2:1;

if (self.type == 1) {

[\_photoBtn setTitle:@"拍照" forState:UIControlStateNormal];

} else {

[\_photoBtn setTitle:@"开始" forState:UIControlStateNormal];

}

if ([\_delegate respondsToSelector:@selector(didChangeTypeAction:type:)]) {

[\_delegate didChangeTypeAction:self type:self.type == 1?2:1];

}

}

// 转换摄像头

-(void)switchCameraClick:(UIButton \*)btn {

if ([\_delegate respondsToSelector:@selector(swicthCameraAction:handle:)]) {

[\_delegate swicthCameraAction:self handle:^(NSError \*error) {

if (error) [self showError:error];

}];

}

}

// 手电筒

-(void)torchClick:(UIButton \*)btn {

if ([\_delegate respondsToSelector:@selector(torchLightAction:handle:)]) {

[\_delegate torchLightAction:self handle:^(NSError \*error) {

if (error) {

[self showError:error];

} else {

self->\_flashBtn.selected = NO;

self->\_torchBtn.selected = !self->\_torchBtn.selected;

}

}];

}

}

// 闪光灯

-(void)flashClick:(UIButton \*)btn {

if ([\_delegate respondsToSelector:@selector(flashLightAction:handle:)]) {

[\_delegate flashLightAction:self handle:^(NSError \*error) {

if (error) {

[self showError:error];

} else {

self->\_flashBtn.selected = !self->\_flashBtn.selected;

self->\_torchBtn.selected = NO;

}

}];

}

}

// 切换翻译源、目标语言

- (void)convertButtonAction:(id)sender{

// NSLog(@"切换语言");

if ([\_delegate respondsToSelector:@selector(convertAction)]) {

[\_delegate convertAction];

}

}

// 左边源语言

- (void)leftLanguageButtonAction:(id)sender{

// NSLog(@"左边源语言");

if ([\_delegate respondsToSelector:@selector(leftLanguageAction)]) {

[\_delegate leftLanguageAction];

}

}

// 右边目标语言

- (void)rightLanguageButtonAction:(id)sender{

// NSLog(@"右边目标语言");

if ([\_delegate respondsToSelector:@selector(rightLanguageAction)]) {

[\_delegate rightLanguageAction];

}

}

// 右侧三个点更多

- (void)moreButtonAction:(id)sender{

// NSLog(@"右侧三个点更多");

if ([\_delegate respondsToSelector:@selector(moreAction)]) {

[\_delegate moreAction];

}

}

// 相册

- (void)photosButtonAction:(id)sender{

// NSLog(@"相册");

if ([\_delegate respondsToSelector:@selector(photosAction)]) {

[\_delegate photosAction];

}

}

#pragma mark - Private methods

// 聚焦、曝光动画

-(void)runFocusAnimation:(UIView \*)view point:(CGPoint)point {

view.center = point;

view.hidden = NO;

[UIView animateWithDuration:0.15f delay:0.0f options:UIViewAnimationOptionCurveEaseInOut animations:^{

view.layer.transform = CATransform3DMakeScale(0.5, 0.5, 1.0);

} completion:^(BOOL complete) {

double delayInSeconds = 0.5f;

dispatch\_time\_t popTime = dispatch\_time(DISPATCH\_TIME\_NOW, (int64\_t)(delayInSeconds \* NSEC\_PER\_SEC));

dispatch\_after(popTime, dispatch\_get\_main\_queue(), ^(void){

view.hidden = YES;

view.transform = CGAffineTransformIdentity;

});

}];

}

// 自动聚焦、曝光动画

- (void)runResetAnimation {

self.focusView.center = CGPointMake(self.previewView.width/2, self.previewView.height/2);

self.exposureView.center = CGPointMake(self.previewView.width/2, self.previewView.height/2);;

self.exposureView.transform = CGAffineTransformMakeScale(1.2f, 1.2f);

self.focusView.hidden = NO;

self.focusView.hidden = NO;

[UIView animateWithDuration:0.15f delay:0.0f options:UIViewAnimationOptionCurveEaseInOut animations:^{

self.focusView.layer.transform = CATransform3DMakeScale(0.5, 0.5, 1.0);

self.exposureView.layer.transform = CATransform3DMakeScale(0.7, 0.7, 1.0);

} completion:^(BOOL complete) {

double delayInSeconds = 0.5f;

dispatch\_time\_t popTime = dispatch\_time(DISPATCH\_TIME\_NOW, (int64\_t)(delayInSeconds \* NSEC\_PER\_SEC));

dispatch\_after(popTime, dispatch\_get\_main\_queue(), ^(void){

self.focusView.hidden = YES;

self.exposureView.hidden = YES;

self.focusView.transform = CGAffineTransformIdentity;

self.exposureView.transform = CGAffineTransformIdentity;

});

}];

}

#pragma mark - set get

- (void)setFromLanguageButtonStr:(NSString \*)fromLanguageButtonStr{

\_fromLanguageButtonStr = fromLanguageButtonStr;

[self.leftLanguageButton setTitle:fromLanguageButtonStr forState:UIControlStateNormal];

// 根据button文字内容计算宽度

CGSize buttonSize0 = [self.leftLanguageButton.titleLabel.text boundingRectWithSize:CGSizeMake(MAXFLOAT, 36)

options:NSStringDrawingTruncatesLastVisibleLine | NSStringDrawingUsesLineFragmentOrigin | NSStringDrawingUsesFontLeading

attributes:@{ NSFontAttributeName:self.leftLanguageButton.titleLabel.font}

context:nil].size;

self.leftLanguageButton.frame = CGRectMake(self.convertButton.frame.origin.x -16 -(buttonSize0.width>90?90:buttonSize0.width), self.convertButton.frame.origin.y, buttonSize0.width>90?90:buttonSize0.width, 36);

}

- (void)setToLanguageButtonStr:(NSString \*)toLanguageButtonStr{

\_toLanguageButtonStr = toLanguageButtonStr;

[self.rightLanguageButton setTitle:toLanguageButtonStr forState:UIControlStateNormal];

// 根据button文字内容计算宽度

CGSize buttonSize = [self.rightLanguageButton.titleLabel.text boundingRectWithSize:CGSizeMake(MAXFLOAT, 36)

options:NSStringDrawingTruncatesLastVisibleLine | NSStringDrawingUsesLineFragmentOrigin | NSStringDrawingUsesFontLeading

attributes:@{ NSFontAttributeName:self.rightLanguageButton.titleLabel.font}

context:nil].size;

self.rightLanguageButton.frame = CGRectMake(self.convertButton.right +4, self.convertButton.frame.origin.y, buttonSize.width>90?90:buttonSize.width, 36);

self.rightArrowImageView.frame = CGRectMake(self.rightLanguageButton.right + 4, self.rightLanguageButton.top + 14, 8, 8);

}

#pragma mark - GADBannerViewDelegate

/// Tells the delegate an ad request loaded an ad.

///告诉委托人广告请求已加载广告。

- (void)adViewDidReceiveAd:(GADBannerView \*)adView {

NSLog(@"adViewDidReceiveAd");

}

/// Tells the delegate an ad request failed.

///告诉委托人广告请求失败。

- (void)adView:(GADBannerView \*)adView

didFailToReceiveAdWithError:(GADRequestError \*)error {

NSLog(@"adView:didFailToReceiveAdWithError: %@", [error localizedDescription]);

}

/// Tells the delegate that a full-screen view will be presented in response

/// to the user clicking on an ad.

///告诉代表将以全屏视图显示

///给点击广告的用户。

- (void)adViewWillPresentScreen:(GADBannerView \*)adView {

NSLog(@"adViewWillPresentScreen");

}

/// Tells the delegate that the full-screen view will be dismissed.

///告诉代表该全屏视图将被关闭。

- (void)adViewWillDismissScreen:(GADBannerView \*)adView {

NSLog(@"adViewWillDismissScreen");

}

/// Tells the delegate that the full-screen view has been dismissed.

///告诉代表该全屏视图已关闭。

- (void)adViewDidDismissScreen:(GADBannerView \*)adView {

NSLog(@"adViewDidDismissScreen");

}

/// Tells the delegate that a user click will open another app (such as

/// the App Store), backgrounding the current app.

///告诉代表，用户单击将打开另一个应用程序（例如

/// App Store），以当前应用为背景。

- (void)adViewWillLeaveApplication:(GADBannerView \*)adView {

NSLog(@"adViewWillLeaveApplication");

}

@end

#import <UIKit/UIKit.h>

#import <AVFoundation/AVFoundation.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTVideoPreview : UIView

@property (strong, nonatomic) AVCaptureSession \*captureSessionsion;

- (CGPoint)captureDevicePointForPoint:(CGPoint)point;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTVideoPreview.h"

@implementation PTVideoPreview

- (instancetype)initWithFrame:(CGRect)frame

{

self = [super initWithFrame:frame];

if (self) {

[(AVCaptureVideoPreviewLayer \*)self.layer setVideoGravity:AVLayerVideoGravityResizeAspectFill];

}

return self;

}

- (AVCaptureSession\*)captureSessionsion {

return [(AVCaptureVideoPreviewLayer\*)self.layer session];

}

- (void)setCaptureSessionsion:(AVCaptureSession \*)session {

[(AVCaptureVideoPreviewLayer\*)self.layer setSession:session];

}

- (CGPoint)captureDevicePointForPoint:(CGPoint)point {

AVCaptureVideoPreviewLayer \*layer = (AVCaptureVideoPreviewLayer \*)self.layer;

return [layer captureDevicePointOfInterestForPoint:point];

}

// 使该view的layer方法返回AVCaptureVideoPreviewLayer对象

+ (Class)layerClass {

return [AVCaptureVideoPreviewLayer class];

}

@end

#import <Foundation/Foundation.h>

#import <AVFoundation/AVFoundation.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTMovieManager : NSObject

@property(nonatomic, assign) AVCaptureVideoOrientation referenceOrientation; // 视频播放方向

@property(nonatomic, assign) AVCaptureVideoOrientation currentOrientation;

@property(nonatomic, strong) AVCaptureDevice \*currentDevice;

- (void)start:(void(^)(NSError \*error))handle;

- (void)stop:(void(^)(NSURL \*url, NSError \*error))handle;

- (void)writeData:(AVCaptureConnection \*)connection

video:(AVCaptureConnection\*)video

// audio:(AVCaptureConnection \*)audio

buffer:(CMSampleBufferRef)buffer;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTMovieManager.h"

@interface PTMovieManager ()

{

BOOL \_readyToRecordVideo;

// BOOL \_readyToRecordAudio;

dispatch\_queue\_t \_movieWritingQueue;

NSURL \*\_movieURL;

AVAssetWriter \*\_movieWriter;

// AVAssetWriterInput \*\_movieAudioInput;

AVAssetWriterInput \*\_movieVideoInput;

}

@end

@implementation PTMovieManager

- (instancetype)init {

self = [super init];

if (self) {

\_movieWritingQueue = dispatch\_queue\_create("Movie.Writing.Queue", DISPATCH\_QUEUE\_SERIAL);

\_movieURL = [NSURL fileURLWithPath:[NSString stringWithFormat:@"%@%@", NSTemporaryDirectory(), @"movie.mov"]];

\_referenceOrientation = AVCaptureVideoOrientationPortrait;

}

return self;

}

- (void)start:(void(^)(NSError \*error))handle{

[self removeFile:\_movieURL];

dispatch\_async(\_movieWritingQueue, ^{

NSError \*error;

if (!self->\_movieWriter) {

self->\_movieWriter = [[AVAssetWriter alloc] initWithURL:self->\_movieURL fileType:AVFileTypeQuickTimeMovie error:&error];

}

handle(error);

});

}

- (void)stop:(void(^)(NSURL \*url, NSError \*error))handle{

\_readyToRecordVideo = NO;

// \_readyToRecordAudio = NO;

dispatch\_async(\_movieWritingQueue, ^{

[self->\_movieWriter finishWritingWithCompletionHandler:^(){

if (self->\_movieWriter.status == AVAssetWriterStatusCompleted) {

dispatch\_async(dispatch\_get\_main\_queue(), ^{

handle(self->\_movieURL, nil);

});

} else {

handle(nil, self->\_movieWriter.error);

}

self->\_movieWriter = nil;

}];

});

}

//- (void)writeData:(AVCaptureConnection \*)connection video:(AVCaptureConnection\*)video audio:(AVCaptureConnection \*)audio buffer:(CMSampleBufferRef)buffer {

- (void)writeData:(AVCaptureConnection \*)connection video:(AVCaptureConnection \*)video buffer:(CMSampleBufferRef)buffer{

CFRetain(buffer);

dispatch\_async(\_movieWritingQueue, ^{

if (connection == video){

if (!self->\_readyToRecordVideo){

self->\_readyToRecordVideo = [self setupAssetWriterVideoInput:CMSampleBufferGetFormatDescription(buffer)] == nil;

}

CFRelease(buffer);

});

}

- (void)writeSampleBuffer:(CMSampleBufferRef)sampleBuffer ofType:(NSString \*)mediaType{

if (\_movieWriter.status == AVAssetWriterStatusUnknown){

if ([\_movieWriter startWriting]){

[\_movieWriter startSessionAtSourceTime:CMSampleBufferGetPresentationTimeStamp(sampleBuffer)];

} else {

NSLog(@"%@", \_movieWriter.error);

}

}

if (\_movieWriter.status == AVAssetWriterStatusWriting){

if (mediaType == AVMediaTypeVideo){

if (!\_movieVideoInput.readyForMoreMediaData){

return;

}

if (![\_movieVideoInput appendSampleBuffer:sampleBuffer]){

NSLog(@"%@", \_movieWriter.error);

}

}

}

}

/// 视频源数据写入配置

- (NSError \*)setupAssetWriterVideoInput:(CMFormatDescriptionRef)currentFormatDescription {

CMVideoDimensions dimensions = CMVideoFormatDescriptionGetDimensions(currentFormatDescription);

NSUInteger numPixels = dimensions.width \* dimensions.height;

CGFloat bitsPerPixel = numPixels < (640 \* 480) ? 4.05 : 11.0;

NSDictionary \*compression = @{AVVideoAverageBitRateKey: [NSNumber numberWithInteger: numPixels \* bitsPerPixel],

AVVideoMaxKeyFrameIntervalKey: [NSNumber numberWithInteger:30]};

NSDictionary \*settings = @{AVVideoCodecKey: AVVideoCodecH264,

AVVideoWidthKey: [NSNumber numberWithInteger:dimensions.width],

AVVideoHeightKey: [NSNumber numberWithInteger:dimensions.height],

AVVideoCompressionPropertiesKey: compression};

if ([\_movieWriter canApplyOutputSettings:settings forMediaType:AVMediaTypeVideo]){

\_movieVideoInput = [AVAssetWriterInput assetWriterInputWithMediaType:AVMediaTypeVideo outputSettings:settings];

\_movieVideoInput.expectsMediaDataInRealTime = YES;

\_movieVideoInput.transform = [self transformFromCurrentVideoOrientationToOrientation:self.referenceOrientation];

if ([\_movieWriter canAddInput:\_movieVideoInput]){

[\_movieWriter addInput:\_movieVideoInput];

} else {

return \_movieWriter.error;

}

} else {

return \_movieWriter.error;

}

return nil;

}

// 获取视频旋转矩阵

- (CGAffineTransform)transformFromCurrentVideoOrientationToOrientation:(AVCaptureVideoOrientation)orientation{

CGFloat orientationAngleOffset = [self angleOffsetFromPortraitOrientationToOrientation:orientation];

CGFloat videoOrientationAngleOffset = [self angleOffsetFromPortraitOrientationToOrientation:self.currentOrientation];

CGFloat angleOffset;

if (self.currentDevice.position == AVCaptureDevicePositionBack) {

angleOffset = videoOrientationAngleOffset - orientationAngleOffset + M\_PI\_2;

