iOS媒体采集 by WeiJi

媒体采集

媒体采集，在这个流程中，这里会存在几个对象：

- AVCaptureDevice。这里代表抽象的硬件设备。
- AVCaptureInput。这里代表输入设备（可以是它的子类），它配置抽象硬件设备的ports。
- AVCaptureOutput。它代表输出数据，管理着输出到一个movie或者图像。
- AVCaptureSession。它是input和output的桥梁。它协调着intput到output的数据传输。

上述对象的关系：

有很多Device的input，也有很多数据类型的Output，都通过一个Capture Session来控制进行传输。也即：CaptureDevice适配AVCaptureInput，通过Session来输入到AVCaptureOutput中。这样也就达到了从设备到文件等持久化传输的目的（如从相机设备采集图像到UIImage中）。

特别注意，这里的关系是可以通过一个Capture Session来同时控制多个设备的输入和输出。

那么存在一个问题了：视频输入(input)就对应视频的输出(output)，而音频输入就应对应音频的输出，因而需要建立对应的Connections，来各自连接它们。而这样的连接对象，是由AVCaptureSession来持有的，这个对象叫AVCaptureConnection。

在一个AVCaptureConnection中，这里维持着对应的数据传输输入到数据输出的过程（detail过程）。这里，AVCaptureInput或其子类对象包含着各种input port，通过各种input port，我们的AVCaptureOutput可以获取到相应的数据。

一个AVCaptureConnection可以控制input到output的数据传输。

当使用addInput:或addOutput:,Connection在所有兼容的输入和输出之间自动形成。手动添Connection是在在添加一个输入或输出时没有形成连接的时候。

AVCaptureDevice

InputDevice即是对硬件的抽象，一对一的。一个AVCaptureDevice对象，对应一个实际的硬件设备。

那么显然，我们可以通过AVCaptureDevice的类方法devices或devicesWithMediaType去获取全部或局部设备列表。（当然也可以检测相应的设备是否可以使用，这里注意有设备抢占问题，

当前是否可用）

CaptureInput的构建和添加到Session中的方法

```
/*创建并配置输入设备*/

AVCaptureDevice *device = [AVCaptureDevice defaultDeviceWithMediaType:AVM
ediaTypeVideo];
NSError *error = nil;
AVCaptureDeviceInput *input = [AVCaptureDeviceInput deviceInputWithDevice
:device error:&error];
if (error){
    //handle the failure.
}

//添加input到session的模式是(检查可否添加到session，然后根据情况添加或者不添加):
AVCaptureSession *captureSession = [[AVCaptureSession alloc] init];
if ([captureSession canAddInput:input]) {
    [captureSession addInput:captureDeviceInput];
}
else{
    //handle the failure.
}
```

output的分类和使用

什么类型的输出,决定了什么输出数据的类型。在ios中，分为MovieFile、VideoData、AudioFile、AudioData、StillImage和Metadata几种output，使用方式类似，只是范围不同。另外，它们都继承于AVCaptureOutput,其自身是抽象类。

第一个是输出成movie文件，第二个适用于逐个Frame的处理，第三个适用于输出成声音文件，第四个适用于声音采集，第五个是still image(静态图像<拍照>)相关，最后一个用于输出元数据（面部识别、条码等）。

他们的添加方式都是使用session的addOutput方法。

二维码的扫描的实现

定义的属性

```
@property (nonatomic, strong)AVCaptureDevice *device;
@property (nonatomic, strong)AVCaptureDeviceInput *input;
@property (nonatomic, strong)AVCaptureSession *session;
@property (nonatomic, strong)AVCaptureMetadataOutput *output;

@property (nonatomic, strong)UIImageView *animationImageView;
@property (nonatomic, strong)NSTimer *timer;
@property (nonatomic,  strong)UIView *preView;
```

首先在`-(void)viewDidAppear:(BOOL)animated`方法中启动二维码扑捉

```
-(void)viewDidAppear:(BOOL)animated{
    [super viewDidAppear:animated];
    [self startReading];
}

-(void)viewWillDisappear:(BOOL)animated{
    [super viewWillDisappear:animated];
    [self stopRead];
}
```

接下来是具体实现`starrReading`方法

```
- (BOOL)startReading {
    //功能实现
}

//停止二维码扫描的方法
-(void)stopRead{
    [self.session stopRunning];
}
```

