短视频分享

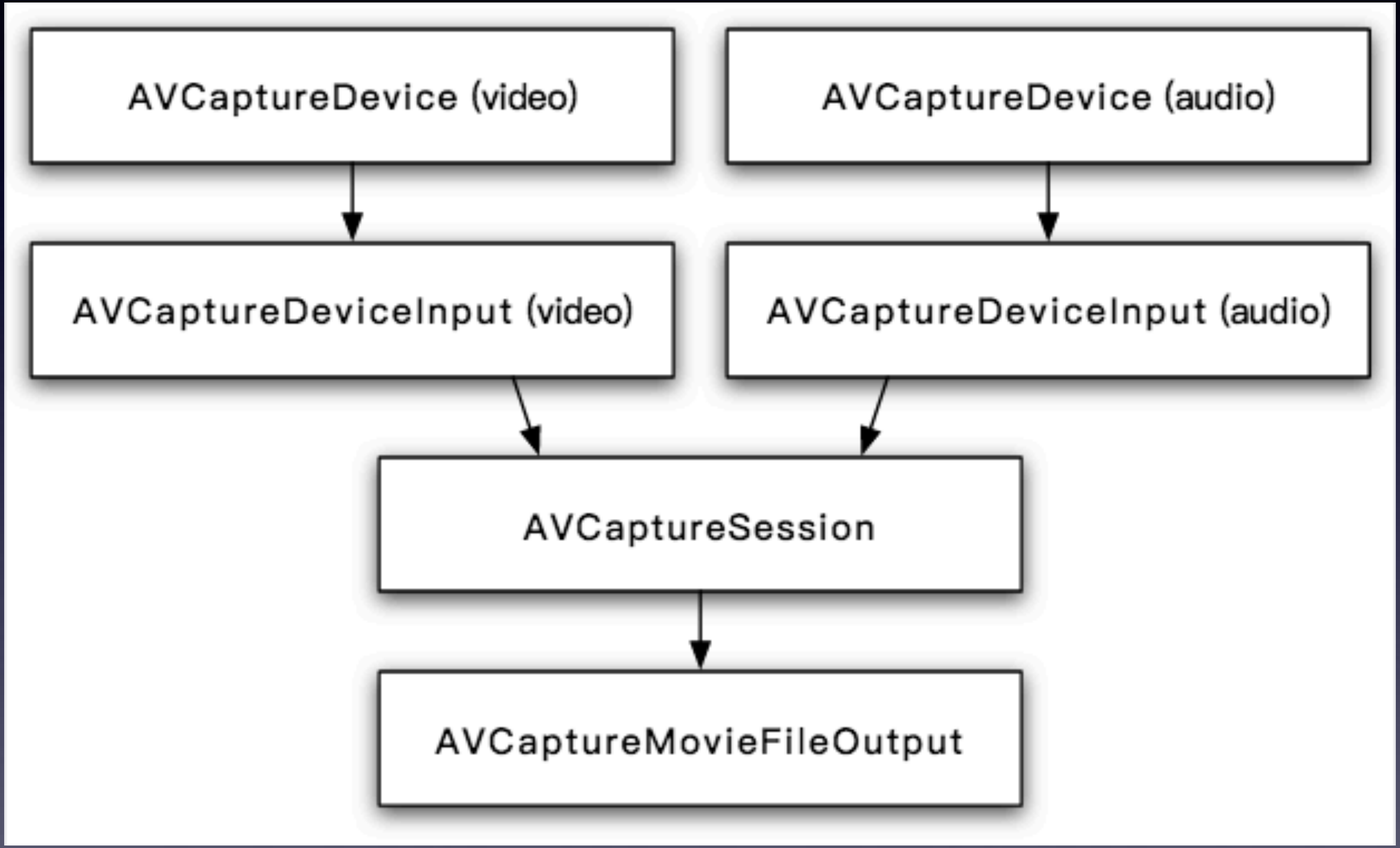
nate

功能概览

- 录制短视频（滤镜）
- 加水印压缩
- 发布（存草稿）

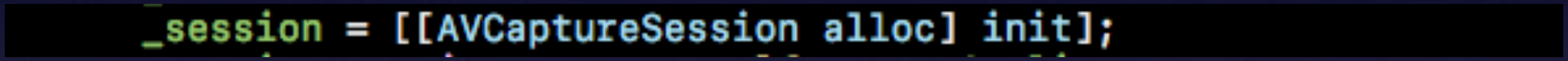
$$\begin{tikzcd}
\text{原生相机} \arrow[d] \\
\text{TUSDK相机} \arrow[d] \\
\text{GPUImage相机}
\end{tikzcd}$$

自定义相机

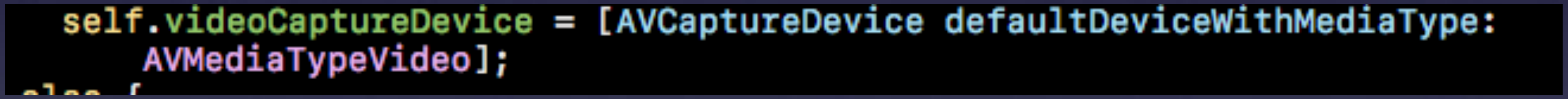

AVCaptureDevice (video)
AVCaptureDevice (audio)
AVCaptureDeviceInput (video)
AVCaptureDeviceInput (audio)
AVCaptureSession
AVCaptureMovieFileOutput

具体代码

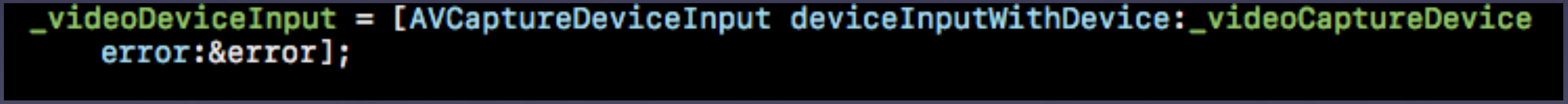
```
@property (strong, nonatomic) AVCaptureSession *session;
@property (strong, nonatomic) AVCaptureDevice *videoCaptureDevice;
@property (strong, nonatomic) AVCaptureDevice *audioCaptureDevice;
@property (strong, nonatomic) AVCaptureDeviceInput *videoDeviceInput;
@property (strong, nonatomic) AVCaptureDeviceInput *audioDeviceInput;
@property (strong, nonatomic) AVCaptureVideoPreviewLayer *captureVideoPreviewLayer;
```



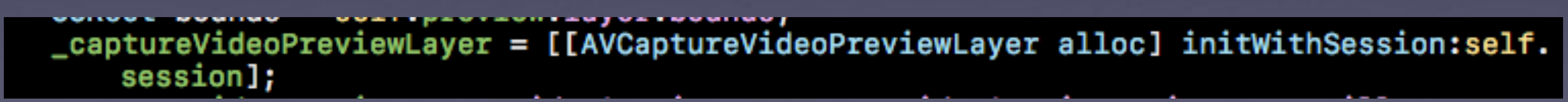
```
    _session = [[AVCaptureSession alloc] init];
```



```
  self.videoCaptureDevice = [AVCaptureDevice defaultDeviceWithMediaType:
      AVMediaTypeVideo];
```

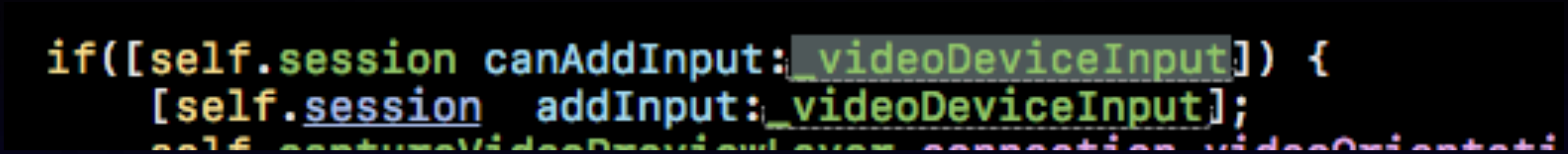


```
_videoDeviceInput = [AVCaptureDeviceInput deviceInputWithDevice:_videoCaptureDevice
    error:&error];
```