} else {

angleOffset = orientationAngleOffset - videoOrientationAngleOffset + M\_PI\_2;

}

CGAffineTransform transform = CGAffineTransformMakeRotation(angleOffset);

return transform;

}

// 获取视频旋转角度

- (CGFloat)angleOffsetFromPortraitOrientationToOrientation:(AVCaptureVideoOrientation)orientation{

CGFloat angle = 0.0;

switch (orientation){

case AVCaptureVideoOrientationPortrait:

angle = 0.0;

break;

case AVCaptureVideoOrientationPortraitUpsideDown:

angle = M\_PI;

break;

case AVCaptureVideoOrientationLandscapeRight:

angle = -M\_PI\_2;

break;

case AVCaptureVideoOrientationLandscapeLeft:

angle = M\_PI\_2;

break;

}

return angle;

}

// 移除文件

- (void)removeFile:(NSURL \*)fileURL{

NSFileManager \*fileManager = [NSFileManager defaultManager];

NSString \*filePath = fileURL.path;

if ([fileManager fileExistsAtPath:filePath]){

NSError \*error;

BOOL success = [fileManager removeItemAtPath:filePath error:&error];

if (!success){

NSLog(@"删除视频文件失败：%@", error);

} else {

NSLog(@"删除视频文件成功");

}

}

}

@end

#import <Foundation/Foundation.h>

#import <AVFoundation/AVFoundation.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTCameraManager : NSObject

- (AVCaptureDeviceInput \*)switchCamera:(AVCaptureSession \*)session

old:(AVCaptureDeviceInput \*)oldinput

new:(AVCaptureDeviceInput \*)newinput;

- (id)resetFocusAndExposure:(AVCaptureDevice \*)device;

- (id)zoom:(AVCaptureDevice \*)device factor:(CGFloat)factor;

- (id)focus:(AVCaptureDevice \*)device point:(CGPoint)point;

- (id)expose:(AVCaptureDevice \*)device point:(CGPoint)point;

- (id)changeFlash:(AVCaptureDevice \*)device mode:(AVCaptureFlashMode)mode;

- (id)changeTorch:(AVCaptureDevice \*)device model:(AVCaptureTorchMode)mode;

- (AVCaptureFlashMode)flashMode:(AVCaptureDevice \*)device;

- (AVCaptureTorchMode)torchMode:(AVCaptureDevice \*)device;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTCameraManager.h"

@implementation PTCameraManager

#pragma mark - -转换摄像头

- (AVCaptureDeviceInput \*)switchCamera:(AVCaptureSession \*)session old:(AVCaptureDeviceInput \*)oldinput new:(AVCaptureDeviceInput \*)newinput {

[session beginConfiguration];

[session removeInput:oldinput];

if ([session canAddInput:newinput]) {

[session addInput:newinput];

[session commitConfiguration];

return newinput;

} else {

[session addInput:oldinput];

[session commitConfiguration];

return oldinput;

}

}

#pragma mark - -缩放

- (id)zoom:(AVCaptureDevice \*)device factor:(CGFloat)factor {

if (device.activeFormat.videoMaxZoomFactor > factor && factor >= 1.0) {

NSError \*error;

if ([device lockForConfiguration:&error]) {

[device rampToVideoZoomFactor:factor withRate:4.0];

[device unlockForConfiguration];

}

return error;

}

return [self error:@"不支持的缩放倍数" code:2000];

}

#pragma mark - -聚焦

- (id)focus:(AVCaptureDevice \*)device point:(CGPoint)point{

BOOL supported = [device isFocusPointOfInterestSupported] &&

[device isFocusModeSupported:AVCaptureFocusModeAutoFocus];

if (supported){

NSError \*error;

if ([device lockForConfiguration:&error]) {

device.focusPointOfInterest = point;

device.focusMode = AVCaptureFocusModeAutoFocus;

[device unlockForConfiguration];

}

return error;

}

return [self error:@"设备不支持对焦" code:2001];

}

#pragma mark - -曝光

static const NSString \*CameraAdjustingExposureContext;

- (id)expose:(AVCaptureDevice \*)device point:(CGPoint)point{

BOOL supported = [device isExposurePointOfInterestSupported] &&

[device isExposureModeSupported:AVCaptureExposureModeContinuousAutoExposure];

if (supported) {

NSError \*error;

if ([device lockForConfiguration:&error]) {

device.exposurePointOfInterest = point;

device.exposureMode = AVCaptureExposureModeContinuousAutoExposure;

if ([device isExposureModeSupported:AVCaptureExposureModeLocked]) {

[device addObserver:self forKeyPath:@"adjustingExposure" options:NSKeyValueObservingOptionNew context:&CameraAdjustingExposureContext];

}

[device unlockForConfiguration];

}

return error;

}

return [self error:@"设备不支持曝光" code:2002];

}

- (void)observeValueForKeyPath:(NSString \*)keyPath ofObject:(id)object change:(NSDictionary \*)change context:(void \*)context{

if (context == &CameraAdjustingExposureContext) {

AVCaptureDevice \*device = (AVCaptureDevice \*)object;

if (!device.isAdjustingExposure && [device isExposureModeSupported:AVCaptureExposureModeLocked]) {

[object removeObserver:self forKeyPath:@"adjustingExposure" context:&CameraAdjustingExposureContext];

dispatch\_async(dispatch\_get\_main\_queue(), ^{

NSError \*error;

if ([device lockForConfiguration:&error]) {

device.exposureMode = AVCaptureExposureModeLocked;

[device unlockForConfiguration];

} else {

NSLog(@"%@", error);

}

});

}

} else {

[super observeValueForKeyPath:keyPath ofObject:object change:change context:context];

}

}

#pragma mark - -自动聚焦、曝光

- (id)resetFocusAndExposure:(AVCaptureDevice \*)device {

AVCaptureFocusMode focusMode = AVCaptureFocusModeContinuousAutoFocus;

AVCaptureExposureMode exposureMode = AVCaptureExposureModeContinuousAutoExposure;

BOOL canResetFocus = [device isFocusPointOfInterestSupported] &&

[device isFocusModeSupported:focusMode];

BOOL canResetExposure = [device isExposurePointOfInterestSupported] &&

[device isExposureModeSupported:exposureMode];

CGPoint centerPoint = CGPointMake(0.5f, 0.5f);

NSError \*error;

if ([device lockForConfiguration:&error]) {

if (canResetFocus) {

device.focusMode = focusMode;

device.focusPointOfInterest = centerPoint;

}

if (canResetExposure) {

device.exposureMode = exposureMode;

device.exposurePointOfInterest = centerPoint;

}

[device unlockForConfiguration];

}

return error;

}

#pragma mark - -闪光灯

- (AVCaptureFlashMode)flashMode:(AVCaptureDevice \*)device{

return [device flashMode];

}

- (id)changeFlash:(AVCaptureDevice \*)device mode:(AVCaptureFlashMode)mode{

if (![device hasFlash]) {

return [self error:@"不支持闪光灯" code:2003];

}

if ([self torchMode:device] == AVCaptureTorchModeOn) {

[self setTorch:device model:AVCaptureTorchModeOff];

}

return [self setFlash:device mode:mode];

}

- (id)setFlash:(AVCaptureDevice \*)device mode:(AVCaptureFlashMode)mode {

if ([device isFlashModeSupported:mode]) {

NSError \*error;

if ([device lockForConfiguration:&error]) {

device.flashMode = mode;

[device unlockForConfiguration];

}

return error;

}

return [self error:@"不支持闪光灯" code:2003];

}

#pragma mark - -手电筒

- (AVCaptureTorchMode)torchMode:(AVCaptureDevice \*)device {

return [device torchMode];

}

- (id)changeTorch:(AVCaptureDevice \*)device model:(AVCaptureTorchMode)mode{

if (![device hasTorch]) {

return [self error:@"不支持手电筒" code:2004];

}

if ([self flashMode:device] == AVCaptureFlashModeOn) {

[self setFlash:device mode:AVCaptureFlashModeOff];

}

return [self setTorch:device model:mode];

}

- (id)setTorch:(AVCaptureDevice \*)device model:(AVCaptureTorchMode)mode {

if ([device isTorchModeSupported:mode]) {

NSError \*error;

if ([device lockForConfiguration:&error]) {

device.torchMode = mode;

[device unlockForConfiguration];

}

return error;

}

return [self error:@"不支持手电筒" code:2004];

}

#pragma mark -

- (NSError \*)error:(NSString \*)text code:(NSInteger)code {

NSDictionary \*desc = @{NSLocalizedDescriptionKey: text};

NSError \*error = [NSError errorWithDomain:@"com.cc.camera" code:code userInfo:desc];

return error;

}

@end

#import "PTBaseViewController.h"

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTCameraViewController : PTBaseViewController

@property (nonatomic, strong) NSDictionary \*fromLanguageDic;

@property (nonatomic, strong) NSDictionary \*toLanguageDic;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTCameraViewController.h"

#import <AssetsLibrary/AssetsLibrary.h>

#import <AVFoundation/AVFoundation.h>

#import <CoreMedia/CMMetadata.h>

#import <Photos/Photos.h>

#import "PTImagePreviewController.h"

#import "PTCameraView.h"

#import "PTCameraManager.h"

#import "PTMotionManager.h"// 关掉陀螺仪，让图片不旋转

#import "PTMovieManager.h"

#import "PTAboutViewController.h"

#import "PTPrivacyViewController.h"

#import "PTSelectLanguageViewController.h"

#import <TZImagePickerController/TZImagePickerController.h>

#define ISIOS9 \_\_IPHONE\_OS\_VERSION\_MAX\_ALLOWED >= \_\_IPHONE\_9\_0

//@interface PTCameraViewController ()<AVCaptureVideoDataOutputSampleBufferDelegate, AVCaptureAudioDataOutputSampleBufferDelegate, PTCameraViewDelegate, TZImagePickerControllerDelegate, GADInterstitialDelegate>

//@interface PTCameraViewController ()<AVCaptureVideoDataOutputSampleBufferDelegate, PTCameraViewDelegate, TZImagePickerControllerDelegate, GADInterstitialDelegate>

@interface PTCameraViewController ()<AVCaptureVideoDataOutputSampleBufferDelegate, PTCameraViewDelegate, TZImagePickerControllerDelegate>

{

// 会话

AVCaptureSession \*\_session;

// 输入

AVCaptureDeviceInput \*\_deviceInput;

// 输出

AVCaptureConnection \*\_videoConnection;

// AVCaptureConnection \*\_audioConnection;

AVCaptureVideoDataOutput \*\_videoOutput;

AVCaptureStillImageOutput \*\_imageOutput;

// 录制

BOOL \_recording;

}

@property(nonatomic, strong) PTCameraView \*cameraView; // 界面布局

@property(nonatomic, strong) PTMovieManager \*movieManager; // 视频管理

@property(nonatomic, strong) PTCameraManager \*cameraManager; // 相机管理

@property(nonatomic, strong) PTMotionManager \*motionManager; // 陀螺仪管理

@property(nonatomic, strong) AVCaptureDevice \*activeCamera; // 当前输入设备

@property(nonatomic, strong) AVCaptureDevice \*inactiveCamera; // 不活跃的设备(这里指前摄像头或后摄像头，不包括外接输入设备)

//@property (nonatomic, strong) GADInterstitial \*interstitial;// 谷歌插页式广告

@end

@implementation PTCameraViewController

- (instancetype)init{

self = [super init];

if (self) {

\_movieManager = [[PTMovieManager alloc] init];

\_motionManager = [[PTMotionManager alloc] init];

\_cameraManager = [[PTCameraManager alloc] init];

}

return self;

}

//- (GADInterstitial \*)createAndLoadInterstitial {

// GADInterstitial \*interstitial =

// [[GADInterstitial alloc] initWithAdUnitID:@"ca-app-pub-9104226499574771/5333647844"];

// interstitial.delegate = self;

// [interstitial loadRequest:[GADRequest request]];

// return interstitial;

//}

- (void)viewDidLoad{

[super viewDidLoad];

// self.interstitial = [self createAndLoadInterstitial];

self.cameraView = [[PTCameraView alloc] initWithFrame:self.view.bounds];

self.cameraView.delegate = self;

[self.view addSubview:self.cameraView];

NSError \*error;

[self setupSession:&error];

if (!error) {

[self.cameraView.previewView setCaptureSessionsion:\_session];

[self startCaptureSession];

}else{

[self.view showError:error];

}

// 默认初始化源语言为自动检测,目标语言为系统语言(如果系统语言没有则为英语)

self.fromLanguageDic = @{@"name":@"Auto",@"code":@"auto"};

self.toLanguageDic = @{@"name":@"English",@"code":@"en"};

// 检测手机语言，设置右侧目标语言为手机语言，如无则默认英语

// 查看当前手机的语言环境

NSString \*languageCode = [NSLocale preferredLanguages][0];// 返回的也是国际通用语言Code+国际通用国家地区代码

NSString \*countryCode = [NSString stringWithFormat:@"-%@", [[NSLocale currentLocale] objectForKey:NSLocaleCountryCode]];

if (languageCode) {

languageCode = [languageCode stringByReplacingOccurrencesOfString:countryCode withString:@""];

}

PTLog(@"系统当前语言:%@", languageCode);

NSArray \*dataArray = @[

@[@{@"name":@"Afrikaans",@"code":@"af"},@{@"name":@"Albanian",@"code":@"sq"},@{@"name":@"Amharic",@"code":@"am"},@{@"name":@"Arabic",@"code":@"ar"},@{@"name":@"Armenian",@"code":@"hy"},@{@"name":@"Azerbaijani",@"code":@"az"}],

@[@{@"name":@"Basque",@"code":@"eu"},@{@"name":@"Belarusian",@"code":@"be"},@{@"name":@"Bengali",@"code":@"bn"},@{@"name":@"Bosnian",@"code":@"bs"},@{@"name":@"Bulgarian",@"code":@"bg"}],

@[@{@"name":@"Catalan",@"code":@"ca"},@{@"name":@"Cebuano",@"code":@"ceb"},@{@"name":@"Chichewa",@"code":@"ny"},@{@"name":@"Chinese (Simplified)",@"code":@"zh-CN"},@{@"name":@"Chinese (Traditional)",@"code":@"zh-TW"},@{@"name":@"Corsican",@"code":@"co"},@{@"name":@"Croatian",@"code":@"hr"},@{@"name":@"Czech",@"code":@"cs"}],

@[@{@"name":@"Danish",@"code":@"da"},@{@"name":@"Dutch",@"code":@"nl"}],

@[@{@"name":@"English",@"code":@"en"},@{@"name":@"Esperanto",@"code":@"eo"},@{@"name":@"Estonian",@"code":@"et"}],

@[@{@"name":@"Filipino",@"code":@"tl"},@{@"name":@"Finnish",@"code":@"fi"},@{@"name":@"French",@"code":@"fr"},@{@"name":@"Frisian",@"code":@"fy"}],

@[@{@"name":@"Galician",@"code":@"gl"},@{@"name":@"Georgian",@"code":@"ka"},@{@"name":@"German",@"code":@"de"},@{@"name":@"Greek",@"code":@"el"},@{@"name":@"Gujarati",@"code":@"gu"}],

@[@{@"name":@"Haitian Creole",@"code":@"ht"},@{@"name":@"Hausa",@"code":@"ha"},@{@"name":@"Hawaiian",@"code":@"haw"},@{@"name":@"Hebrew",@"code":@"iw"},@{@"name":@"Hindi",@"code":@"hi"},@{@"name":@"Hmong",@"code":@"hmn"},@{@"name":@"Hungarian",@"code":@"hu"}],

@[@{@"name":@"Icelandic",@"code":@"is"},@{@"name":@"Igbo",@"code":@"ig"},@{@"name":@"Indonesian",@"code":@"id"},@{@"name":@"Irish",@"code":@"ga"},@{@"name":@"Italian",@"code":@"it"}],