1.获得扫描设备：（摄像头）。

```
AVCaptureDevice *device = [AVCaptureDevice defaultDeviceWithMediaType:AVM
ediaTypeVideo];
```

2.生成连接设备和session的input。

```
NSError *error;
AVCaptureDeviceInput *input = [AVCaptureDeviceInput deviceInputWithDevice
:device error:&error];
if(error){
    NSLog(@"%@", error.localizedDescription);
}
```

3.AVCaptureOutput。

在ios6之后，新增Output的实现类AVCaptureMetadataOutput。

```
AVCaptureMetadataOutput * output = [[AVCaptureMetadataOutput alloc] init]
;
dispatch_queue_t dispatchQueue;
dispatchQueue = dispatch_queue_create("hellp", NULL);
[output setMetadataObjectsDelegate:self queue:dispatchQueue];
```

4.AVCaptureSession。

```
AVCaptureSession *session  = [[AVCaptureSession alloc] init];
[session addInput:input];
[session addOutput:output];
//注意：设置AVMetadataMachineReadableCodeObject的可输出的元数据类型，一定要讲out
put添加到session中后，再设置；
[output setMetadataObjectTypes:[NSArray arrayWithObjects:
                                                        AVMetadataObjectTypeEA
N13Code,
                                                        AVMetadataObjectTypeCo
de39Code,
                                                        AVMetadataObjectTypeEA
N8Code,
                                                        AVMetadataObjectTypeCo
de128Code,
                                                        AVMetadataObjectTypeQR
Code, nil]];
```

5.AVCaptureVideoPreviewLayer,CALayer的子类,可以预览device扑捉的画面。

```
AVCaptureVideoPreviewLayer *videoPreviewLayer = [[AVCaptureVideoPreviewLa
yer alloc] initWithSession:_session];

[layer setVideoGravity:AVLayerVideoGravityResizeAspectFill];
[layer setFrame:self.view.layer.bounds];
```

6.开启服务

```
[self.session startRunning];
```

7.代理服务,注意该方法的调用是在串行队列中

```
- (void)captureOutput:(AVCaptureOutput *)captureOutput didOutputMetadataO
bjects:(NSArray *)metadataObjects fromConnection:(AVCaptureConnection *)c
onnection{
    NSLog(@"%@", [[metadataObjects firstObject] stringValue]);
    [self performSelectorOnMainThread:@selector(close) withObject:nil wai
tUntilDone:YES];

}
```

UI的实现

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.
//    self.edgesForExtendedLayout = UIRectEdgeNone;
    self.navigationController.navigationBar.translucent = NO;

    //添加展示View
    self.preView = [[UIView alloc] initWithFrame:self.view.bounds];
    [self.view addSubview:self.preView];
    UIBarButtonItem *close = [[UIBarButtonItem alloc] initWithTitle:@"关闭
" style:UIBarButtonItemStyleDone target:self action:@selector(close)];
    self.navigationItem.leftBarButtonItem = close;
    self.title = @"二维码";
    self.navigationController.navigationBar.tintColor = [UIColor orangeCo
lor];

    //添加边框
    UIImageView *boundImage = [[UIImageView alloc] initWithFrame:CGRectMa
ke(40, 20, 240, 240)];
    UIImage * image = [UIImage imageNamed:@"qrcode_border"];
    image = [image resizableImageWithCapInsets:UIEdgeInsetsMake(25, 25, 2
6, 26)];
    boundImage.image = image;
    [self.view addSubview:boundImage];
    boundImage.clipsToBounds = YES;

    //添加移动的Image
    self.animationImageView = [[UIImageView alloc] initWithFrame:boundIma
ge.bounds];
    [self.animationImageView setImage:[UIImage imageNamed:@"qrcode_scanli
ne_qrcode"]];
    [boundImage addSubview:self.animationImageView];

    通过Timer移动ImageView
    self.timer = [NSTimer scheduledTimerWithTimeInterval:0.03f target:sel
f selector:@selector(changeImage:) userInfo:nil repeats:YES];


}
```

在timer触发事件中移动图片

```
-(void)changeImage:(NSTimer *)timer{
    self.animationImageView.frame = CGRectOffset(self.animationImageView.
frame, 0, 5);
    if (self.animationImageView.frame.origin.y >= self.animationImageView
.frame.size.height - 120) {
        self.animationImageView.frame = CGRectMake(0, -self.animationImag
eView.frame.size.height , self.animationImageView.frame.size.width, self.
animationImageView.frame.size.height);
    }

}
```