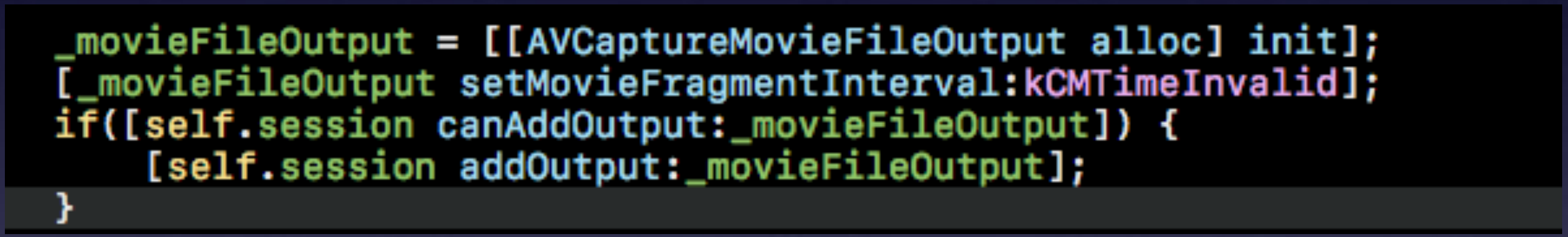
```
  _captureVideoPreviewLayer = [[AVCaptureVideoPreviewLayer alloc] initWithSession:self.
      session];
```

添加输入

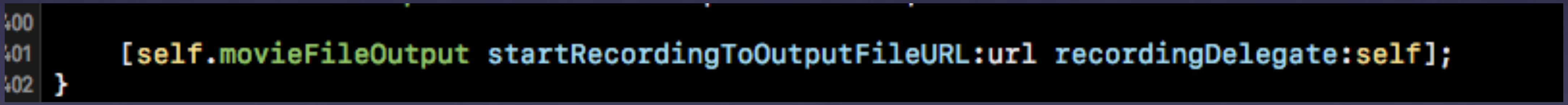


```
if([self.session canAddInput:_videoDeviceInput]) {
    [self.session  addInput:_videoDeviceInput];
```

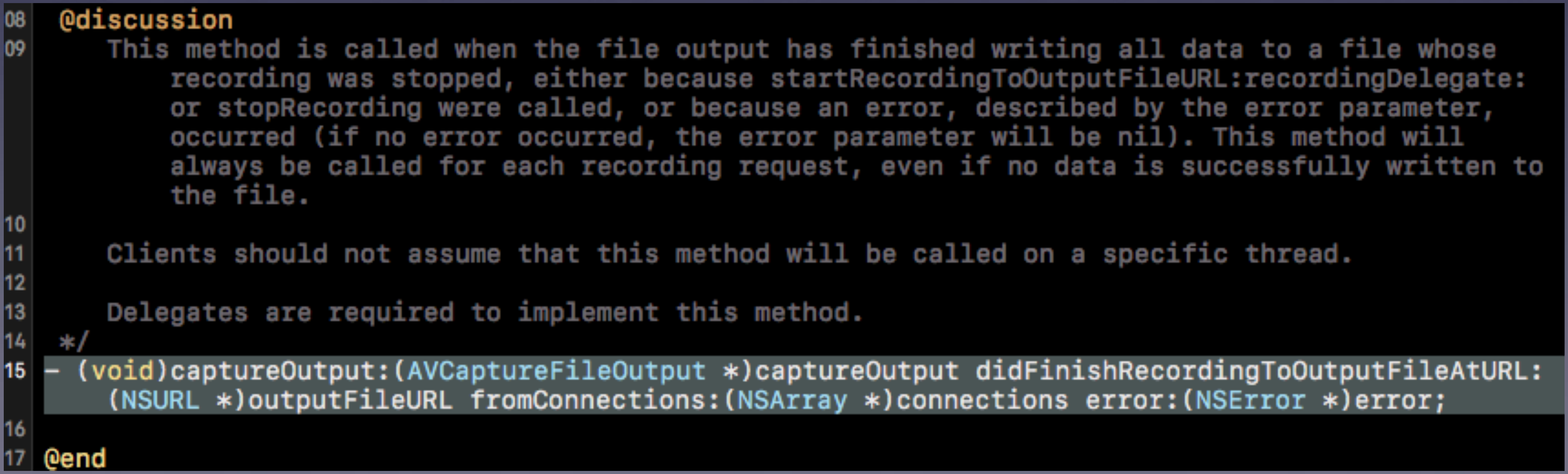
添加输出



```
_movieFileOutput = [[AVCaptureMovieFileOutput alloc] init];
[_movieFileOutput setMovieFragmentInterval:kCMTimeInvalid];
if([self.session canAddOutput:_movieFileOutput]) {
    [self.session addOutput:_movieFileOutput];
}
```

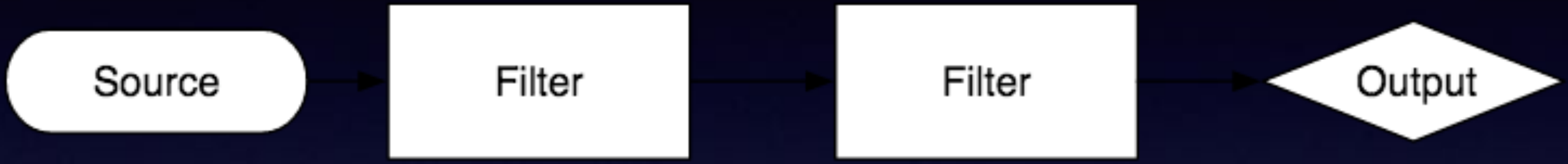


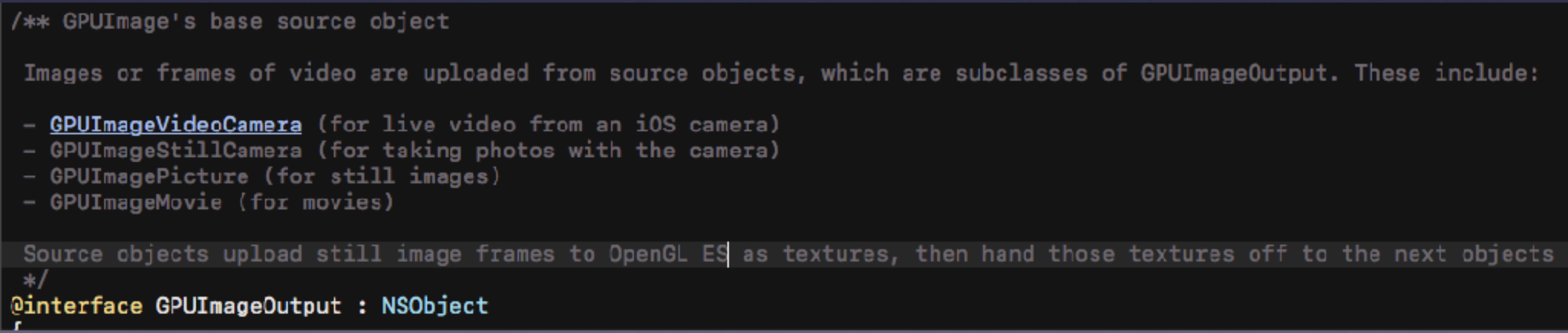
```
    [self.movieFileOutput startRecordingToOutputFileURL:url recordingDelegate:self];
}
```



```
 @discussion
    This method is called when the file output has finished writing all data to a file whose
        recording was stopped, either because startRecordingToOutputFileURL:recordingDelegate:
        or stopRecording were called, or because an error, described by the error parameter,
        occurred (if no error occurred, the error parameter will be nil). This method will
        always be called for each recording request, even if no data is successfully written to
        the file.

    Clients should not assume that this method will be called on a specific thread.

    Delegates are required to implement this method.
 */
- (void)captureOutput:(AVCaptureFileOutput *)captureOutput didFinishRecordingToOutputFileAtURL:
    (NSURL *)outputFileURL fromConnections:(NSArray *)connections error:(NSError *)error;

@end
```