@[@{@"name":@"Japanese",@"code":@"ja"},@{@"name":@"Javanese",@"code":@"jw"}],

@[@{@"name":@"Kannada",@"code":@"kn"},@{@"name":@"Kazakh",@"code":@"kk"},@{@"name":@"Khmer",@"code":@"km"},@{@"name":@"Kinyarwanda",@"code":@"rw"},@{@"name":@"Korean",@"code":@"ko"},@{@"name":@"Kurdish (Kurmanji)",@"code":@"ku"},@{@"name":@"Kyrgyz",@"code":@"ky"}],

@[@{@"name":@"Lao",@"code":@"lo"},@{@"name":@"Latin",@"code":@"la"},@{@"name":@"Latvian",@"code":@"lv"},@{@"name":@"Lithuanian",@"code":@"lt"},@{@"name":@"Luxembourgish",@"code":@"lb"}],

@[@{@"name":@"Macedonian",@"code":@"mk"},@{@"name":@"Malagasy",@"code":@"mg"},@{@"name":@"Malay",@"code":@"ms"},@{@"name":@"Malayalam",@"code":@"ml"},@{@"name":@"Maltese",@"code":@"mt"},@{@"name":@"Maori",@"code":@"mi"},@{@"name":@"Marathi",@"code":@"mr"},@{@"name":@"Mongolian",@"code":@"mn"},@{@"name":@"Myanmar (Burmese)",@"code":@"my"}],

@[@{@"name":@"Nepali",@"code":@"ne"},@{@"name":@"Norwegian",@"code":@"no"}],

@[@{@"name":@"Odia (Oriya)",@"code":@"or"}],

@[@{@"name":@"Pashto",@"code":@"ps"},@{@"name":@"Persian",@"code":@"fa"},@{@"name":@"Polish",@"code":@"pl"},@{@"name":@"Portuguese",@"code":@"pt"},@{@"name":@"Punjabi",@"code":@"pa"}],

@[@{@"name":@"Romanian",@"code":@"ro"},@{@"name":@"Russian",@"code":@"ru"}],

@[@{@"name":@"Samoan",@"code":@"sm"},@{@"name":@"Scots Gaelic",@"code":@"gd"},@{@"name":@"Serbian",@"code":@"sr"},@{@"name":@"Sesotho",@"code":@"st"},@{@"name":@"Shona",@"code":@"sn"},@{@"name":@"Sindhi",@"code":@"sd"},@{@"name":@"Sinhala",@"code":@"si"},@{@"name":@"Slovak",@"code":@"sk"},@{@"name":@"Slovenian",@"code":@"sl"},@{@"name":@"Somali",@"code":@"so"},@{@"name":@"Spanish",@"code":@"es"},@{@"name":@"Sundanese",@"code":@"su"},@{@"name":@"Swahili",@"code":@"sw"},@{@"name":@"Swedish",@"code":@"sv"}],

@[@{@"name":@"Tajik",@"code":@"tg"},@{@"name":@"Tamil",@"code":@"ta"},@{@"name":@"Tatar",@"code":@"tt"},@{@"name":@"Telugu",@"code":@"te"},@{@"name":@"Thai",@"code":@"th"},@{@"name":@"Turkish",@"code":@"tr"},@{@"name":@"Turkmen",@"code":@"tk"}],

@[@{@"name":@"Ukrainian",@"code":@"uk"},@{@"name":@"Urdu",@"code":@"ur"},@{@"name":@"Uyghur",@"code":@"ug"},@{@"name":@"Uzbek",@"code":@"uz"}],

@[@{@"name":@"Vietnamese",@"code":@"vi"}],

@[@{@"name":@"Welsh",@"code":@"cy"}],

@[@{@"name":@"Xhosa",@"code":@"xh"}],

@[@{@"name":@"Yiddish",@"code":@"yi"},@{@"name":@"Yoruba",@"code":@"yo"}],

@[@{@"name":@"Zulu",@"code":@"zu"}],

];

for (NSArray \*array in dataArray) {

for (NSDictionary \*dic in array) {

if ([languageCode isEqualToString:@"zh-Hans"]) {

if ([[dic objectForKey:@"code"] isEqualToString:@"zh-CN"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"zh-Hant"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"zh-TW"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"en"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"en"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"af"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"af"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"sq"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"sq"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"am"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"am"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ar-ER"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ar"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"az"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"az"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"bn"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"bn"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"bs"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"bs"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"bg"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"bg"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"hr-HR"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"hr"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"cs-CZ"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"cs"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"da-DK"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"da"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"nl-AW"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"nl"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"et"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"et"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"fi-FI"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"fi"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"fr-DZ"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"fr"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ka"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ka"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"de-AT"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"de"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"el-CY"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"el"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ha"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ha"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"he-IL"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"iw"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"hi-IN"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"hi"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"hu-HU"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"hu"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"id-ID"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"id"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"it-IT"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"it"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ja-JP"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ja"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ko-KP"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ko"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"lv"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"lv"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ms-Arab"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ms"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"nb-NO"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"no"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"fa"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"fa"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ps"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ps"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"pl-PL"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"pl"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"pt"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"pt"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ro-MD"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ro"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ru-BY"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ru"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"sr"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"sr"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"sk-SK"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"sk"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"sl"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"sl"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"so"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"so"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"es-AI"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"es"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"sw"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"sw"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"sv-AX"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"sv"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"fil"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"tl"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ta"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ta"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"th-TH"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"th"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"tr-CY"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"tr"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"uk-UA"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"uk"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"ur"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"ur"]) {

self.toLanguageDic = dic;

}

}else if ([languageCode isEqualToString:@"vi-VN"]){

if ([[dic objectForKey:@"code"] isEqualToString:@"vi"]) {

self.toLanguageDic = dic;

}

}

}

}

}

- (void)viewWillAppear:(BOOL)animated{

[super viewWillAppear:animated];

[self.navigationController setNavigationBarHidden:YES animated:YES];

// 检测是否同意隐私政策

[self privacyView];

}

- (void)viewWillDisappear:(BOOL)animated{

[super viewWillDisappear:animated];

[self.navigationController setNavigationBarHidden:NO animated:YES];

}

- (void)dealloc{

NSLog(@"相机界面销毁了");

}

#pragma mark - -输入设备

- (AVCaptureDevice \*)cameraWithPosition:(AVCaptureDevicePosition)position{

NSArray \*devices = [AVCaptureDevice devicesWithMediaType:AVMediaTypeVideo];

for (AVCaptureDevice \*device in devices) {

if (device.position == position) {

return device;

}

}

return nil;

}

- (AVCaptureDevice \*)activeCamera{

return \_deviceInput.device;

}

- (AVCaptureDevice \*)inactiveCamera{

AVCaptureDevice \*device = nil;

if ([[AVCaptureDevice devicesWithMediaType:AVMediaTypeVideo] count] > 1) {

if ([self activeCamera].position == AVCaptureDevicePositionBack) {

device = [self cameraWithPosition:AVCaptureDevicePositionFront];

} else {

device = [self cameraWithPosition:AVCaptureDevicePositionBack];

}

}

return device;

}

#pragma mark - -相关配置

/// 会话

- (void)setupSession:(NSError \*\*)error{

\_session = [[AVCaptureSession alloc]init];

\_session.sessionPreset = AVCaptureSessionPresetHigh;

[self setupSessionInputs:error];

[self setupSessionOutputs:error];

}

/// 输入

- (void)setupSessionInputs:(NSError \*\*)error{

// 视频输入

AVCaptureDevice \*videoDevice = [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeVideo];

AVCaptureDeviceInput \*videoInput = [AVCaptureDeviceInput deviceInputWithDevice:videoDevice error:error];

if (videoInput) {

if ([\_session canAddInput:videoInput]){

[\_session addInput:videoInput];

}

}

\_deviceInput = videoInput;

// // 音频输入

// AVCaptureDevice \*audioDevice = [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeAudio];

// AVCaptureDeviceInput \*audioIn = [[AVCaptureDeviceInput alloc] initWithDevice:audioDevice error:error];

// if ([\_session canAddInput:audioIn]){

// [\_session addInput:audioIn];

// }

}

/// 输出

- (void)setupSessionOutputs:(NSError \*\*)error{

dispatch\_queue\_t captureQueue = dispatch\_queue\_create("com.cc.captureQueue", DISPATCH\_QUEUE\_SERIAL);

// 视频输出

AVCaptureVideoDataOutput \*videoOut = [[AVCaptureVideoDataOutput alloc] init];

[videoOut setAlwaysDiscardsLateVideoFrames:YES];

[videoOut setVideoSettings:@{(id)kCVPixelBufferPixelFormatTypeKey: [NSNumber numberWithInt:kCVPixelFormatType\_32BGRA]}];

[videoOut setSampleBufferDelegate:self queue:captureQueue];

if ([\_session canAddOutput:videoOut]){

[\_session addOutput:videoOut];

}

\_videoOutput = videoOut;

\_videoConnection = [videoOut connectionWithMediaType:AVMediaTypeVideo];

// // 音频输出

// AVCaptureAudioDataOutput \*audioOut = [[AVCaptureAudioDataOutput alloc] init];

// [audioOut setSampleBufferDelegate:self queue:captureQueue];

// if ([\_session canAddOutput:audioOut]){

// [\_session addOutput:audioOut];

// }

// \_audioConnection = [audioOut connectionWithMediaType:AVMediaTypeAudio];

// 静态图片输出

AVCaptureStillImageOutput \*imageOutput = [[AVCaptureStillImageOutput alloc] init];

imageOutput.outputSettings = @{AVVideoCodecKey:AVVideoCodecJPEG};

if ([\_session canAddOutput:imageOutput]) {

[\_session addOutput:imageOutput];

}

\_imageOutput = imageOutput;

}

#pragma mark - -会话控制

// 开启捕捉

- (void)startCaptureSession{

if (!\_session.isRunning){

[\_session startRunning];

}

}

// 停止捕捉

- (void)stopCaptureSession{

if (\_session.isRunning){

[\_session stopRunning];

}

}

#pragma mark - -操作相机

// 缩放

-(void)zoomAction:(PTCameraView \*)cameraView factor:(CGFloat)factor {

NSError \*error = [\_cameraManager zoom:[self activeCamera] factor:factor];

if (error) NSLog(@"%@", error);

}

// 聚焦

-(void)focusAction:(PTCameraView \*)cameraView point:(CGPoint)point handle:(void (^)(NSError \*))handle {

NSError \*error = [\_cameraManager focus:[self activeCamera] point:point];

handle(error);

NSLog(@"%f", [self activeCamera].activeFormat.videoMaxZoomFactor);

}

// 曝光

-(void)exposAction:(PTCameraView \*)cameraView point:(CGPoint)point handle:(void (^)(NSError \*))handle {

NSError \*error = [\_cameraManager expose:[self activeCamera] point:point];

handle(error);

}

// 自动聚焦、曝光

-(void)autoFocusAndExposureAction:(PTCameraView \*)cameraView handle:(void (^)(NSError \*))handle {

NSError \*error = [\_cameraManager resetFocusAndExposure:[self activeCamera]];

handle(error);

}

// 闪光灯

-(void)flashLightAction:(PTCameraView \*)cameraView handle:(void (^)(NSError \*))handle {

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_FlashLightButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

BOOL on = [\_cameraManager flashMode:[self activeCamera]] == AVCaptureFlashModeOn;

AVCaptureFlashMode mode = on ? AVCaptureFlashModeOff : AVCaptureFlashModeOn;

NSError \*error = [\_cameraManager changeFlash:[self activeCamera] mode: mode];

handle(error);

}

// 手电筒

-(void)torchLightAction:(PTCameraView \*)cameraView handle:(void (^)(NSError \*))handle {

BOOL on = [\_cameraManager torchMode:[self activeCamera]] == AVCaptureTorchModeOn;

AVCaptureTorchMode mode = on ? AVCaptureTorchModeOff : AVCaptureTorchModeOn;

NSError \*error = [\_cameraManager changeTorch:[self activeCamera] model:mode];

handle(error);

}

// 转换摄像头

- (void)swicthCameraAction:(PTCameraView \*)cameraView handle:(void (^)(NSError \*))handle {

NSError \*error;

AVCaptureDevice \*videoDevice = [self inactiveCamera];

AVCaptureDeviceInput \*videoInput = [AVCaptureDeviceInput deviceInputWithDevice:videoDevice error:&error];

if (videoInput) {

// 动画效果

CATransition \*animation = [CATransition animation];

animation.type = @"oglFlip";

animation.subtype = kCATransitionFromLeft;

animation.duration = 0.5;

[self.cameraView.previewView.layer addAnimation:animation forKey:@"flip"];

// 当前闪光灯状态

AVCaptureFlashMode mode = [\_cameraManager flashMode:[self activeCamera]];

// 转换摄像头

\_deviceInput = [\_cameraManager switchCamera:\_session old:\_deviceInput new:videoInput];

// 重新设置视频输出链接

\_videoConnection = [\_videoOutput connectionWithMediaType:AVMediaTypeVideo];

// 如果后置转前置，系统会自动关闭手电筒(如果之前打开的，需要更新UI)

if (videoDevice.position == AVCaptureDevicePositionFront) {

[self.cameraView changeTorch:NO];

}

// 前后摄像头的闪光灯不是同步的，所以在转换摄像头后需要重新设置闪光灯

[\_cameraManager changeFlash:[self activeCamera] mode:mode];

}

handle(error);

}

#pragma mark - -拍摄照片

// 拍照

- (void)takePhotoAction:(PTCameraView \*)cameraView{

AVCaptureConnection \*connection = [\_imageOutput connectionWithMediaType:AVMediaTypeVideo];

if (connection.isVideoOrientationSupported) {

connection.videoOrientation = [self currentVideoOrientation];// 关掉陀螺仪

}

[\_imageOutput captureStillImageAsynchronouslyFromConnection:connection completionHandler:^(CMSampleBufferRef \_Nullable imageDataSampleBuffer, NSError \* \_Nullable error) {

if (error) {

[self.view showError:error];

return;

}

NSData \*imageData = [AVCaptureStillImageOutput jpegStillImageNSDataRepresentation:imageDataSampleBuffer];

UIImage \*image = [[UIImage alloc]initWithData:imageData];

PTImagePreviewController \*vc = [[PTImagePreviewController alloc] init];

vc.image = image;

vc.rect = self.cameraView.previewView.frame;

vc.fromLanguageDic = self.fromLanguageDic;

vc.toLanguageDic = self.toLanguageDic;

[self.navigationController pushViewController:vc animated:NO];

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_TakePhotoButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

// // 展示广告

// if (self.interstitial.isReady) {

// [self.interstitial presentFromRootViewController:self];

// }else{// 广告尚未准备好

// NSLog(@"Ad wasn't ready");

// }

}];

}

// 取消拍照

- (void)cancelAction:(PTCameraView \*)cameraView{

[self.navigationController popViewControllerAnimated:YES];

}

#pragma mark - -录制视频

// 开始录像

-(void)startRecordVideoAction:(PTCameraView \*)cameraView{

\_recording = YES;

\_movieManager.currentDevice = [self activeCamera];

// \_movieManager.currentOrientation = [self currentVideoOrientation];// 关掉陀螺仪

[\_movieManager start:^(NSError \* \_Nonnull error) {

if (error) [self.view showError:error];

}];

}

// 停止录像

-(void)stopRecordVideoAction:(PTCameraView \*)cameraView{

\_recording = NO;

[\_movieManager stop:^(NSURL \* \_Nonnull url, NSError \* \_Nonnull error) {

if (error) {

[self.view showError:error];

} else {

[self.view showAlertView:@"是否保存到相册" ok:^(UIAlertAction \*act) {

[self saveMovieToCameraRoll: url];

} cancel:nil];

}

}];

}

// 保存视频

- (void)saveMovieToCameraRoll:(NSURL \*)url{

[self.view showLoadHUD:@"保存中..."];

if (ISIOS9) {

[PHPhotoLibrary requestAuthorization:^( PHAuthorizationStatus status ) {

if (status != PHAuthorizationStatusAuthorized) return;