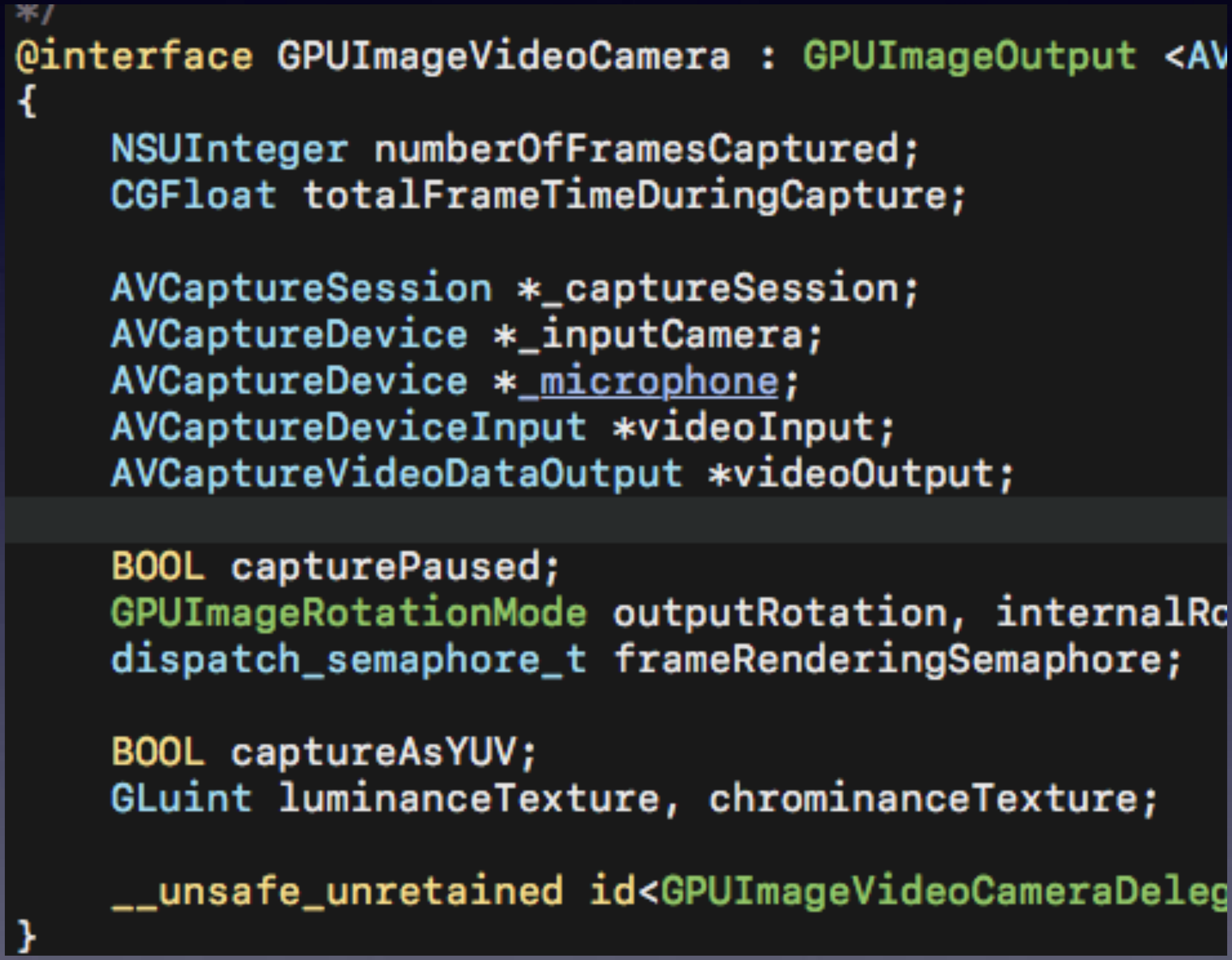
GPUImage





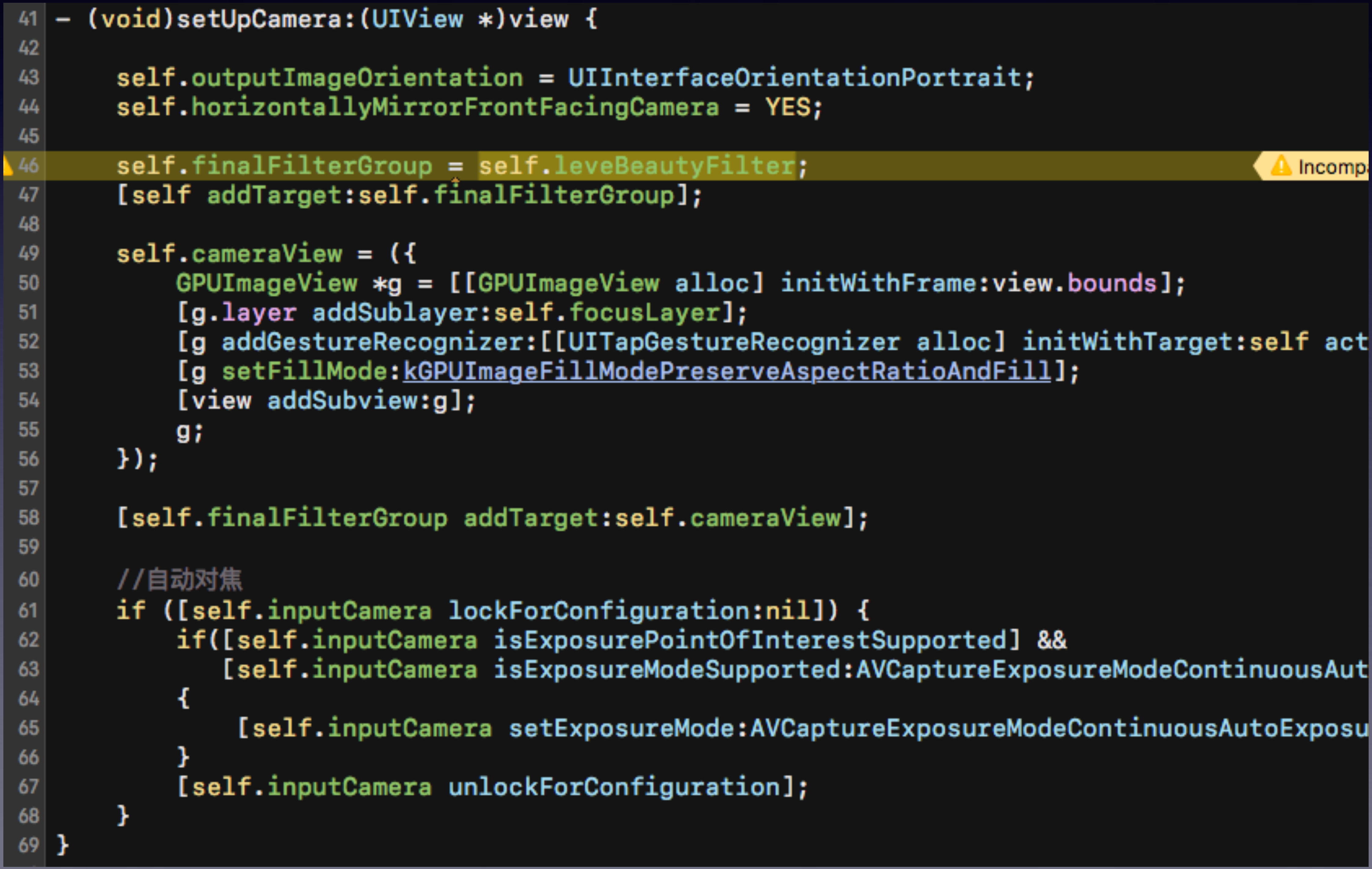
```
/** GPUImage's base source object

 Images or frames of video are uploaded from source objects, which are subclasses of GPUImageOutput. These include:

 - GPUImageVideoCamera (for live video from an iOS camera)
 - GPUImageStillCamera (for taking photos with the camera)
 - GPUImagePicture (for still images)
 - GPUImageMovie (for movies)

 Source objects upload still image frames to OpenGL ES as textures, then hand those textures off to the next objects
 */
@interface GPUImageOutput : NSObject
```

GPUImageVideoCamera



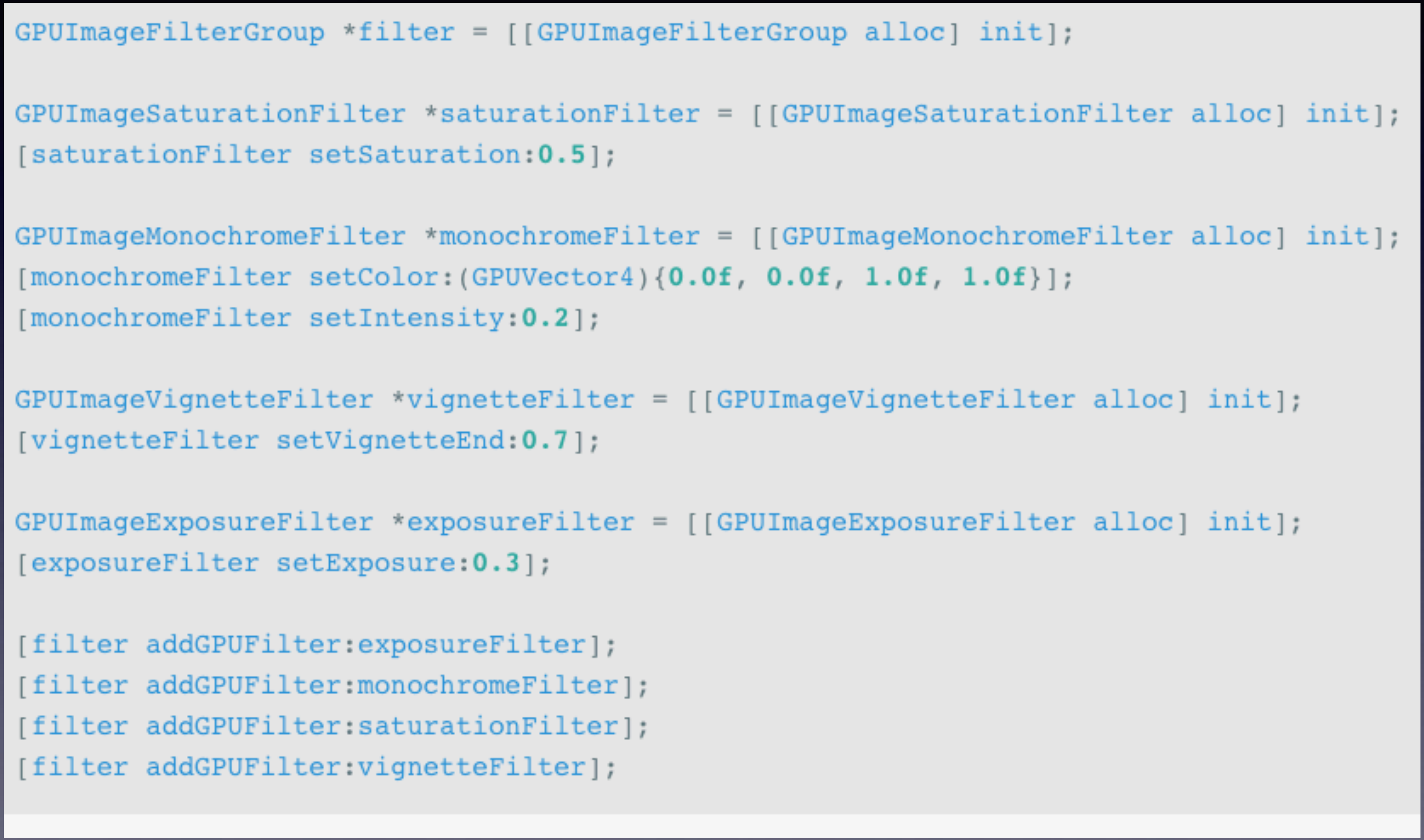
```
*/
@interface GPUImageVideoCamera : GPUImageOutput <AV
{
    NSUInteger numberOfFramesCaptured;
    CGFloat totalFrameTimeDuringCapture;

    AVCaptureSession *_captureSession;
    AVCaptureDevice *_inputCamera;
    AVCaptureDevice *_microphone;
    AVCaptureDeviceInput *videoInput;
    AVCaptureVideoDataOutput *videoOutput;

    BOOL capturePaused;
    GPUImageRotationMode outputRotation, internalRo
    dispatch_semaphore_t frameRenderingSemaphore;

    BOOL captureAsYUV;
    GLuint luminanceTexture, chrominanceTexture;

    __unsafe_unretained id<GPUImageVideoCameraDeleg
}
```

PDVideoCamera



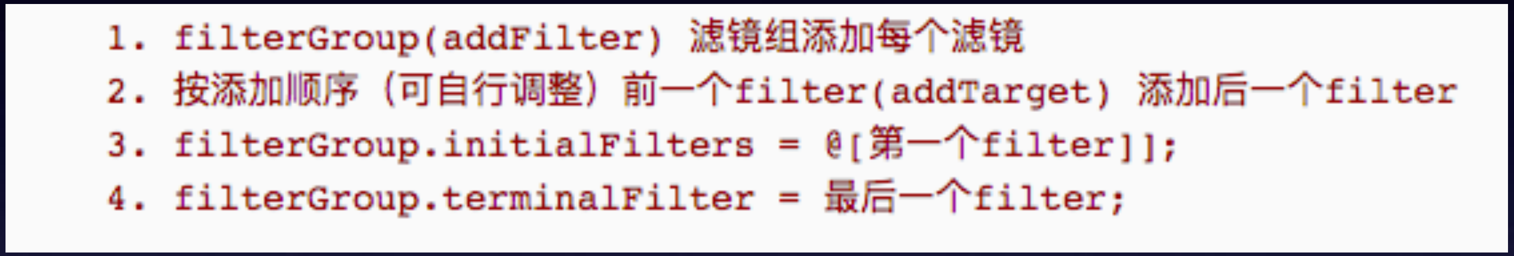
```
- (void)setUpCamera:(UIView *)view {

    self.outputImageOrientation = UIInterfaceOrientationPortrait;
    self.horizontallyMirrorFrontFacingCamera = YES;

    self.finalFilterGroup = self.leveBeautyFilter;
    [self addTarget:self.finalFilterGroup];

    self.cameraView = ({
        GPUImageView *g = [[GPUImageView alloc] initWithFrame:view.bounds];
        [g.layer addSublayer:self.focusLayer];
        [g addGestureRecognizer:[[UITapGestureRecognizer alloc] initWithTarget:self act
        [g setFillMode:kGPUImageFillModePreserveAspectRatioAndFill];
        [view addSubview:g];
        g;
    });

    [self.finalFilterGroup addTarget:self.cameraView];

    //自动对焦
    if ([self.inputCamera lockForConfiguration:nil]) {
        if([self.inputCamera isExposurePointOfInterestSupported] &&
           [self.inputCamera isExposureModeSupported:AVCaptureExposureModeContinuousAut
        {
            [self.inputCamera setExposureMode:AVCaptureExposureModeContinuousAutoExposu
        }
        [self.inputCamera unlockForConfiguration];
    }
}
```