[[PHPhotoLibrary sharedPhotoLibrary] performChanges:^{

PHAssetCreationRequest \*videoRequest = [PHAssetCreationRequest creationRequestForAsset];

[videoRequest addResourceWithType:PHAssetResourceTypeVideo fileURL:url options:nil];

} completionHandler:^( BOOL success, NSError \* \_Nullable error ) {

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[self.view hideHUD];

});

success?:[self.view showError:error];

}];

}];

} else {

ALAssetsLibrary \*lab = [[ALAssetsLibrary alloc]init];

[lab writeVideoAtPathToSavedPhotosAlbum:url completionBlock:^(NSURL \*assetURL, NSError \*error) {

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[self.view hideHUD];

});

!error?:[self.view showError:error];

}];

}

}

#pragma mark - 相机上面翻译按钮代理

// 左边翻译源语言

- (void)leftLanguageAction{

NSLog(@"左边源语言222");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_LeftLanguageButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

PTSelectLanguageViewController \*vc = [[PTSelectLanguageViewController alloc] init];

vc.isShowAutoLanguage = YES;

vc.selectLanguageBlock = ^(NSDictionary \* \_Nonnull languageDic) {

NSLog(@"选择了--> %@", languageDic);

self.fromLanguageDic = languageDic;

};

PTBaseNavigationController \*nav = [[PTBaseNavigationController alloc] initWithRootViewController:vc];

self.definesPresentationContext = YES;

nav.modalPresentationStyle = UIModalPresentationOverFullScreen;

[self presentViewController:nav animated:YES completion:^{

}];

}

// 右边翻译目标语言

- (void)rightLanguageAction{

NSLog(@"右边翻译目标语言222");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_RightLanguageButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

PTSelectLanguageViewController \*vc = [[PTSelectLanguageViewController alloc] init];

vc.isShowAutoLanguage = NO;

vc.selectLanguageBlock = ^(NSDictionary \* \_Nonnull languageDic) {

NSLog(@"选择了--> %@", languageDic);

self.toLanguageDic = languageDic;

};

PTBaseNavigationController \*nav = [[PTBaseNavigationController alloc] initWithRootViewController:vc];

self.definesPresentationContext = YES;

nav.modalPresentationStyle = UIModalPresentationOverFullScreen;

[self presentViewController:nav animated:YES completion:^{

}];

}

// 中间切换button

- (void)convertAction{

NSLog(@"切换语言222");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_ConvertButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

// 如果左边是自动检测，不让切换

if ([[self.fromLanguageDic objectForKey:@"name"] isEqualToString:@"Auto"]) {

return;

}

NSDictionary \*dic = self.fromLanguageDic;

self.fromLanguageDic = self.toLanguageDic;

self.toLanguageDic = dic;

}

// 右侧三个点更多

- (void)moreAction{

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_MoreButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

NSLog(@"右侧三个点更多222");

PTAboutViewController \*vc = [[PTAboutViewController alloc] init];

[self.navigationController pushViewController:vc animated:YES];

}

// 相册

- (void)photosAction{

NSLog(@"相册222");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_PhotosButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

TZImagePickerController \*imagePickerVc = [[TZImagePickerController alloc] initWithMaxImagesCount:1 delegate:self];

/// 默认为YES，如果设置为NO,用户将不能选择视频

imagePickerVc.allowPickingVideo = NO;

/// 默认为YES，如果设置为NO,预览按钮将隐藏,用户将不能去预览照片

imagePickerVc.allowPreview = NO;

// You can get the photos by block, the same as by delegate.

// 你可以通过block或者代理，来得到用户选择的照片.

[imagePickerVc setDidFinishPickingPhotosHandle:^(NSArray<UIImage \*> \*photos, NSArray \*assets, BOOL isSelectOriginalPhoto) {

UIImage \*image = [photos firstObject];

PTImagePreviewController \*vc = [[PTImagePreviewController alloc] init];

vc.image = image;

vc.rect = self.cameraView.previewView.frame;

vc.fromLanguageDic = self.fromLanguageDic;

vc.toLanguageDic = self.toLanguageDic;

[self.navigationController pushViewController:vc animated:NO];

// // 展示广告

// if (self.interstitial.isReady) {

// [self.interstitial presentFromRootViewController:self];

// }else{// 广告尚未准备好

// NSLog(@"Ad wasn't ready");

// }

}];

[self presentViewController:imagePickerVc animated:YES completion:nil];

}

#pragma mark - -输出代理

//- (void)captureOutput:(AVCaptureOutput \*)captureOutput didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer fromConnection:(AVCaptureConnection \*)connection{

// if (\_recording) {

// [\_movieManager writeData:connection video:\_videoConnection audio:\_audioConnection buffer:sampleBuffer];

// }

//}

#pragma mark - -其它方法

// 当前设备取向

//- (AVCaptureVideoOrientation)currentVideoOrientation{

// AVCaptureVideoOrientation orientation;

// switch (self.motionManager.deviceOrientation) {

// case UIDeviceOrientationPortrait:

// orientation = AVCaptureVideoOrientationPortrait;

// break;

// case UIDeviceOrientationLandscapeLeft:

// orientation = AVCaptureVideoOrientationLandscapeRight;

// break;

// case UIDeviceOrientationLandscapeRight:

// orientation = AVCaptureVideoOrientationLandscapeLeft;

// break;

// case UIDeviceOrientationPortraitUpsideDown:

// orientation = AVCaptureVideoOrientationPortraitUpsideDown;

// break;

// default:

// orientation = AVCaptureVideoOrientationPortrait;

// break;

// }

// return orientation;

//}

// 都不改变图片的方向,所拍即所得

- (AVCaptureVideoOrientation)currentVideoOrientation{

AVCaptureVideoOrientation orientation;

switch (self.motionManager.deviceOrientation) {

case UIDeviceOrientationPortrait:

orientation = AVCaptureVideoOrientationPortrait;

break;

case UIDeviceOrientationLandscapeLeft:

orientation = AVCaptureVideoOrientationPortrait;

break;

case UIDeviceOrientationLandscapeRight:

orientation = AVCaptureVideoOrientationPortrait;

break;

case UIDeviceOrientationPortraitUpsideDown:

orientation = AVCaptureVideoOrientationPortrait;

break;

default:

orientation = AVCaptureVideoOrientationPortrait;

break;

}

return orientation;

}

- (void)didReceiveMemoryWarning {

[super didReceiveMemoryWarning];

// Dispose of any resources that can be recreated.

}

#pragma mark - 隐私政策

- (void)privacyView{

// 判断是否查看过隐私政策

NSUserDefaults \*userDefault =[NSUserDefaults standardUserDefaults];

if ([userDefault objectForKey:@"Privacy"]) {

// 已查看

}else{

// 未查看

// 弹出弹窗,如同意协议,set值

PTPrivacyViewController \*vc = [[PTPrivacyViewController alloc] init];

//同意隐私协议

WeakSelf(self);

vc.acceptButtonBlock = ^{

[weakSelf dismissViewControllerAnimated:NO completion:^{

}];

[userDefault setObject:@"YES"forKey:@"Privacy"];

[userDefault synchronize];

};

self.definesPresentationContext = YES;

vc.modalPresentationStyle = UIModalPresentationOverFullScreen;

[self presentViewController:vc animated:NO completion:^{

}];

}

}

#pragma mark - SetGet

- (void)setFromLanguageDic:(NSDictionary \*)fromLanguageDic{

\_fromLanguageDic = fromLanguageDic;

self.cameraView.fromLanguageButtonStr = [fromLanguageDic objectForKey:@"name"];

}

- (void)setToLanguageDic:(NSDictionary \*)toLanguageDic{

\_toLanguageDic = toLanguageDic;

self.cameraView.toLanguageButtonStr = [toLanguageDic objectForKey:@"name"];

}

@end

#import "PTBaseViewController.h"

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTImagePreviewController : PTBaseViewController

@property (nonatomic, strong) UIImage \*image;

@property (nonatomic, readwrite) CGRect rect;

@property (nonatomic, strong) NSDictionary \*fromLanguageDic;

@property (nonatomic, strong) NSDictionary \*toLanguageDic;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTImagePreviewController.h"

#import "PTTextComparedViewController.h"

#import <Photos/Photos.h>

#import "PTImageTool.h"

#import "PTSelectLanguageViewController.h"

#import <CommonCrypto/CommonDigest.h>

@interface PTImagePreviewController ()

@property (nonatomic, strong) UIButton \*convertButton;

@property (nonatomic, strong) UIButton \*leftLanguageButton;

@property (nonatomic, strong) UIButton \*rightLanguageButton;

@property (nonatomic, strong) UIImageView \*rightArrowImageView;

@property (nonatomic, strong) UIImageView \*imageView;// 内容imageview

@property (nonatomic, strong) UIView \*hoodView;// 翻译成功之后的图片灰色遮罩

@property (nonatomic, strong) NSString \*detectedTextString;

@property (nonatomic, strong) NSString \*translatedTextString;

@property (nonatomic, assign) int postCount;// 请求次数，最多5次

@end

@implementation PTImagePreviewController

- (void)viewWillAppear:(BOOL)animated{

[super viewWillAppear:animated];

[self.navigationController setNavigationBarHidden:YES animated:YES];

}

- (void)viewWillDisappear:(BOOL)animated{

[super viewWillDisappear:animated];

[self.navigationController setNavigationBarHidden:NO animated:YES];

}

- (void)viewDidLoad {

[super viewDidLoad];

// 上面的view

UIView \*topView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, SCREEN\_WIDTH, StatusbarAndNavigationbarHeight)];

topView.backgroundColor = [UIColor blackColor];

[self.view addSubview:topView];

// 返回

UIButton \*backButton = [[UIButton alloc] initWithFrame:CGRectMake(18, StatusbarAndNavigationbarHeight -4 -36, 36, 36)];

[backButton addTarget:self action:@selector(backButtonAction:) forControlEvents:UIControlEventTouchUpInside];

[backButton setImage:[UIImage imageNamed:@"back"] forState:UIControlStateNormal];

[self.view addSubview:backButton];

// 中间转换button

self.convertButton = [[UIButton alloc] initWithFrame:CGRectMake((topView.width - 56)/2, topView.height -4 -36, 56, 36)];

[self.convertButton setImage:[UIImage imageNamed:@"convert"] forState:UIControlStateNormal];

[self.convertButton addTarget:self action:@selector(convertButtonAction:) forControlEvents:UIControlEventTouchUpInside];

[topView addSubview:self.convertButton];

// 左边源语言button

self.leftLanguageButton = [[UIButton alloc] init];

[self.leftLanguageButton setTitle:[self.fromLanguageDic objectForKey:@"name"] forState:UIControlStateNormal];

[self.leftLanguageButton setContentHorizontalAlignment:UIControlContentHorizontalAlignmentRight];

self.leftLanguageButton.titleLabel.lineBreakMode = NSLineBreakByTruncatingTail;//省略号靠右侧

[self.leftLanguageButton setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];

self.leftLanguageButton.titleLabel.font = [UIFont systemFontOfSize:16];

[self.leftLanguageButton addTarget:self action:@selector(leftLanguageButtonAction:) forControlEvents:UIControlEventTouchUpInside];

// 根据button文字内容计算宽度

CGSize buttonSize = [self.leftLanguageButton.titleLabel.text boundingRectWithSize:CGSizeMake(MAXFLOAT, 36)

options:NSStringDrawingTruncatesLastVisibleLine | NSStringDrawingUsesLineFragmentOrigin | NSStringDrawingUsesFontLeading

attributes:@{ NSFontAttributeName:self.leftLanguageButton.titleLabel.font}

context:nil].size;

self.leftLanguageButton.frame = CGRectMake(self.convertButton.frame.origin.x -16 -(buttonSize.width>90?90:buttonSize.width), self.convertButton.frame.origin.y, buttonSize.width>90?90:buttonSize.width, 36);

[topView addSubview:self.leftLanguageButton];

UIImageView \*leftArrowImageView = [[UIImageView alloc] initWithFrame:CGRectMake(self.leftLanguageButton.right + 4, self.leftLanguageButton.top + 14, 8, 8)];

leftArrowImageView.image = [UIImage imageNamed:@"down\_arrow"];

[topView addSubview:leftArrowImageView];

// 右边目标语言button

self.rightLanguageButton = [[UIButton alloc] init];

[self.rightLanguageButton setTitle:[self.toLanguageDic objectForKey:@"name"] forState:UIControlStateNormal];

[self.rightLanguageButton setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];

self.rightLanguageButton.titleLabel.lineBreakMode = NSLineBreakByTruncatingTail;//省略号靠右侧

[self.rightLanguageButton setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];

self.rightLanguageButton.titleLabel.font = [UIFont systemFontOfSize:16];

[self.rightLanguageButton addTarget:self action:@selector(rightLanguageButtonAction:) forControlEvents:UIControlEventTouchUpInside];

// 根据button文字内容计算宽度

CGSize buttonSize1 = [self.rightLanguageButton.titleLabel.text boundingRectWithSize:CGSizeMake(MAXFLOAT, 36)

options:NSStringDrawingTruncatesLastVisibleLine | NSStringDrawingUsesLineFragmentOrigin | NSStringDrawingUsesFontLeading

attributes:@{ NSFontAttributeName:self.rightLanguageButton.titleLabel.font}

context:nil].size;

self.rightLanguageButton.frame = CGRectMake(self.convertButton.right +4, self.convertButton.frame.origin.y, buttonSize1.width>90?90:buttonSize1.width, 36);

[topView addSubview:self.rightLanguageButton];

self.rightArrowImageView = [[UIImageView alloc] initWithFrame:CGRectMake(self.rightLanguageButton.right + 4, self.rightLanguageButton.top + 14, 8, 8)];

self.rightArrowImageView.image = [UIImage imageNamed:@"down\_arrow"];

[topView addSubview:self.rightArrowImageView];

// 内容image

self.imageView = [[UIImageView alloc]initWithImage:self.image];

self.imageView.layer.masksToBounds = YES;

self.imageView.contentMode = UIViewContentModeScaleAspectFill;

self.imageView.frame = CGRectMake(0, StatusbarAndNavigationbarHeight, self.rect.size.width, self.rect.size.height);

self.imageView.userInteractionEnabled = YES;

[self.view addSubview:self.imageView];

//单击

UITapGestureRecognizer \*clearHoodViewTap = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector(clearHoodViewTapAction:)];

[self.imageView addGestureRecognizer:clearHoodViewTap];

// 下面的view

UIView \*bottomView = [[UIView alloc] init];

if (is\_iPhoneX) {

bottomView.frame = CGRectMake(0, self.imageView.bottom, SCREEN\_WIDTH, 49+34);

}else{

bottomView.frame = CGRectMake(0, self.imageView.bottom, SCREEN\_WIDTH, 64);

}

bottomView.backgroundColor = [UIColor blackColor];

[self.view addSubview:bottomView];

// 文本对照

UIButton \*textButton = [[UIButton alloc] initWithFrame:CGRectMake(0, 0, SCREEN\_WIDTH/2, is\_iPhoneX?49:64)];

[textButton setTitle:NSLocalizedString(@"Contrast", nil) forState:UIControlStateNormal];

[textButton setImage:[UIImage imageNamed:@"text"] forState:UIControlStateNormal];

[textButton addTarget:self action:@selector(textButtonAction:) forControlEvents:UIControlEventTouchUpInside];

[bottomView addSubview:textButton];

textButton.titleEdgeInsets = UIEdgeInsetsMake(0.0, 0.0, 0.0, 60);

textButton.imageEdgeInsets = UIEdgeInsetsMake(0.0, 0.0, 0.0, 60);

// 导出图片

UIButton \*exportButton = [[UIButton alloc] initWithFrame:CGRectMake(SCREEN\_WIDTH/2, 0, SCREEN\_WIDTH/2, is\_iPhoneX?49:64)];

[exportButton setTitle:NSLocalizedString(@"Export", nil) forState:UIControlStateNormal];

[exportButton setImage:[UIImage imageNamed:@"export"] forState:UIControlStateNormal];

[exportButton addTarget:self action:@selector(exportButtonAction:) forControlEvents:UIControlEventTouchUpInside];

[bottomView addSubview:exportButton];

exportButton.titleEdgeInsets = UIEdgeInsetsMake(0.0, 60, 0.0, 0.0);

exportButton.imageEdgeInsets = UIEdgeInsetsMake(0.0, 60, 0.0, 0.0);

// 拿到图片后请求接口,调上传图片和翻译接口...