Incomp

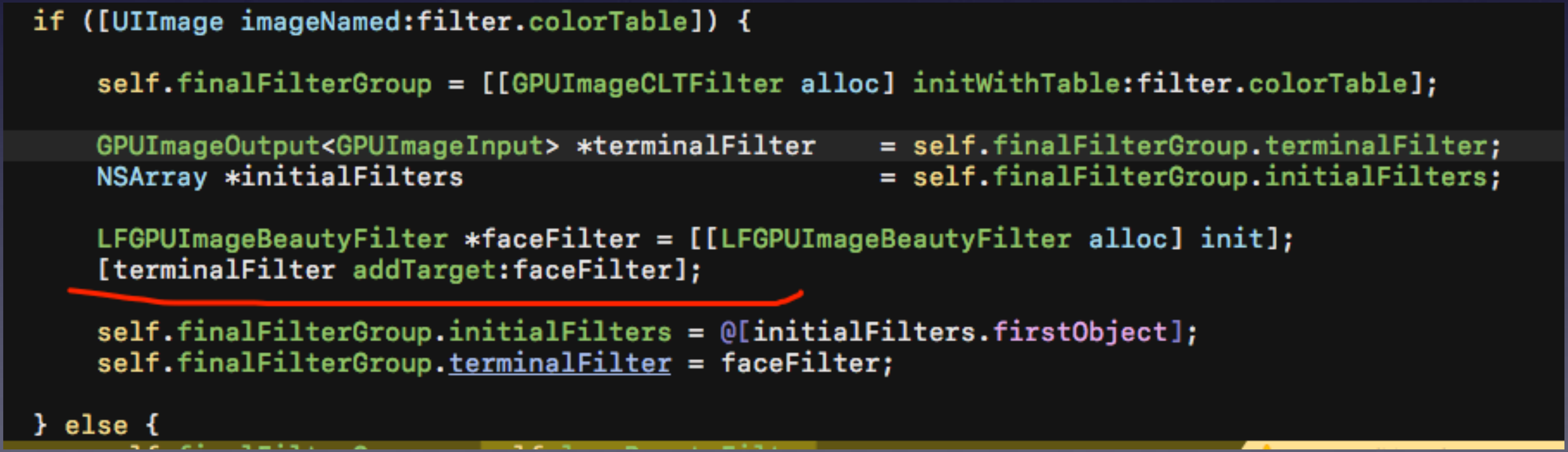


```
GPUImageFilterGroup *filter = [[GPUImageFilterGroup alloc] init];

GPUImageSaturationFilter *saturationFilter = [[GPUImageSaturationFilter alloc] init];
[saturationFilter setSaturation:0.5];

GPUImageMonochromeFilter *monochromeFilter = [[GPUImageMonochromeFilter alloc] init];
[monochromeFilter setColor:(GPUVector4){0.0f, 0.0f, 1.0f, 1.0f}];
[monochromeFilter setIntensity:0.2];

GPUImageVignetteFilter *vignetteFilter = [[GPUImageVignetteFilter alloc] init];
[vignetteFilter setVignetteEnd:0.7];

GPUImageExposureFilter *exposureFilter = [[GPUImageExposureFilter alloc] init];
[exposureFilter setExposure:0.3];

[filter addGPUFilter:exposureFilter];
[filter addGPUFilter:monochromeFilter];
[filter addGPUFilter:saturationFilter];
[filter addGPUFilter:vignetteFilter];
```

添加滤镜

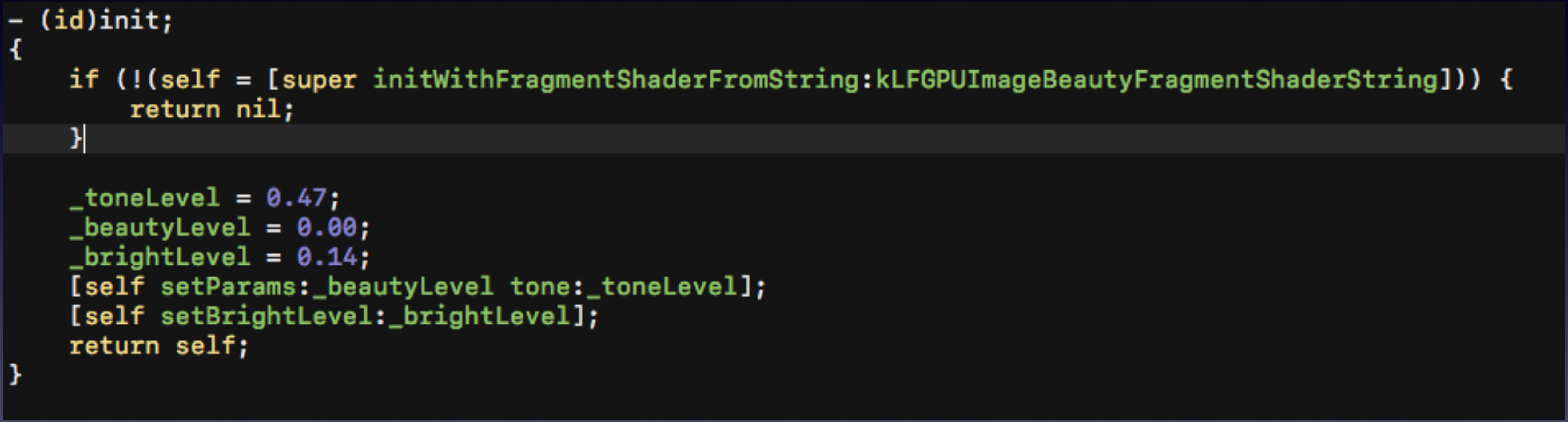


1. filterGroup(addFilter) 滤镜组添加每个滤镜
2. 按添加顺序（可自行调整）前一个filter(addTarget) 添加后一个filter
3. filterGroup.initialFilters = @[第一个filter]];
4. filterGroup.terminalFilter = 最后一个filter;



```
if ([UIImage imageNamed:filter.colorTable]) {

    self.finalFilterGroup = [[GPUImageCLTFilter alloc] initWithTable:filter.colorTable];

    GPUImageOutput<GPUImageInput> *terminalFilter    = self.finalFilterGroup.terminalFilter;
    NSArray *initialFilters                          = self.finalFilterGroup.initialFilters;

    LFGPUImageBeautyFilter *faceFilter = [[LFGPUImageBeautyFilter alloc] init];
    [terminalFilter addTarget:faceFilter];

    self.finalFilterGroup.initialFilters = @[initialFilters.firstObject];
    self.finalFilterGroup.terminalFilter = faceFilter;

} else {
```