[self uploadImage:self.image];

}

- (void)didReceiveMemoryWarning {

[super didReceiveMemoryWarning];

// Dispose of any resources that can be recreated.

}

#pragma mark - Action

- (void)backButtonAction:(id)sender{

[self.navigationController popViewControllerAnimated:NO];

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_back" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

}

// 切换翻译源、目标语言

- (void)convertButtonAction:(id)sender{

NSLog(@"切换语言");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_ConvertButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

// 如果左边是自动检测，不让切换

if ([[self.fromLanguageDic objectForKey:@"name"] isEqualToString:@"Auto"]) {

return;

}

NSDictionary \*dic = self.fromLanguageDic;

self.fromLanguageDic = self.toLanguageDic;

self.toLanguageDic = dic;

// 修改了源语言,重新调上传图片和翻译接口...

[self uploadImage:self.image];

}

// 左边源语言

- (void)leftLanguageButtonAction:(id)sender{

NSLog(@"左边源语言");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_LeftLanguageButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

PTSelectLanguageViewController \*vc = [[PTSelectLanguageViewController alloc] init];

vc.isShowAutoLanguage = YES;

WeakSelf(self);

vc.selectLanguageBlock = ^(NSDictionary \* \_Nonnull languageDic) {

NSLog(@"选择了--> %@", languageDic);

weakSelf.fromLanguageDic = languageDic;

// 修改了源语言,重新调上传图片和翻译接口...

[self uploadImage:self.image];

};

PTBaseNavigationController \*nav = [[PTBaseNavigationController alloc] initWithRootViewController:vc];

self.definesPresentationContext = YES;

nav.modalPresentationStyle = UIModalPresentationOverFullScreen;

[self presentViewController:nav animated:YES completion:^{

}];

}

// 右边目标语言

- (void)rightLanguageButtonAction:(id)sender{

NSLog(@"右边目标语言");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_RightLanguageButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

PTSelectLanguageViewController \*vc = [[PTSelectLanguageViewController alloc] init];

vc.isShowAutoLanguage = NO;

WeakSelf(self);

vc.selectLanguageBlock = ^(NSDictionary \* \_Nonnull languageDic) {

NSLog(@"选择了--> %@", languageDic);

weakSelf.toLanguageDic = languageDic;

// 修改了目标语言,重新调上传图片和翻译接口...

[self uploadImage:self.image];

};

PTBaseNavigationController \*nav = [[PTBaseNavigationController alloc] initWithRootViewController:vc];

self.definesPresentationContext = YES;

nav.modalPresentationStyle = UIModalPresentationOverFullScreen;

[self presentViewController:nav animated:YES completion:^{

}];

}

// 文本对照

- (void)textButtonAction:(id)sender{

NSLog(@"文本对照");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_ContrastButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

PTTextComparedViewController \*vc = [[PTTextComparedViewController alloc] init];

vc.fromText = self.detectedTextString;

vc.toText = self.translatedTextString;

vc.fromLanguageDic = self.fromLanguageDic;

vc.toLanguageDic = self.toLanguageDic;

[self.navigationController pushViewController:vc animated:YES];

}

// 导出图片

- (void)exportButtonAction:(id)sender{

NSLog(@"导出图片");

// [[PTEventRecord shareManager] addEventWithType:@"touch" Name:@"btn\_exportButton" Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

// 1.获取翻译内容:截屏全屏

UIImage \*fullImage = [PTImageTool fullScreenShot];

// 将图片裁剪出来，去掉顶部和底部

// 因为截屏时使用了UIGraphicsBeginImageContextWithOptions, 所以裁剪时也要将裁剪rect范围根据屏幕分辨率[UIScreen mainScreen].scale缩放下

UIImage \*oneImage = [PTImageTool cropImageWithImage:fullImage Rect:CGRectMake(0\*[UIScreen mainScreen].scale, StatusbarAndNavigationbarHeight\*[UIScreen mainScreen].scale, self.rect.size.width\*[UIScreen mainScreen].scale, self.rect.size.height\*[UIScreen mainScreen].scale)];

// 2.获取底部分享图片

UIImage \*bottomImage = [UIImage imageNamed:@"share"];

// 缩放图片 (根据屏幕分辨率[UIScreen mainScreen].scale缩放下,保持和截屏的图片比例一样)

bottomImage = [PTImageTool reSizeImage:bottomImage toSize:CGSizeMake(bottomImage.size.width\*[UIScreen mainScreen].scale, bottomImage.size.height\*[UIScreen mainScreen].scale)];

// 3.拼接图片和底部二维码图片

//合并

UIImage \*shareImage = [PTImageTool mergeImageWithImage:oneImage image:bottomImage];

// 保存图片到相册

[[PHPhotoLibrary sharedPhotoLibrary] performChanges:^{

[PHAssetChangeRequest creationRequestForAssetFromImage:shareImage];

} completionHandler:^(BOOL success, NSError \* \_Nullable error) {

if (success) {

NSLog(@"保存图片成功!");

dispatch\_async(dispatch\_get\_main\_queue(), ^{

UIAlertController \*alert = [UIAlertController alertControllerWithTitle:NSLocalizedString(@"Picture Saved Successfully", nil) message:nil preferredStyle:UIAlertControllerStyleAlert];

[alert addAction:[UIAlertAction actionWithTitle:@"OK" style:UIAlertActionStyleDefault handler:^(UIAlertAction \* \_Nonnull action) {

// NSLog(@"用户点击了OK");

}]];

[self presentViewController:alert animated:YES completion:nil];

});

} else {

NSLog(@"保存图片失败:%@", error);

}

}];

}

#pragma mark - SetGet

- (void)setFromLanguageDic:(NSDictionary \*)fromLanguageDic{

\_fromLanguageDic = fromLanguageDic;

if (self.leftLanguageButton) {

[self.leftLanguageButton setTitle:[fromLanguageDic objectForKey:@"name"] forState:UIControlStateNormal];

// 根据button文字内容计算宽度

CGSize buttonSize = [self.leftLanguageButton.titleLabel.text boundingRectWithSize:CGSizeMake(MAXFLOAT, 36)

options:NSStringDrawingTruncatesLastVisibleLine | NSStringDrawingUsesLineFragmentOrigin | NSStringDrawingUsesFontLeading

attributes:@{ NSFontAttributeName:self.leftLanguageButton.titleLabel.font}

context:nil].size;

self.leftLanguageButton.frame = CGRectMake(self.convertButton.frame.origin.x -16 -(buttonSize.width>90?90:buttonSize.width), self.convertButton.frame.origin.y, buttonSize.width>90?90:buttonSize.width, 36);

}

}

- (void)setToLanguageDic:(NSDictionary \*)toLanguageDic{

\_toLanguageDic = toLanguageDic;

if (self.rightLanguageButton) {

[self.rightLanguageButton setTitle:[toLanguageDic objectForKey:@"name"] forState:UIControlStateNormal];

// 根据button文字内容计算宽度

CGSize buttonSize1 = [self.rightLanguageButton.titleLabel.text boundingRectWithSize:CGSizeMake(MAXFLOAT, 36)

options:NSStringDrawingTruncatesLastVisibleLine | NSStringDrawingUsesLineFragmentOrigin | NSStringDrawingUsesFontLeading

attributes:@{ NSFontAttributeName:self.rightLanguageButton.titleLabel.font}

context:nil].size;

self.rightLanguageButton.frame = CGRectMake(self.convertButton.right +4, self.convertButton.frame.origin.y, buttonSize1.width>90?90:buttonSize1.width, 36);

self.rightArrowImageView.frame = CGRectMake(self.rightLanguageButton.right + 4, self.rightLanguageButton.top + 14, 8, 8);

}

}

#pragma mark - loadData

- (void)uploadImage:(UIImage \*)image{

[self clearTranslatedData];// 清除上次翻译的数据

// 如果左边是自动，不传源语言参数, 否则传源语言参数

NSString \*url;

if ([[self.fromLanguageDic objectForKey:@"name"] isEqualToString:@"Auto"]) {// 不传源语言

url = [NSString stringWithFormat:@"https://gmgmapps.com/image/texts/?t=%@&god=gm", [self.toLanguageDic objectForKey:@"code"]];

}else{// 传源语言

url = [NSString stringWithFormat:@"https://gmgmapps.com/image/texts/?t=%@&s=%@&god=gm", [self.toLanguageDic objectForKey:@"code"],[self.fromLanguageDic objectForKey:@"code"]];

}

NSString \*ts = [PTTools getNowTimeTimestamp];

NSString \*nonce = [PTTools randomStringWithLength:[PTTools getRandomNumber:10 to:18]];

NSString \*str = [PTTools sha1:[NSString stringWithFormat:@"%@%@%s" , ts, nonce, SECRET]];

NSDictionary \*headerDic = @{@"X-TimeStamp":ts,@"X-Nonce":nonce,@"X-Signature":str};

[MBProgressHUD showHUD];

WeakSelf(self);

// 上传图片-------

[[AFHTTPSessionManager manager] POST:url parameters:nil headers:headerDic constructingBodyWithBlock:^(id<AFMultipartFormData> \_Nonnull formData) {

NSData \*imgData = UIImageJPEGRepresentation(image, 0.5);

[formData appendPartWithFileData:imgData name:@"file" fileName:[NSString stringWithFormat:@"%@.jpg", [PTTools getNowTimeTimestamp]] mimeType:@"image/\*"];

} progress:^(NSProgress \* \_Nonnull uploadProgress) {

NSLog(@"uploadProgress = %@",uploadProgress);

} success:^(NSURLSessionDataTask \* \_Nonnull task, id \_Nullable responseObject) {

if ([[responseObject objectForKey:@"status"] isEqualToString:@"waiting"]) {// 成功

self->\_postCount = 0;

// 请求成功根据id查询识别与翻译结果-------

[weakSelf getImageCheckWithID:[responseObject objectForKey:@"id"] ts:ts nonce:nonce];

}else{// 错误

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

}

} failure:^(NSURLSessionDataTask \* \_Nullable task, NSError \* \_Nonnull error) {

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

}];

}

/// 根据uuid查询识别与翻译结果

/// @param idStr id

/// @param ts 时间戳

/// @param nonce 随机字符串

- (void)getImageCheckWithID:(NSString \*)idStr ts:(NSString \*)ts nonce:(NSString \*)nonce{

if (\_postCount >= 20) {

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

return;

}

WeakSelf(self);

AFHTTPSessionManager \*manager = [AFHTTPSessionManager manager];

manager.requestSerializer.timeoutInterval = 20;

// request

NSError \*requestError = nil;

NSMutableURLRequest \*request = [manager.requestSerializer requestWithMethod:@"POST" URLString:@"https://gmgmapps.com/image/status/query/" parameters:nil error:&requestError];

// body

NSString \*jsonStr = @{@"id":idStr}.mj\_JSONString;

NSData \*postData = [jsonStr dataUsingEncoding:NSUTF8StringEncoding];

[request setHTTPBody:postData];

// 设置header

NSString \*str222 = [PTTools sha1:[NSString stringWithFormat:@"%@%@%@%s" , [PTTools md5:jsonStr], ts, nonce, SECRET]];

[request setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];

[request setValue:ts forHTTPHeaderField:@"X-TimeStamp"];

[request setValue:nonce forHTTPHeaderField:@"X-Nonce"];

[request setValue:str222 forHTTPHeaderField:@"X-Signature"];

NSURLSessionDataTask \*dataTask = [manager.session dataTaskWithRequest:request completionHandler:^(NSData \* \_Nullable data, NSURLResponse \* \_Nullable response, NSError \* \_Nullable error) {

if (error) { // failed

NSLog("\n请求失败：%@", error);

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

}else{ // succese

NSError \* error = nil;

id responseObject = [NSJSONSerialization JSONObjectWithData:data options:NSJSONReadingMutableContainers error:&error];

if (error) {

NSLog("\n解析数据失败：%@", responseObject);

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Json error"];

});

}else {

NSLog("\n请求成功 数据：\n%@", responseObject);

if ([[responseObject objectForKey:@"status"] isEqualToString:@"completed"]) {// 成功

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

});

NSArray \*array = [responseObject objectForKey:@"result"];

if (array.count > 0) {

// 先取第一个下标-------

NSDictionary \*testDic = array.firstObject;

self.detectedTextString = [testDic objectForKey:@"description"];

self.translatedTextString = [testDic objectForKey:@"translated\_text"];

// 图片上加一层灰色透明遮罩------------

dispatch\_async(dispatch\_get\_main\_queue(), ^{

self.hoodView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, self.rect.size.width, self.rect.size.height)];

self.hoodView.backgroundColor = [[UIColor blackColor] colorWithAlphaComponent:0.5];

[self.imageView addSubview:self.hoodView];

self.hoodView.userInteractionEnabled = YES;

//单击

UITapGestureRecognizer \*clearHoodViewTap = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector(clearHoodViewTapAction:)];

[self.hoodView addGestureRecognizer:clearHoodViewTap];

NSDictionary \*boundingPolyDic = [testDic objectForKey:@"boundingPoly"];

NSArray \*verticesArray = [boundingPolyDic objectForKey:@"vertices"];

double x = [[verticesArray[0] objectForKey:@"x"] doubleValue];

double y = [[verticesArray[0] objectForKey:@"y"] doubleValue];

double width = [[verticesArray[1] objectForKey:@"x"] doubleValue] - [[verticesArray[0] objectForKey:@"x"] doubleValue];

double height = [[verticesArray[2] objectForKey:@"y"] doubleValue] - [[verticesArray[1] objectForKey:@"y"] doubleValue];

// 比例

double bl = self.image.size.width / SCREEN\_WIDTH;

NSLog(@"dsfhdlsghldsgfa--> %f tukuan--> %f fankuan--> %f tugao--> %f fangao --> %f", bl, self.image.size.width, width, self.image.size.height, height);

UILabel \*label = [[UILabel alloc] initWithFrame:CGRectMake(x/bl , (y/bl)-StatusbarAndNavigationbarHeight, width/bl+10, height/bl+20)];

label.backgroundColor = [UIColor clearColor];

// label.alpha = 0.3;

label.textColor = [UIColor whiteColor];

label.numberOfLines = 0;

label.text = [testDic objectForKey:@"translated\_text"];

label.adjustsFontSizeToFitWidth = YES;

[self.hoodView addSubview:label];

});

// 如果是自动检测语言翻译,将源语言改成识别之后的语言

if ([[self.fromLanguageDic objectForKey:@"name"] isEqualToString:@"Auto"]) {

// 记录识别出来的源语言code,自动检测的情况要修改左边源语言button

NSString \*code = [testDic objectForKey:@"source\_language\_code"];

[self setLeftLanguageButtonCode:code];

}else{

// 非自动检测语言翻译，不用修改

}

}

//-------------------------------

}else if ([[responseObject objectForKey:@"status"] isEqualToString:@"waiting"] || [[responseObject objectForKey:@"status"] isEqualToString:@"processing"]){

// 最多请求5次

self->\_postCount += 1;

[weakSelf getImageCheckWithID:idStr ts:ts nonce:nonce];

}else{// 错误

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[MBProgressHUD hideHUD];

[MBProgressHUD showMessage:@"Please check the network"];