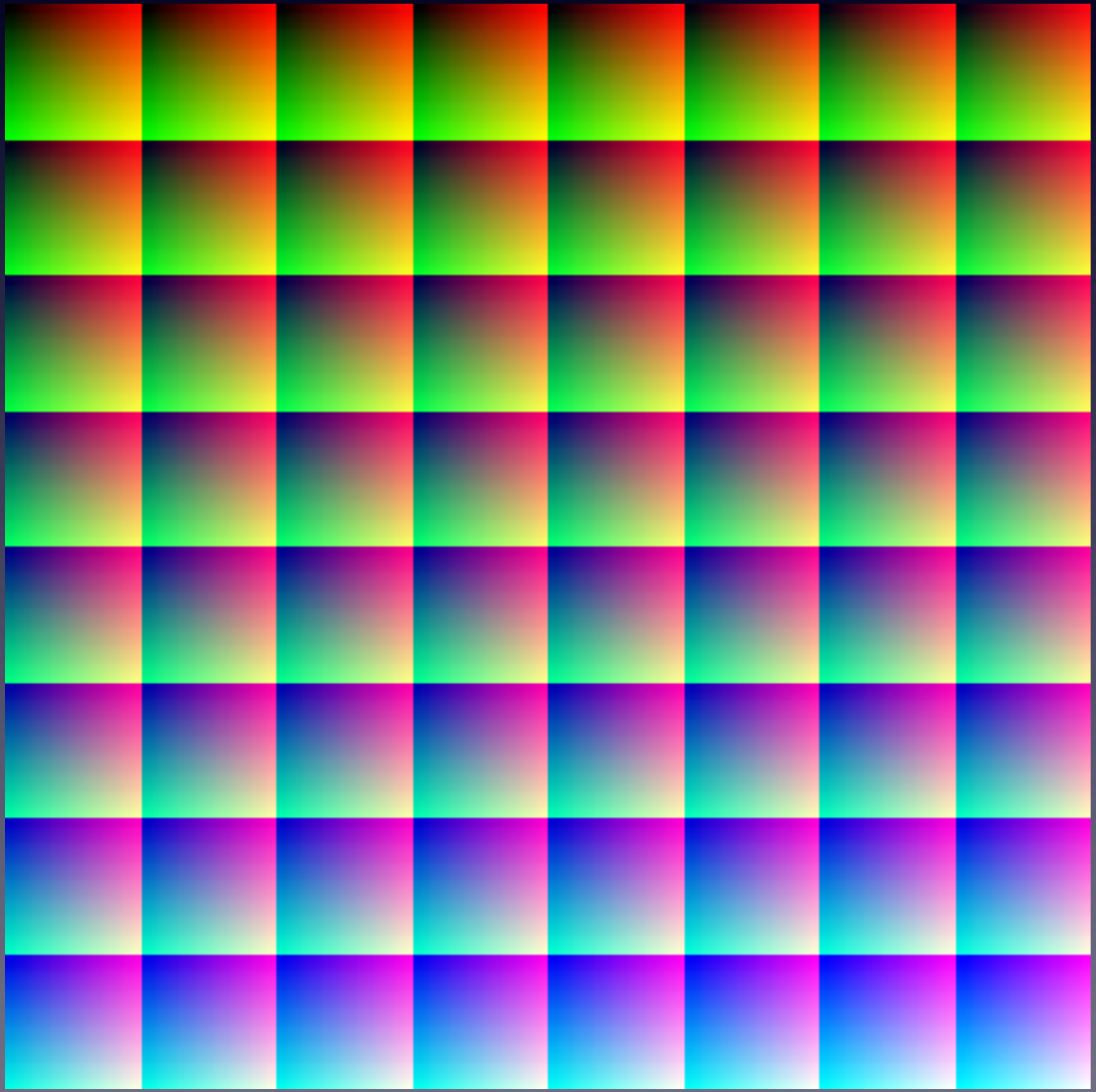
美颜滤镜

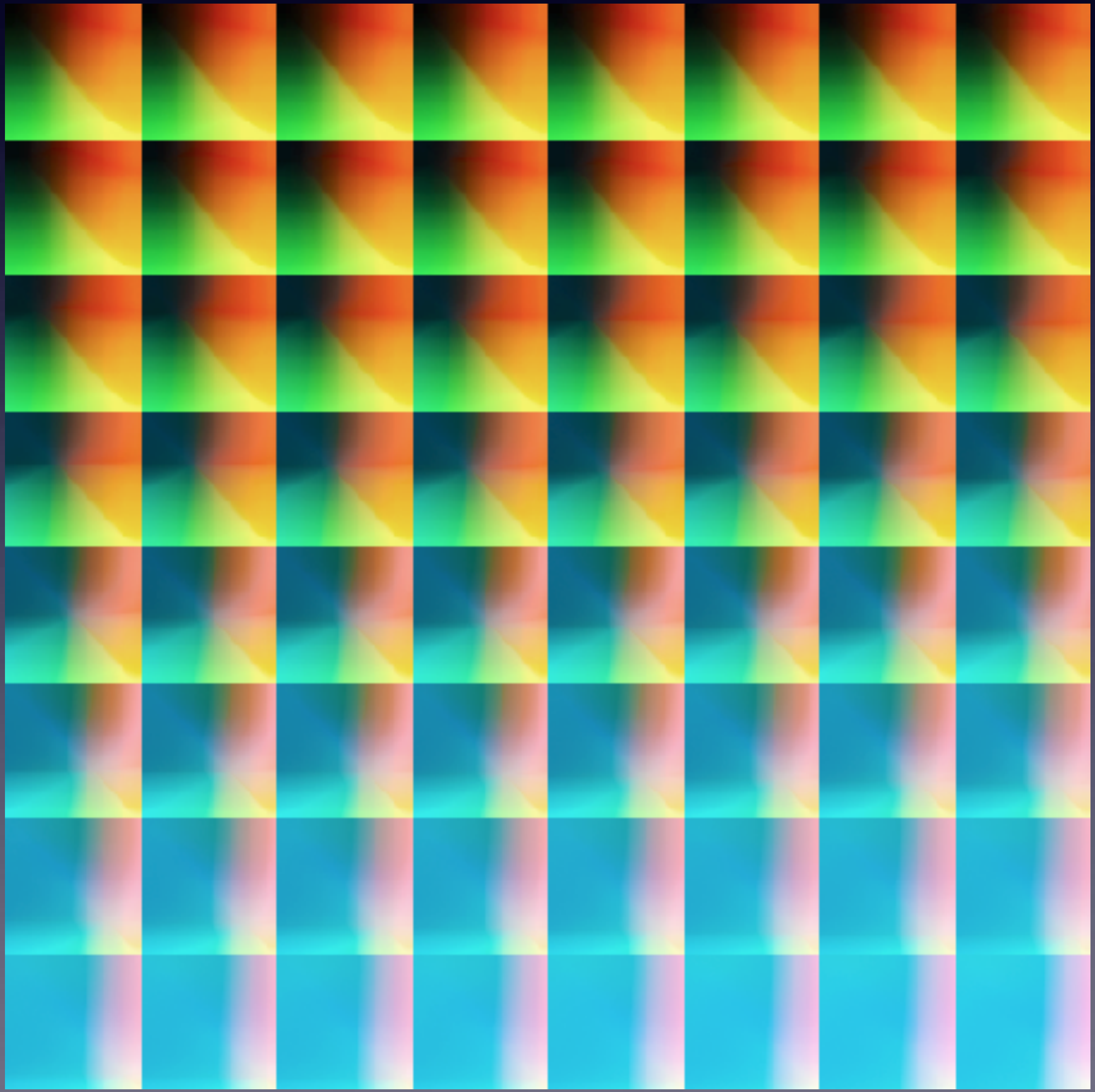


```
- (id)init;
{
    if (!(self = [super initWithFragmentShaderFromString:kLFGPUImageBeautyFragmentShaderString])) {
        return nil;
    }

    _toneLevel = 0.47;
    _beautyLevel = 0.00;
    _brightLevel = 0.14;
    [self setParams:_beautyLevel tone:_toneLevel];
    [self setBrightLevel:_brightLevel];
    return self;
}
```

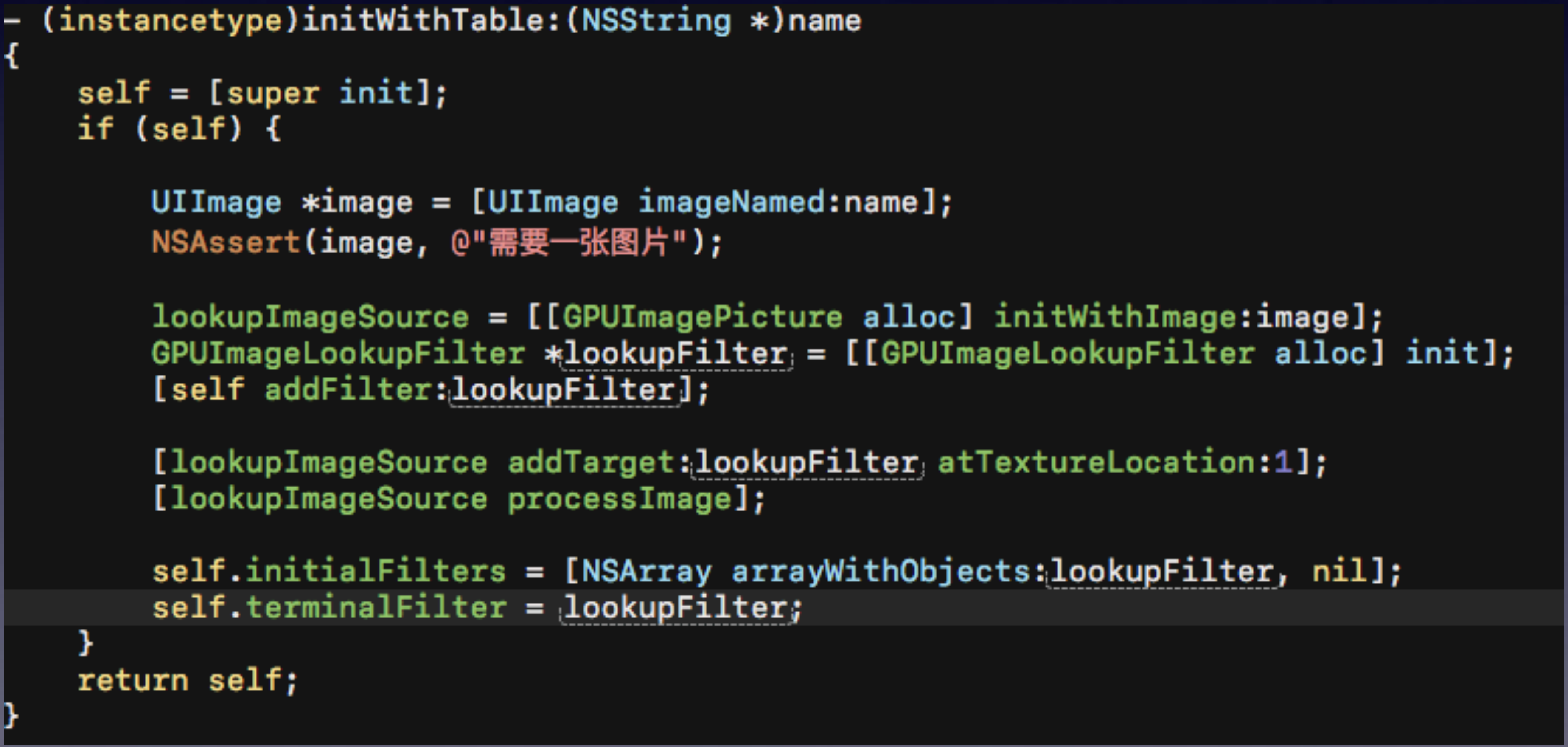
Lookup滤镜

转换



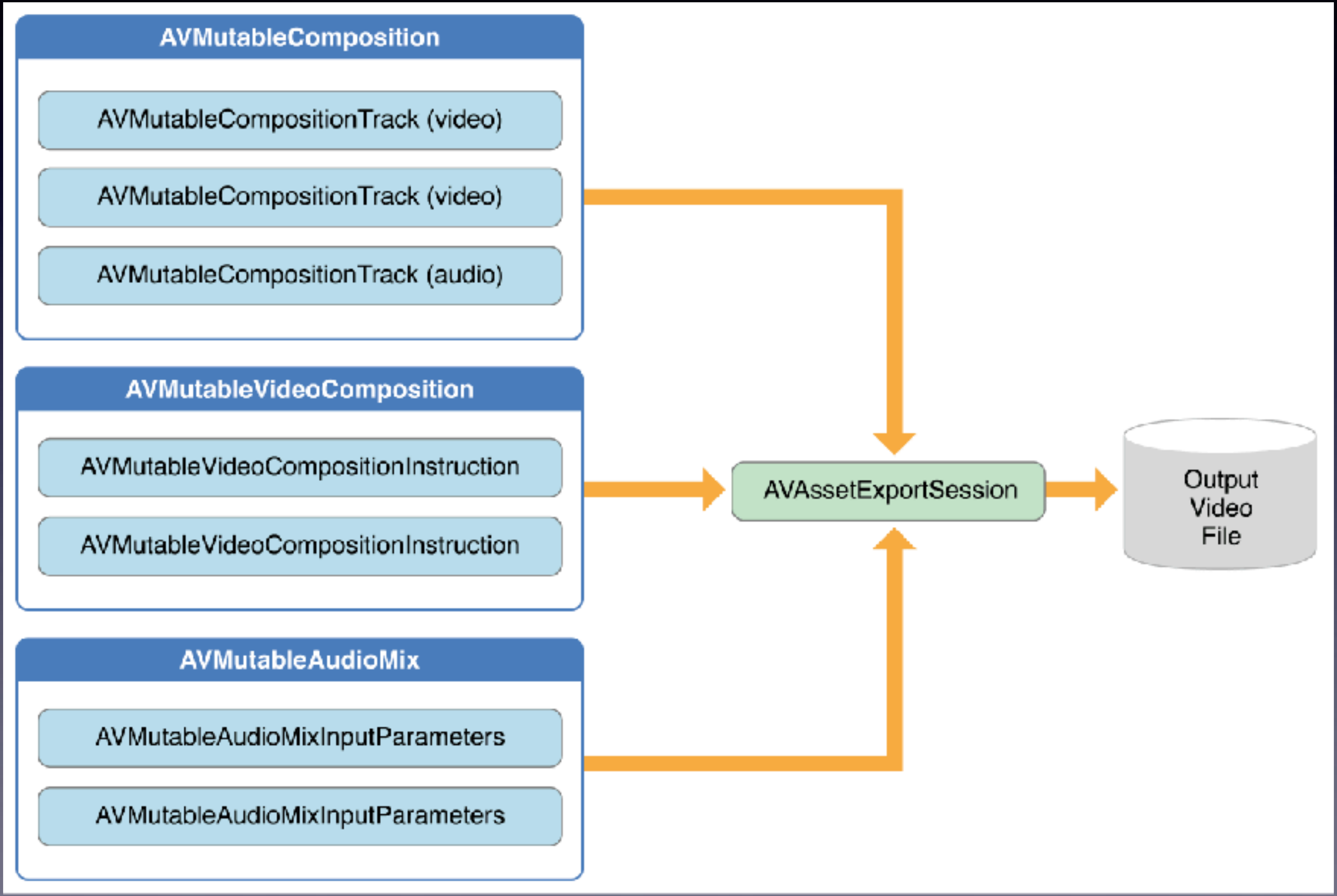


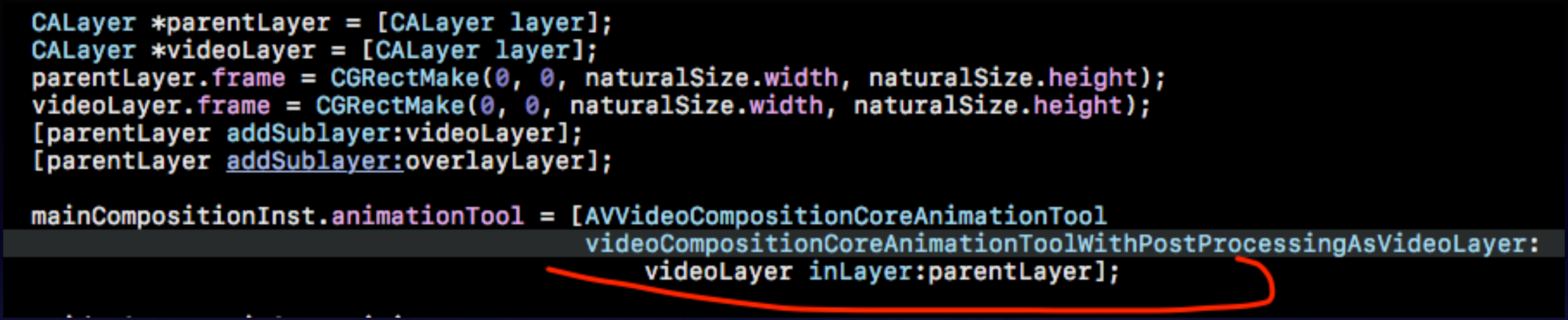
实现



```
- (instancetype)initWithTable:(NSString *)name
{
    self = [super init];
    if (self) {

        UIImage *image = [UIImage imageNamed:name];
        NSAssert(image, @"需要一张图片");

        lookupImageSource = [[GPUImagePicture alloc] initWithImage:image];
        GPUImageLookupFilter *lookupFilter = [[GPUImageLookupFilter alloc] init];
        [self addFilter:lookupFilter];

        [lookupImageSource addTarget:lookupFilter atTextureLocation:1];
        [lookupImageSource processImage];

        self.initialFilters = [NSArray arrayWithObjects:lookupFilter, nil];
        self.terminalFilter = lookupFilter;
    }
    return self;
}
```