});

}

}

}

}];

[dataTask resume];

}

//单击方法

-(void)clearHoodViewTapAction:(UITapGestureRecognizer \*)tap{

self.hoodView.hidden = !self.hoodView.hidden;

}

// 每次翻译前都要清楚上一次翻译的信息

- (void)clearTranslatedData{

[self.hoodView removeFromSuperview];// 已显示的结果删除

self.detectedTextString = @"";

self.translatedTextString = @"";

}

- (UILabel \*)labelWithDic:(NSDictionary \*)dic{

UILabel \*label = [[UILabel alloc] init];

label.text = [dic objectForKey:@"translated\_text"];

label.textColor = [UIColor whiteColor];

// 坐标 = 百分比 \* 图片尺寸

double x = [[dic objectForKey:@"x"] doubleValue] \* self.imageView.size.width;

double y = [[dic objectForKey:@"y"] doubleValue] \* self.imageView.size.height;

double width = [[dic objectForKey:@"width"] doubleValue] \* self.imageView.size.width;

double height = [[dic objectForKey:@"height"] doubleValue] \* self.imageView.size.height;

// 根据内容计算宽度高度(因为有时候后台返回的宽度高度不符合文字内容会造成省略号...,所以根据内容计算一边，取最大值)

CGSize attrStrSize = [label.text boundingRectWithSize:CGSizeMake(MAXFLOAT, MAXFLOAT) options:NSStringDrawingUsesLineFragmentOrigin attributes:@{NSFontAttributeName:label.font} context:nil].size;

label.frame = CGRectMake(x, y, width>attrStrSize.width?width:attrStrSize.width, height>attrStrSize.height?height:attrStrSize.height);

return label;

}

- (void)setLeftLanguageButtonCode:(NSString \*)code{

// 遍历资源，设置左边源语言button

NSArray \*dataArray = @[

@[@{@"name":@"Afrikaans",@"code":@"af"},@{@"name":@"Albanian",@"code":@"sq"},@{@"name":@"Amharic",@"code":@"am"},@{@"name":@"Arabic",@"code":@"ar"},@{@"name":@"Armenian",@"code":@"hy"},@{@"name":@"Azerbaijani",@"code":@"az"}],

@[@{@"name":@"Basque",@"code":@"eu"},@{@"name":@"Belarusian",@"code":@"be"},@{@"name":@"Bengali",@"code":@"bn"},@{@"name":@"Bosnian",@"code":@"bs"},@{@"name":@"Bulgarian",@"code":@"bg"}],

@[@{@"name":@"Catalan",@"code":@"ca"},@{@"name":@"Cebuano",@"code":@"ceb"},@{@"name":@"Chichewa",@"code":@"ny"},@{@"name":@"Chinese (Simplified)",@"code":@"zh-CN"},@{@"name":@"Chinese (Traditional)",@"code":@"zh-TW"},@{@"name":@"Corsican",@"code":@"co"},@{@"name":@"Croatian",@"code":@"hr"},@{@"name":@"Czech",@"code":@"cs"}],

@[@{@"name":@"Danish",@"code":@"da"},@{@"name":@"Dutch",@"code":@"nl"}],

@[@{@"name":@"English",@"code":@"en"},@{@"name":@"Esperanto",@"code":@"eo"},@{@"name":@"Estonian",@"code":@"et"}],

@[@{@"name":@"Filipino",@"code":@"tl"},@{@"name":@"Finnish",@"code":@"fi"},@{@"name":@"French",@"code":@"fr"},@{@"name":@"Frisian",@"code":@"fy"}],

@[@{@"name":@"Galician",@"code":@"gl"},@{@"name":@"Georgian",@"code":@"ka"},@{@"name":@"German",@"code":@"de"},@{@"name":@"Greek",@"code":@"el"},@{@"name":@"Gujarati",@"code":@"gu"}],

@[@{@"name":@"Haitian Creole",@"code":@"ht"},@{@"name":@"Hausa",@"code":@"ha"},@{@"name":@"Hawaiian",@"code":@"haw"},@{@"name":@"Hebrew",@"code":@"iw"},@{@"name":@"Hindi",@"code":@"hi"},@{@"name":@"Hmong",@"code":@"hmn"},@{@"name":@"Hungarian",@"code":@"hu"}],

@[@{@"name":@"Icelandic",@"code":@"is"},@{@"name":@"Igbo",@"code":@"ig"},@{@"name":@"Indonesian",@"code":@"id"},@{@"name":@"Irish",@"code":@"ga"},@{@"name":@"Italian",@"code":@"it"}],

@[@{@"name":@"Japanese",@"code":@"ja"},@{@"name":@"Javanese",@"code":@"jw"}],

@[@{@"name":@"Kannada",@"code":@"kn"},@{@"name":@"Kazakh",@"code":@"kk"},@{@"name":@"Khmer",@"code":@"km"},@{@"name":@"Kinyarwanda",@"code":@"rw"},@{@"name":@"Korean",@"code":@"ko"},@{@"name":@"Kurdish (Kurmanji)",@"code":@"ku"},@{@"name":@"Kyrgyz",@"code":@"ky"}],

@[@{@"name":@"Lao",@"code":@"lo"},@{@"name":@"Latin",@"code":@"la"},@{@"name":@"Latvian",@"code":@"lv"},@{@"name":@"Lithuanian",@"code":@"lt"},@{@"name":@"Luxembourgish",@"code":@"lb"}],

@[@{@"name":@"Macedonian",@"code":@"mk"},@{@"name":@"Malagasy",@"code":@"mg"},@{@"name":@"Malay",@"code":@"ms"},@{@"name":@"Malayalam",@"code":@"ml"},@{@"name":@"Maltese",@"code":@"mt"},@{@"name":@"Maori",@"code":@"mi"},@{@"name":@"Marathi",@"code":@"mr"},@{@"name":@"Mongolian",@"code":@"mn"},@{@"name":@"Myanmar (Burmese)",@"code":@"my"}],

@[@{@"name":@"Nepali",@"code":@"ne"},@{@"name":@"Norwegian",@"code":@"no"}],

@[@{@"name":@"Odia (Oriya)",@"code":@"or"}],

@[@{@"name":@"Pashto",@"code":@"ps"},@{@"name":@"Persian",@"code":@"fa"},@{@"name":@"Polish",@"code":@"pl"},@{@"name":@"Portuguese",@"code":@"pt"},@{@"name":@"Punjabi",@"code":@"pa"}],

@[@{@"name":@"Romanian",@"code":@"ro"},@{@"name":@"Russian",@"code":@"ru"}],

@[@{@"name":@"Samoan",@"code":@"sm"},@{@"name":@"Scots Gaelic",@"code":@"gd"},@{@"name":@"Serbian",@"code":@"sr"},@{@"name":@"Sesotho",@"code":@"st"},@{@"name":@"Shona",@"code":@"sn"},@{@"name":@"Sindhi",@"code":@"sd"},@{@"name":@"Sinhala",@"code":@"si"},@{@"name":@"Slovak",@"code":@"sk"},@{@"name":@"Slovenian",@"code":@"sl"},@{@"name":@"Somali",@"code":@"so"},@{@"name":@"Spanish",@"code":@"es"},@{@"name":@"Sundanese",@"code":@"su"},@{@"name":@"Swahili",@"code":@"sw"},@{@"name":@"Swedish",@"code":@"sv"}],

@[@{@"name":@"Tajik",@"code":@"tg"},@{@"name":@"Tamil",@"code":@"ta"},@{@"name":@"Tatar",@"code":@"tt"},@{@"name":@"Telugu",@"code":@"te"},@{@"name":@"Thai",@"code":@"th"},@{@"name":@"Turkish",@"code":@"tr"},@{@"name":@"Turkmen",@"code":@"tk"}],

@[@{@"name":@"Ukrainian",@"code":@"uk"},@{@"name":@"Urdu",@"code":@"ur"},@{@"name":@"Uyghur",@"code":@"ug"},@{@"name":@"Uzbek",@"code":@"uz"}],

@[@{@"name":@"Vietnamese",@"code":@"vi"}],

@[@{@"name":@"Welsh",@"code":@"cy"}],

@[@{@"name":@"Xhosa",@"code":@"xh"}],

@[@{@"name":@"Yiddish",@"code":@"yi"},@{@"name":@"Yoruba",@"code":@"yo"}],

@[@{@"name":@"Zulu",@"code":@"zu"}],

];

for (NSArray \*array in dataArray) {

for (NSDictionary \*dic in array) {

if ([[dic objectForKey:@"code"] isEqualToString:code]) {

dispatch\_async(dispatch\_get\_main\_queue(), ^{

self.fromLanguageDic = dic;

});

}else{

}

}

}

}

@end

#import <UIKit/UIKit.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface UIView (PTHUD)

@property(nonatomic, strong ,readonly)UIAlertController \*alertController;

// 加载框

-(void)showHUD:(NSString \*)message; // 没有菊花

-(void)showLoadHUD:(NSString \*)message; // 有菊花

-(void)hideHUD;

// 提示框

-(void)showAutoDismissHUD:(NSString \*)message;

-(void)showAutoDismissHUD:(NSString \*)message delay:(NSTimeInterval)delay;

// 弹出框

-(void)showError:(NSError \*)error;

-(void)showAlertView:(NSString \*)message ok:(void(^)(UIAlertAction \* action))ok cancel:(void(^)(UIAlertAction \* action))cancel;

@end

NS\_ASSUME\_NONNULL\_END

#import "UIView+PTHUD.h"

#import <objc/runtime.h>

#define KEY\_CC\_ALERT\_VIEW "UIView.AlertController"

@implementation UIView (PTHUD)

@dynamic alertController;

-(UIAlertController \*)alertController{

NSObject \* obj = objc\_getAssociatedObject(self, KEY\_CC\_ALERT\_VIEW);

if (obj && [obj isKindOfClass:[UIAlertController class]]){

return (UIAlertController \*)obj;

}

return nil;

}

-(void)setAlertController:(UIAlertController \*)alertController

{

if (nil == alertController){ return; }

objc\_setAssociatedObject(self, KEY\_CC\_ALERT\_VIEW, alertController, OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC);

}

#pragma mark - 加载框

-(void)showHUD:(NSString \*)message{

[self showHUD:message isLoad:NO];

}

-(void)showLoadHUD:(NSString \*)message{

[self showHUD:message isLoad:YES];

}

-(void)showHUD:(NSString \*)message isLoad:(BOOL)isLoad{

UIAlertController \*alertController = [self getAVC];

alertController.message = [NSString stringWithFormat:@"\n\n\n%@", message];

if (isLoad) {

[self findLabel:alertController.view succ:^(UIView \*label) {

dispatch\_async(dispatch\_get\_main\_queue(), ^{

UIActivityIndicatorView \*activityView = [[UIActivityIndicatorView alloc]initWithActivityIndicatorStyle:UIActivityIndicatorViewStyleWhiteLarge];

activityView.color = [UIColor lightGrayColor];

activityView.center = CGPointMake(label.width/2, 25);

[label addSubview:activityView];

[activityView startAnimating];

});

}];

}

[self.viewController presentViewController:alertController animated:YES completion:nil];

}

#pragma mark - 提示框

-(void)showAutoDismissHUD:(NSString \*)message{

[self showAutoDismissHUD:message delay:0.3];

}

-(void)showAutoDismissHUD:(NSString \*)message delay:(NSTimeInterval)delay{

UIAlertController \*alertController = [self getAVC];

alertController.message = message;

[self.viewController presentViewController:alertController animated:YES completion:nil];

[NSTimer scheduledTimerWithTimeInterval:delay

target:self

selector:@selector(hideHUD)

userInfo:alertController

repeats:NO];

}

-(void)hideHUD{

[[self getAVC] dismissViewControllerAnimated:YES completion:nil];

}

#pragma mark - 弹出框

- (void)showError:(NSError \*)error{

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[self showAlertView:error.localizedDescription ok:^(UIAlertAction \*action) {

} cancel:nil];

});

}

-(void)showAlertView:(NSString \*)message ok:(void(^)(UIAlertAction \* action))ok cancel:(void(^)(UIAlertAction \* action))cancel{

UIAlertController \*alertController = [UIAlertController alertControllerWithTitle:nil

message:message

preferredStyle:UIAlertControllerStyleAlert];

if (cancel) {

UIAlertAction \*cancelAction = [UIAlertAction actionWithTitle:@"取消" style:UIAlertActionStyleCancel handler:^(UIAlertAction \* \_Nonnull action) {

!cancel ? : cancel(action) ;

}];

[alertController addAction:cancelAction];

}

if (ok) {

UIAlertAction \*okAction = [UIAlertAction actionWithTitle:@"确定" style:UIAlertActionStyleDefault handler:^(UIAlertAction \* \_Nonnull action) {

!ok ? : ok(action) ;

}];

[alertController addAction:okAction];

}

[self.viewController presentViewController:alertController animated:YES completion:nil];

}

#pragma mark - Private methods

-(void)findLabel:(UIView\*)view succ:(void(^)(UIView \*label))succ{

for (UIView\* subView in view.subviews)

{

if ([subView isKindOfClass:[UILabel class]]) {

if (succ) {

succ(subView);

}

}

[self findLabel:subView succ:succ];

}

}

-(UIAlertController \*)getAVC{

if (!self.alertController) {

self.alertController = [UIAlertController alertControllerWithTitle:nil

message:@""

preferredStyle:UIAlertControllerStyleAlert];

}

return self.alertController;

}

@end

#import <UIKit/UIKit.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface UIView (PTAdditions)

@property (nonatomic) CGFloat top;

@property (nonatomic) CGFloat left;

@property (nonatomic) CGFloat right;

@property (nonatomic) CGFloat bottom;

@property (nonatomic) CGFloat width;

@property (nonatomic) CGFloat height;

@property (nonatomic) CGFloat centerX;

@property (nonatomic) CGFloat centerY;

@property (nonatomic) CGPoint origin;

@property (nonatomic) CGSize size;

- (UIViewController \*)viewController;

@end

NS\_ASSUME\_NONNULL\_END

#import "UIView+PTAdditions.h"

@implementation UIView (PTAdditions)

- (CGFloat)top {

return self.frame.origin.y;

}

- (void)setTop:(CGFloat)y {

CGRect frame = self.frame;

frame.origin.y = y;

self.frame = frame;

}

- (CGFloat)left {

return self.frame.origin.x;

}

- (void)setLeft:(CGFloat)x {

CGRect frame = self.frame;

frame.origin.x = x;

self.frame = frame;

}

- (CGFloat)right {

return self.frame.origin.x + self.frame.size.width;

}

- (void)setRight:(CGFloat)right {

CGRect frame = self.frame;

frame.origin.x = right - frame.size.width;

self.frame = frame;

}

- (CGFloat)bottom {

return self.frame.origin.y + self.frame.size.height;

}

- (void)setBottom:(CGFloat)bottom {

CGRect frame = self.frame;

frame.origin.y = bottom - frame.size.height;

self.frame = frame;

}

- (CGFloat)centerX {

return self.center.x;

}

- (void)setCenterX:(CGFloat)centerX {

self.center = CGPointMake(centerX, self.center.y);

}

- (CGFloat)centerY {

return self.center.y;

}

- (void)setCenterY:(CGFloat)centerY {

self.center = CGPointMake(self.center.x, centerY);

}

- (CGFloat)width {

return self.frame.size.width;

}

- (void)setWidth:(CGFloat)width {

CGRect frame = self.frame;

frame.size.width = width;

self.frame = frame;

}

- (CGFloat)height {

return self.frame.size.height;

}

- (void)setHeight:(CGFloat)height {

CGRect frame = self.frame;

frame.size.height = height;

self.frame = frame;

}

- (CGPoint)origin {

return self.frame.origin;

}

- (void)setOrigin:(CGPoint)origin {

CGRect frame = self.frame;

frame.origin = origin;

self.frame = frame;

}

- (CGSize)size {

return self.frame.size;

}

- (void)setSize:(CGSize)size {

CGRect frame = self.frame;

frame.size = size;

self.frame = frame;

}

- (UIViewController \*)viewController

{

if ([[self nextResponder] isKindOfClass:[UIViewController class]]) {

return (UIViewController \*)[self nextResponder];

}

for (UIView\* next = [self superview]; next; next = next.superview)

{

UIResponder \*nextResponder = [next nextResponder];

if ([nextResponder isKindOfClass:[UIViewController class]])

{

return (UIViewController \*)nextResponder;

}

}

return nil;

}

@end

#import <Foundation/Foundation.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTTools : NSObject

+ (void)createGIFfromURL:(NSURL\*)videoURL loopCount:(int)loopCount completion:(void(^)(NSURL \*GifURL))completionBlock;

+ (void)createGIFfromURL:(NSURL\*)videoURL frameCount:(int)frameCount delayTime:(float)delayTime loopCount:(int)loopCount completion:(void(^)(NSURL \*GifURL))completionBlock;

// 指定范围内的随机数

+ (int)getRandomNumber:(int)from to:(int)to;

// 随机字符串 - 生成指定长度的字符串

+ (NSString \*)randomStringWithLength:(NSInteger)len;

// 获取当前时间戳 （以毫秒为单位）

+(NSString \*)getNowTimeTimestamp;

// md5加密

+ (NSString \*)md5:(NSString \*)string;

// sha1加密方式

+ (NSString \*)sha1:(NSString \*)input;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTTools.h"

#import <MobileCoreServices/MobileCoreServices.h>

#import <AVFoundation/AVFoundation.h>

#import <CoreGraphics/CoreGraphics.h>

#import <ImageIO/ImageIO.h>

#import <CommonCrypto/CommonDigest.h>

#define gifFileName @"gifName.gif"

#define timeInterval @(600)

#define tolerance @(0.01)

typedef NS\_ENUM(NSInteger, GIFSize) {

GIFSizeVeryLow = 2,

GIFSizeLow = 3,

GIFSizeMedium = 5,

GIFSizeHigh = 7,

GIFSizeOriginal = 10

};

@implementation PTTools

+ (void)createGIFfromURL:(NSURL\*)videoURL loopCount:(int)loopCount completion:(void(^)(NSURL \*GifURL))completionBlock{

// 大小

AVURLAsset \*asset = [AVURLAsset assetWithURL:videoURL];

float videoWidth = [[[asset tracksWithMediaType:AVMediaTypeVideo] objectAtIndex:0] naturalSize].width;

float videoHeight = [[[asset tracksWithMediaType:AVMediaTypeVideo] objectAtIndex:0] naturalSize].height;

GIFSize optimalSize = GIFSizeMedium;

if (videoWidth >= 1200 || videoHeight >= 1200){

optimalSize = GIFSizeVeryLow;

}

else if (videoWidth >= 800 || videoHeight >= 800){

optimalSize = GIFSizeLow;

}

else if (videoWidth >= 400 || videoHeight >= 400){

optimalSize = GIFSizeMedium;

}

else if (videoWidth < 400|| videoHeight < 400){

optimalSize = GIFSizeHigh;

}

// 每秒取贞的时间点

float videoLength = (float)asset.duration.value/asset.duration.timescale;

int framesPerSecond = 4;

int frameCount = videoLength \* framesPerSecond;

float increment = (float)videoLength / frameCount;

NSMutableArray \*timePoints = [NSMutableArray array];

for (int currentFrame = 0; currentFrame < frameCount; ++currentFrame) {

float seconds = (float)increment \* currentFrame;

CMTime time = CMTimeMakeWithSeconds(seconds, [timeInterval intValue]);

[timePoints addObject:[NSValue valueWithCMTime:time]];

}

// 循环属性

NSDictionary \*fileProperties = [self filePropertiesWithLoopCount:loopCount];

// 延迟属性

float delayTime = 0.1f;

NSDictionary \*frameProperties = [self framePropertiesWithDelayTime:delayTime];

dispatch\_group\_t gifQueue = dispatch\_group\_create();

dispatch\_group\_enter(gifQueue);

\_\_block NSURL \*gifURL;

dispatch\_async(dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0), ^{

gifURL = [self createGIFforTimePoints:timePoints fromURL:videoURL fileProperties:fileProperties frameProperties:frameProperties frameCount:frameCount gifSize:optimalSize];

dispatch\_group\_leave(gifQueue);

});

dispatch\_group\_notify(gifQueue, dispatch\_get\_main\_queue(), ^{

completionBlock(gifURL);

});

}

+ (void)createGIFfromURL:(NSURL\*)videoURL frameCount:(int)frameCount delayTime:(float)delayTime loopCount:(int)loopCount completion:(void(^)(NSURL \*GifURL))completionBlock{

// 循环属性

NSDictionary \*fileProperties = [self filePropertiesWithLoopCount:loopCount];

// 延迟属性

NSDictionary \*frameProperties = [self framePropertiesWithDelayTime:delayTime];

// 大小

AVURLAsset \*asset = [AVURLAsset assetWithURL:videoURL];

float videoLength = (float)asset.duration.value/asset.duration.timescale;

float increment = (float)videoLength/frameCount;

NSMutableArray \*timePoints = [NSMutableArray array];

for (int currentFrame = 0; currentFrame<frameCount; ++currentFrame) {

float seconds = (float)increment \* currentFrame;

CMTime time = CMTimeMakeWithSeconds(seconds, [timeInterval intValue]);

[timePoints addObject:[NSValue valueWithCMTime:time]];

}

dispatch\_group\_t gifQueue = dispatch\_group\_create();

dispatch\_group\_enter(gifQueue);

\_\_block NSURL \*gifURL;

dispatch\_async(dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0), ^{

gifURL = [self createGIFforTimePoints:timePoints fromURL:videoURL fileProperties:fileProperties frameProperties:frameProperties frameCount:frameCount gifSize:GIFSizeMedium];

dispatch\_group\_leave(gifQueue);

});

dispatch\_group\_notify(gifQueue, dispatch\_get\_main\_queue(), ^{

completionBlock(gifURL);

});

}

#pragma mark - Base methods

+ (NSURL \*)createGIFforTimePoints:(NSArray \*)timePoints fromURL:(NSURL \*)url fileProperties:(NSDictionary \*)fileProperties frameProperties:(NSDictionary \*)frameProperties frameCount:(int)frameCount gifSize:(GIFSize)gifSize{

NSString \*temporaryFile = [NSTemporaryDirectory() stringByAppendingString:gifFileName];

NSURL \*fileURL = [NSURL fileURLWithPath:temporaryFile];

if (fileURL == nil) return nil;

CGImageDestinationRef destination = CGImageDestinationCreateWithURL((\_\_bridge CFURLRef)fileURL, kUTTypeGIF , frameCount, NULL);

AVURLAsset \*asset = [AVURLAsset URLAssetWithURL:url options:nil];

AVAssetImageGenerator \*generator = [AVAssetImageGenerator assetImageGeneratorWithAsset:asset];

generator.appliesPreferredTrackTransform = YES;

CMTime tol = CMTimeMakeWithSeconds([tolerance floatValue], [timeInterval intValue]);

generator.requestedTimeToleranceBefore = tol;

generator.requestedTimeToleranceAfter = tol;

NSError \*error = nil;

CGImageRef previousImageRefCopy = nil;

for (NSValue \*time in timePoints) {

CGImageRef imageRef;

if ((float)gifSize/10 != 1) {

imageRef = createImageWithScale([generator copyCGImageAtTime:[time CMTimeValue] actualTime:nil error:&error], (float)gifSize/10);

}

else{

imageRef = [generator copyCGImageAtTime:[time CMTimeValue] actualTime:nil error:&error];

}

if (error) {

NSLog(@"Error copying image: %@", error);

}

if (imageRef) {

CGImageRelease(previousImageRefCopy);

previousImageRefCopy = CGImageCreateCopy(imageRef);

}

else if (previousImageRefCopy) {

imageRef = CGImageCreateCopy(previousImageRefCopy);

}

else {

NSLog(@"Error copying image and no previous frames to duplicate");

return nil;

}

CGImageDestinationAddImage(destination, imageRef, (CFDictionaryRef)frameProperties);

CGImageRelease(imageRef);

}

CGImageRelease(previousImageRefCopy);

CGImageDestinationSetProperties(destination, (CFDictionaryRef)fileProperties);

if (!CGImageDestinationFinalize(destination)) {

NSLog(@"Failed to finalize GIF destination: %@", error);

if (destination != nil) {

CFRelease(destination);

}

return nil;

}

CFRelease(destination);

return fileURL;

}

#pragma mark - Helpers

CGImageRef createImageWithScale(CGImageRef imageRef, float scale) {

CGSize newSize = CGSizeMake(CGImageGetWidth(imageRef)\*scale, CGImageGetHeight(imageRef)\*scale);

CGRect newRect = CGRectIntegral(CGRectMake(0, 0, newSize.width, newSize.height));

UIGraphicsBeginImageContextWithOptions(newSize, NO, 0);

CGContextRef context = UIGraphicsGetCurrentContext();

if (!context) return nil;

CGContextSetInterpolationQuality(context, kCGInterpolationHigh);

CGAffineTransform flipVertical = CGAffineTransformMake(1, 0, 0, -1, 0, newSize.height);

CGContextConcatCTM(context, flipVertical);

CGContextDrawImage(context, newRect, imageRef);

CFRelease(imageRef);

imageRef = CGBitmapContextCreateImage(context);

UIGraphicsEndImageContext();

return imageRef;

}

#pragma mark - Properties

+ (NSDictionary \*)filePropertiesWithLoopCount:(int)loopCount {

return @{(NSString \*)kCGImagePropertyGIFDictionary:@{(NSString \*)kCGImagePropertyGIFLoopCount:@(loopCount)}};

}

+ (NSDictionary \*)framePropertiesWithDelayTime:(float)delayTime {

return @{(NSString \*)kCGImagePropertyGIFDictionary:@{(NSString \*)kCGImagePropertyGIFDelayTime:@(delayTime)},

(NSString \*)kCGImagePropertyColorModel:(NSString \*)kCGImagePropertyColorModelRGB};

}

// 指定范围内的随机数

+ (int)getRandomNumber:(int)from to:(int)to {

return (int)(from + (arc4random() % (to - from + 1)));

}

// 随机字符串 - 生成指定长度的字符串

+ (NSString \*)randomStringWithLength:(NSInteger)len {

NSString \*letters = @"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

NSMutableString \*randomString = [NSMutableString stringWithCapacity: len];

for (NSInteger i = 0; i < len; i++) {

[randomString appendFormat: @"%C", [letters characterAtIndex: arc4random\_uniform([letters length])]];

}

return randomString;

}

// 获取当前时间戳 （以毫秒为单位）

+(NSString \*)getNowTimeTimestamp{

NSDateFormatter \*formatter = [[NSDateFormatter alloc] init] ;

[formatter setDateStyle:NSDateFormatterMediumStyle];

[formatter setTimeStyle:NSDateFormatterShortStyle];

[formatter setDateFormat:@"YYYY-MM-dd HH:mm:ss SSS"]; // ----------设置你想要的格式,hh与HH的区别:分别表示12小时制,24小时制

//设置时区,这个对于时间的处理有时很重要

NSTimeZone\* timeZone = [NSTimeZone timeZoneWithName:@"Asia/Shanghai"];

[formatter setTimeZone:timeZone];

NSDate \*datenow = [NSDate date];//现在时间,你可以输出来看下是什么格式

NSString \*timeSp = [NSString stringWithFormat:@"%ld", (long)[datenow timeIntervalSince1970]\*1000];

return timeSp;

}

// MD5加密

#pragma mark - md5加密方法

//+ (NSString \*)md5:(NSString \*)string {

// const char \*cStr = [string UTF8String];

// unsigned char digest[CC\_MD5\_DIGEST\_LENGTH];

// CC\_MD5(cStr, (CC\_LONG)strlen(cStr), digest);

// NSMutableString \*result = [NSMutableString stringWithCapacity:CC\_MD5\_DIGEST\_LENGTH \* 2];

// for (int i = 0; i < CC\_MD5\_DIGEST\_LENGTH; i++) {

// [result appendFormat:@"%02X", digest[i]];

// }

// return result;

//}

+ (NSString \*)md5:(NSString \*)string {

const char \*original\_str = [string UTF8String];

unsigned char result[CC\_MD5\_DIGEST\_LENGTH];

CC\_MD5(original\_str, strlen(original\_str), result);

NSMutableString \*hash = [NSMutableString string];

for (int i = 0; i < 16; i++)

[hash appendFormat:@"%02X", result[i]];

return [hash lowercaseString];

}

#pragma mark - SHA1加密

// sha1加密方式

+ (NSString \*)sha1:(NSString \*)input{

// NSData \*data = [input dataUsingEncoding:NSUTF8StringEncoding];

//

// uint8\_t digest[CC\_SHA1\_DIGEST\_LENGTH];

//

// CC\_SHA1(data.bytes, (unsigned int)data.length, digest);

//

// NSMutableString \*output = [NSMutableString stringWithCapacity:CC\_SHA1\_DIGEST\_LENGTH \* 2];

//

// for(int i=0; i<CC\_SHA1\_DIGEST\_LENGTH; i++) {

// [output appendFormat:@"%02x", digest[i]];

// }

// return output;

const char \*cstr = [input UTF8String];

//使用对应的CC\_SHA1,CC\_SHA256,CC\_SHA384,CC\_SHA512的长度分别是20,32,48,64

unsigned char digest[CC\_SHA1\_DIGEST\_LENGTH];

//使用对应的CC\_SHA256,CC\_SHA384,CC\_SHA512

CC\_SHA1(cstr, strlen(cstr), digest);

NSMutableString\* result = [NSMutableString stringWithCapacity:CC\_SHA1\_DIGEST\_LENGTH \* 2];

for(int i = 0; i < CC\_SHA1\_DIGEST\_LENGTH; i++) {

[result appendFormat:@"%02x", digest[i]];

}

return result;

}

@end

#import <Foundation/Foundation.h>

#import <AVFoundation/AVFoundation.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTMotionManager : NSObject

@property(nonatomic, assign)UIDeviceOrientation deviceOrientation;

@property(nonatomic, assign)AVCaptureVideoOrientation videoOrientation;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTMotionManager.h"

#import <CoreMotion/CoreMotion.h>

@interface PTMotionManager ()

@property(nonatomic, strong) CMMotionManager \* motionManager;

@end

@implementation PTMotionManager

-(instancetype)init

{

self = [super init];

if (self) {

\_motionManager = [[CMMotionManager alloc] init];

\_motionManager.deviceMotionUpdateInterval = 1/15.0;

if (!\_motionManager.deviceMotionAvailable) {

\_motionManager = nil;

return self;

}

@weakify(self)

[\_motionManager startDeviceMotionUpdatesToQueue:[NSOperationQueue currentQueue] withHandler: ^(CMDeviceMotion \*motion, NSError \*error){

@strongify(self)

[self performSelectorOnMainThread:@selector(handleDeviceMotion:) withObject:motion waitUntilDone:YES];

}];

}

return self;

}

- (void)handleDeviceMotion:(CMDeviceMotion \*)deviceMotion{

double x = deviceMotion.gravity.x;

double y = deviceMotion.gravity.y;

if (fabs(y) >= fabs(x)) {

if (y >= 0) {

\_deviceOrientation = UIDeviceOrientationPortraitUpsideDown;

\_videoOrientation = AVCaptureVideoOrientationPortraitUpsideDown;

} else {

\_deviceOrientation = UIDeviceOrientationPortrait;

\_videoOrientation = AVCaptureVideoOrientationPortrait;

}

} else {

if (x >= 0) {

\_deviceOrientation = UIDeviceOrientationLandscapeRight;

\_videoOrientation = AVCaptureVideoOrientationLandscapeRight;

} else {

\_deviceOrientation = UIDeviceOrientationLandscapeLeft;

\_videoOrientation = AVCaptureVideoOrientationLandscapeLeft;

}

}

}

-(void)dealloc{

NSLog(@"陀螺仪对象销毁了");

[\_motionManager stopDeviceMotionUpdates];

}

@end

#import <Foundation/Foundation.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTImageTool : NSObject

// 全屏截图

+ (UIImage \*)fullScreenShot;

// 裁剪图片

+ (UIImage \*)cropImageWithImage:(UIImage \*)image Rect:(CGRect )rect;

// 合并图片

+ (UIImage \*)mergeImageWithImage:(UIImage \*)oneImage image:(UIImage \*)twoImage;

// 缩放图片(自定义大小)

+ (UIImage \*) reSizeImage:(UIImage \*)image toSize:(CGSize)reSize;

// 缩放图片(等比缩放)

+ (UIImage \*) scaleImage:(UIImage \*)image toScale:(float)scaleSize;

+ (UIImage \*)imageFromColor:(UIColor \*)color withSize:(CGSize)size;

+ (UIImage \*)imageWithColor:(UIColor \*)color;

@end

NS\_ASSUME\_NONNULL\_END

#import "PTImageTool.h"

@implementation PTImageTool

+ (UIImage \*)fullScreenShot{

UIWindow \*screenWindow = [[UIApplication sharedApplication] keyWindow];

// UIGraphicsBeginImageContext(screenWindow.frame.size);//全屏截图，包括window// 会模糊

/\*

UIGraphicsBeginImageContextWithOptions(CGSize size, BOOL opaque, CGFloat scale)

opaque 透明度，不透明设为YES；

scale 缩放因子，设0时系统自动设置缩放比例图片清晰；设1.0时模糊. 表示位图的缩放比例，如果设置为 0，表示让图片的缩放因子根据屏幕的分辨率而变化。和 [UIScreen mainScreen].scale相等的。

官方文档说明：The scale factor to apply to the bitmap. If you specify a value of 0.0, the scale factor is set to the scale factor of the device’s main screen.

\*/

UIGraphicsBeginImageContextWithOptions(screenWindow.frame.size,YES, 0.0); // 不模糊

[screenWindow.layer renderInContext:UIGraphicsGetCurrentContext()];

UIImage \*image = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();

return image;

}

+ (UIImage \*)cropImageWithImage:(UIImage \*)image Rect:(CGRect )rect{

UIImage \*image2 = [UIImage imageWithCGImage:CGImageCreateWithImageInRect([image CGImage], rect)];

UIGraphicsEndImageContext();

return image2;

}

+ (UIImage \*)mergeImageWithImage:(UIImage \*)oneImage image:(UIImage \*)twoImage{

//计算画布大小

CGFloat width = oneImage.size.width;

CGFloat height = oneImage.size.height + twoImage.size.height;

CGSize resultSize = CGSizeMake(width, height);

// UIGraphicsBeginImageContext(resultSize);

UIGraphicsBeginImageContextWithOptions(resultSize,YES, 0.0); // 不模糊

//放第一个图片

CGRect oneRect = CGRectMake(0, 0, resultSize.width, oneImage.size.height);

[oneImage drawInRect:oneRect];

//放第二个图片

CGRect otherRect = CGRectMake(0, oneRect.size.height, resultSize.width, twoImage.size.height);

[twoImage drawInRect:otherRect];

UIImage \*resultImage = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();

return resultImage;

}

// 缩放图片(自定义大小)

+ (UIImage \*) reSizeImage:(UIImage \*)image toSize:(CGSize)reSize {

// UIGraphicsBeginImageContext(CGSizeMake(reSize.width, reSize.height));

UIGraphicsBeginImageContextWithOptions(CGSizeMake(reSize.width, reSize.height),YES, 0.0); // 不模糊

[image drawInRect:CGRectMake(0, 0, reSize.width, reSize.height)];

UIImage \*reSizeImage = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();

return reSizeImage;

}

// 缩放图片(等比缩放)

+ (UIImage \*) scaleImage:(UIImage \*)image toScale:(float)scaleSize {

// UIGraphicsBeginImageContext(CGSizeMake(image.size.width \* scaleSize, image.size.height \* scaleSize));

UIGraphicsBeginImageContextWithOptions(CGSizeMake(image.size.width \* scaleSize, image.size.height \* scaleSize),YES, 0.0); // 不模糊

[image drawInRect:CGRectMake(0, 0, image.size.width \* scaleSize, image.size.height \* scaleSize)];

UIImage \*scaledImage = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();

return scaledImage;

}

+ (UIImage \*)imageFromColor:(UIColor \*)color withSize:(CGSize)size{

CGRect rect=CGRectMake(0.0f, 0.0f, size.width, size.height);

UIGraphicsBeginImageContext(rect.size);

CGContextRef context = UIGraphicsGetCurrentContext();

CGContextSetFillColorWithColor(context, [color CGColor]);

CGContextFillRect(context, rect);

UIImage \*theImage = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();

return theImage;

}

+ (UIImage \*)imageWithColor:(UIColor \*)color{

CGRect rect = CGRectMake(0.0f, 0.0f, 1.0f, 1.0f);

UIGraphicsBeginImageContext(rect.size);

CGContextRef context = UIGraphicsGetCurrentContext();

CGContextSetFillColorWithColor(context, [color CGColor]);

CGContextFillRect(context, rect);

UIImage \*image = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();

return image;

}

@end

#import <MBProgressHUD/MBProgressHUD.h>

// 统一的显示时长

#define HudShowTime 1.5

NS\_ASSUME\_NONNULL\_BEGIN

@interface MBProgressHUD (PT)

#pragma mark 在window上显示hud

+ (void)showMessage:(NSString \*)message;

+ (void)showHUD;

+ (void)hideHUD;

@end

NS\_ASSUME\_NONNULL\_END

#import "MBProgressHUD+PT.h"

@implementation MBProgressHUD (PT)

+ (void)showMessage:(NSString \*)message{

MBProgressHUD \*hud = [MBProgressHUD showHUDAddedTo:KEY\_WINDOW animated:YES];

hud.label.text = message;

hud.mode = MBProgressHUDModeText;

// 隐藏时候从父控件中移除

hud.removeFromSuperViewOnHide = YES;

// 指定时间之后再消失

[hud hideAnimated:YES afterDelay:HudShowTime];

}

+ (void)showHUD{

[MBProgressHUD showHUDAddedTo:KEY\_WINDOW animated:YES];

}

+ (void)hideHUD{

[MBProgressHUD hideHUDForView:KEY\_WINDOW animated:YES];

}

@end

#import <Foundation/Foundation.h>

@interface NSArray (Unicode)

@end

#import "NSArray+Unicode.h"

#import "NSObject+SafeSwizzle.h"

#import "NSString+Unicode.h"

@implementation NSArray (Unicode)

+ (void)load{

static dispatch\_once\_t onceToken;

dispatch\_once(&onceToken, ^{

[self exchangeInstance:[self class] selector:@selector(description) withSwizzledSelector:@selector(chinese\_description)];

[self exchangeInstance:[self class] selector:@selector(descriptionWithLocale:) withSwizzledSelector:@selector(chinese\_descriptionWithLocale:)];

[self exchangeInstance:[self class] selector:@selector(descriptionWithLocale:indent:) withSwizzledSelector:@selector(chinese\_descriptionWithLocale:indent:)];

});

}

- (NSString \*)chinese\_description{

return [[self chinese\_description] stringByReplaceUnicode];

}

- (NSString \*)chinese\_descriptionWithLocale:(nullable id)locale{

return [[self chinese\_descriptionWithLocale:locale] stringByReplaceUnicode];

}

- (NSString \*)chinese\_descriptionWithLocale:(nullable id)locale indent:(NSUInteger)level{

return [[self chinese\_descriptionWithLocale:locale indent:level] stringByReplaceUnicode];

}

@end

#import <Foundation/Foundation.h>

@interface NSDictionary (Unicode)

@end

#import "NSDictionary+Unicode.h"

#import "NSObject+SafeSwizzle.h"

#import "NSString+Unicode.h"

@implementation NSDictionary (Unicode)

+ (void)load{

static dispatch\_once\_t onceToken;

dispatch\_once(&onceToken, ^{

[self exchangeInstance:[self class] selector:@selector(description) withSwizzledSelector:@selector(chinese\_description)];

[self exchangeInstance:[self class] selector:@selector(descriptionWithLocale:) withSwizzledSelector:@selector(chinese\_descriptionWithLocale:)];

[self exchangeInstance:[self class] selector:@selector(descriptionWithLocale:indent:) withSwizzledSelector:@selector(chinese\_descriptionWithLocale:indent:)];

});

}

- (NSString \*)chinese\_description{

return [[self chinese\_description] stringByReplaceUnicode];

}

- (NSString \*)chinese\_descriptionWithLocale:(nullable id)locale{

return [[self chinese\_descriptionWithLocale:locale] stringByReplaceUnicode];

}

- (NSString \*)chinese\_descriptionWithLocale:(nullable id)locale indent:(NSUInteger)level{

return [[self chinese\_descriptionWithLocale:locale indent:level] stringByReplaceUnicode];

}

@end

#import <Foundation/Foundation.h>

#import <objc/runtime.h>

@interface NSObject (SafeSwizzle)

//交换对象方法

//+ (BOOL)swizzleSelector:(SEL)originalSelector withSwizzledSelector:(SEL)swizzledSelector;

+ (BOOL)exchangeInstance:(Class)class selector:(SEL)originalSelector withSwizzledSelector: (SEL)swizzledSelector;

@end

#import "NSObject+SafeSwizzle.h"

@implementation NSObject (SafeSwizzle)

+ (BOOL)swizzleSelector:(SEL)originalSelector withSwizzledSelector:(SEL)swizzledSelector {

Class class = [self class];

Method originalMethod = class\_getInstanceMethod(class, originalSelector);

Method swizzledMethod = class\_getInstanceMethod(class, swizzledSelector);

if(!originalMethod || !swizzledMethod){

return NO;

}

// 若已经存在，则添加会失败

BOOL didAddMethod = class\_addMethod(class,

originalSelector,

method\_getImplementation(swizzledMethod),

method\_getTypeEncoding(swizzledMethod));

// 若原来的方法并不存在，则添加即可

if (didAddMethod) {

class\_replaceMethod(class,

swizzledSelector,

method\_getImplementation(originalMethod),

method\_getTypeEncoding(originalMethod));

} else {

method\_exchangeImplementations(originalMethod, swizzledMethod);

}

return YES;

}

+ (BOOL)exchangeInstance:(Class)class selector:(SEL)originalSelector withSwizzledSelector: (SEL)swizzledSelector{

Method originalMethod = class\_getInstanceMethod(class, originalSelector);

Method swizzledMethod = class\_getInstanceMethod(class, swizzledSelector);

if(!originalMethod || !swizzledMethod){

return NO;

}

// 若已经存在，则添加会失败

BOOL didAddMethod = class\_addMethod(class,

originalSelector,

method\_getImplementation(swizzledMethod),

method\_getTypeEncoding(swizzledMethod));

// 若原来的方法并不存在，则添加即可

if (didAddMethod) {

class\_replaceMethod(class,

swizzledSelector,

method\_getImplementation(originalMethod),

method\_getTypeEncoding(originalMethod));

} else {

method\_exchangeImplementations(originalMethod, swizzledMethod);

}

return YES;

}

@end

#import <Foundation/Foundation.h>

@interface NSString (Unicode)

- (NSString\*)stringByReplaceUnicode;

@end

#import "NSString+Unicode.h"

#import "NSObject+SafeSwizzle.h"

@implementation NSString (Unicode)

+ (void)load{

static dispatch\_once\_t onceToken;

dispatch\_once(&onceToken, ^{

[self exchangeInstance:[self class] selector:@selector(description) withSwizzledSelector:@selector(chinese\_description)];

[self exchangeInstance:[self class] selector:@selector(descriptionWithLocale:) withSwizzledSelector:@selector(chinese\_descriptionWithLocale:)];

[self exchangeInstance:[self class] selector:@selector(descriptionWithLocale:indent:) withSwizzledSelector:@selector(chinese\_descriptionWithLocale:indent:)];

});

}

- (NSString \*)chinese\_description{

return [[self chinese\_description] stringByReplaceUnicode];

}

- (NSString \*)chinese\_descriptionWithLocale:(nullable id)locale{

return [[self chinese\_descriptionWithLocale:locale] stringByReplaceUnicode];

}

- (NSString \*)chinese\_descriptionWithLocale:(nullable id)locale indent:(NSUInteger)level{

return [[self chinese\_descriptionWithLocale:locale indent:level] stringByReplaceUnicode];

}

- (NSString\*)stringByReplaceUnicode{

NSMutableString\*convertedString = [self mutableCopy];

[convertedString replaceOccurrencesOfString:@"\\U" withString:@"\\u" options:0 range:NSMakeRange(0, convertedString.length)];

CFStringRef transform =CFSTR("Any-Hex/Java");

CFStringTransform((\_\_bridge CFMutableStringRef)convertedString,NULL, transform,YES);

return convertedString;

}

@end

#import <UIKit/UIKit.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTBaseViewController : UIViewController

@end

NS\_ASSUME\_NONNULL\_END

#import "PTBaseViewController.h"

@interface PTBaseViewController ()

@end

@implementation PTBaseViewController

- (void)viewWillAppear:(BOOL)animated{

[super viewWillAppear:animated];

// if (![NSStringFromClass([super class]) isEqualToString:@"PTBaseViewController"]) {

// [[PTEventRecord shareManager] addEventWithType:@"enter" Name:[NSString stringWithFormat:@"page\_%@", NSStringFromClass([super class])] Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

// }

}

- (void)viewWillDisappear:(BOOL)animated{

[super viewWillDisappear:animated];

// if (![NSStringFromClass([super class]) isEqualToString:@"PTBaseViewController"]) {

// [[PTEventRecord shareManager] addEventWithType:@"leave" Name:[NSString stringWithFormat:@"page\_%@", NSStringFromClass([super class])] Extras:[NSString stringWithFormat:@"函数名:%s行号:%d", \_\_FUNCTION\_\_, \_\_LINE\_\_]];

// }

}

- (void)viewDidLoad {

[super viewDidLoad];

// Do any additional setup after loading the view.

self.edgesForExtendedLayout = UIRectEdgeNone;

self.view.backgroundColor = [UIColor whiteColor];

}

- (void)dealloc {

NSLog(@"dealloc: %@", self);

}

- (UIStatusBarStyle)preferredStatusBarStyle {

// 如果app绝大多数页面要设置黑色样式，可以不写此方法，因为默认样式就是黑色的。

// return UIStatusBarStyleDefault;

// 白色样式

return UIStatusBarStyleLightContent;

}

@end

#import <UIKit/UIKit.h>

NS\_ASSUME\_NONNULL\_BEGIN

@interface PTBaseNavigationController : UINavigationController

@end

NS\_ASSUME\_NONNULL\_END

#import "PTBaseNavigationController.h"

#import "PTImageTool.h"

@interface PTBaseNavigationController ()

@end

@implementation PTBaseNavigationController

- (void)viewDidLoad {

[super viewDidLoad];

// Do any additional setup after loading the view.

self.navigationBar.translucent = NO;

//通过设置shadowImage为 navigationBar设置分割线颜色

[self.navigationBar setBackgroundImage:[PTImageTool imageWithColor:[UIColor clearColor]] forBarPosition:UIBarPositionAny barMetrics:UIBarMetricsDefault];

[self.navigationBar setShadowImage:[UIImage new]];

UIView \*line = [[UIView alloc] initWithFrame:CGRectMake(0, self.navigationBar.height - 0.33, SCREEN\_WIDTH, 0.33)];

line.backgroundColor = UIColor(0x000000, 0.1);

[self.navigationBar addSubview:line];

}

@end