加水印压缩


AVMutableComposition
AVMutableCompositionTrack (video)
AVMutableCompositionTrack (video)
AVMutableCompositionTrack (audio)
AVMutableVideoComposition
AVMutableVideoCompositionInstruction
AVMutableVideoCompositionInstruction
AVMutableAudioMix
AVMutableAudioMixInputParameters
AVMutableAudioMixInputParameters
AVAssetExportSession
Output Video File



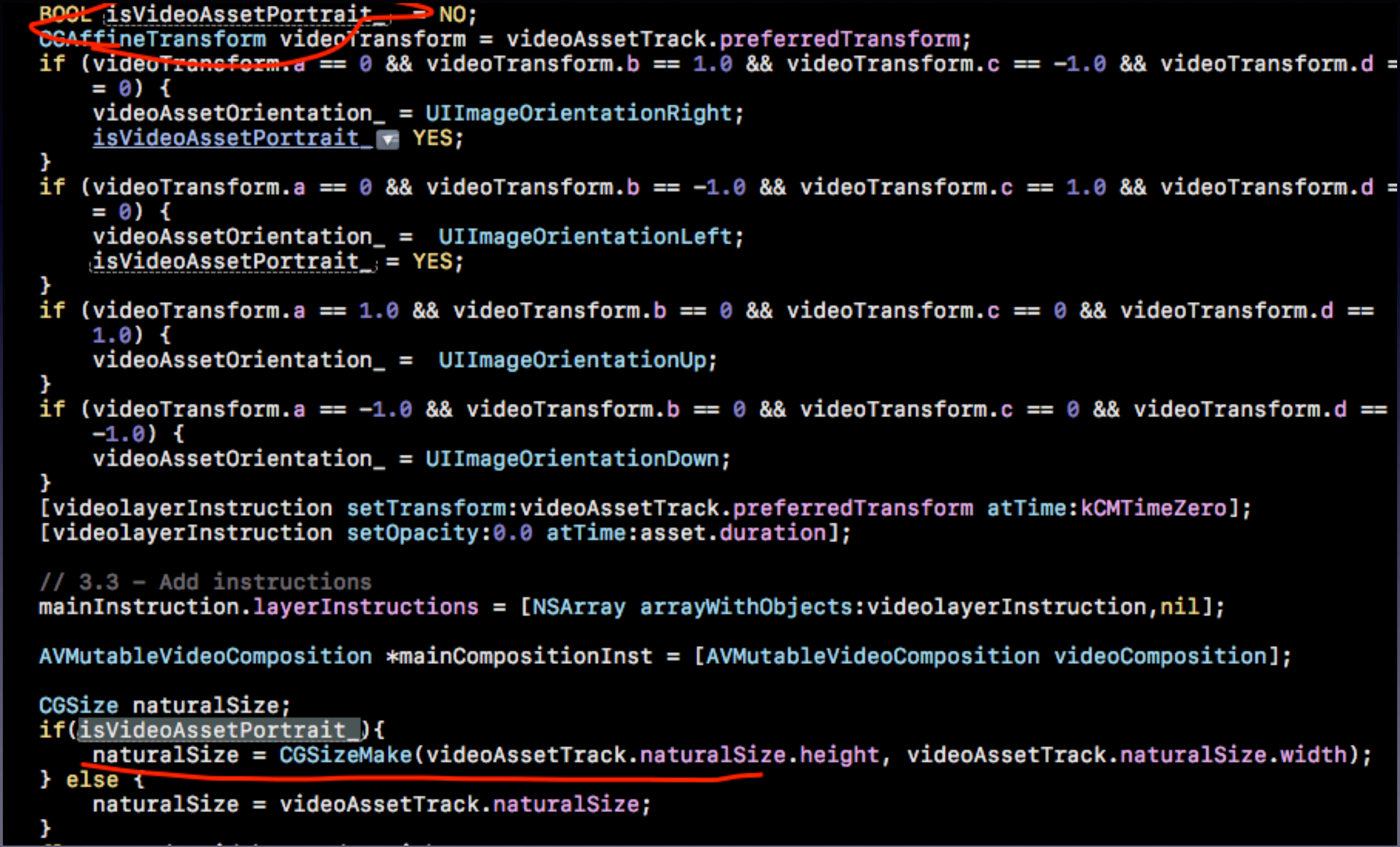
```
CALayer *parentLayer = [CALayer layer];
CALayer *videoLayer = [CALayer layer];
parentLayer.frame = CGRectMake(0, 0, naturalSize.width, naturalSize.height);
videoLayer.frame = CGRectMake(0, 0, naturalSize.width, naturalSize.height);
[parentLayer addSublayer:videoLayer];
[parentLayer addSublayer:overlayLayer];

mainCompositionInst.animationTool = [AVVideoCompositionCoreAnimationTool
                                     videoCompositionCoreAnimationToolWithPostProcessingAsVideoLayer:
                                         videoLayer inLayer:parentLayer];
```

详细了解：AVFoundation Programming Guide

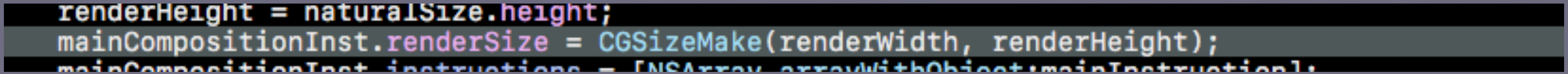
遇到的一些问题

- 视频压缩错位
- 视频绿边
- 第一帧黑屏

视频压缩错位



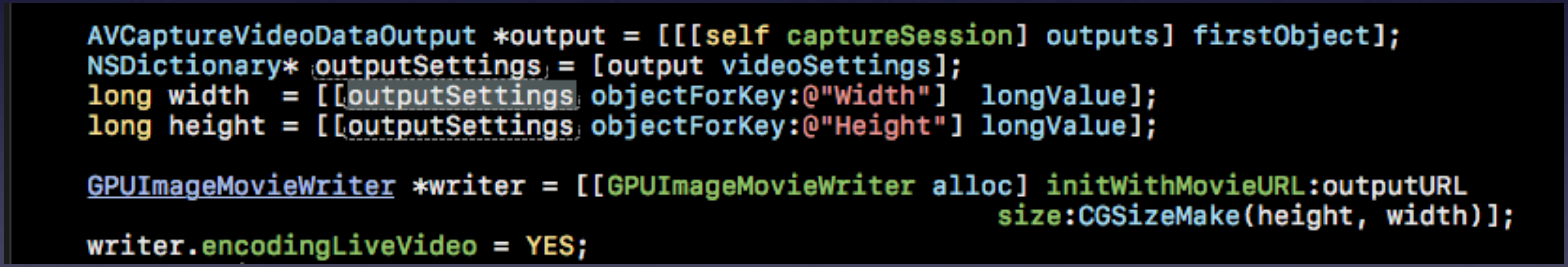
```
BOOL isVideoAssetPortrait_ = NO;
CGAffineTransform videoTransform = videoAssetTrack.preferredTransform;
if (videoTransform.a == 0 && videoTransform.b == 1.0 && videoTransform.c == -1.0 && videoTransform.d =
    = 0) {
    videoAssetOrientation_ = UIImageOrientationRight;
    isVideoAssetPortrait_ = YES;
}
if (videoTransform.a == 0 && videoTransform.b == -1.0 && videoTransform.c == 1.0 && videoTransform.d =
    = 0) {
    videoAssetOrientation_ =  UIImageOrientationLeft;
    isVideoAssetPortrait_ = YES;
}
if (videoTransform.a == 1.0 && videoTransform.b == 0 && videoTransform.c == 0 && videoTransform.d ==
    1.0) {
    videoAssetOrientation_ =  UIImageOrientationUp;
}
if (videoTransform.a == -1.0 && videoTransform.b == 0 && videoTransform.c == 0 && videoTransform.d ==
    -1.0) {
    videoAssetOrientation_ = UIImageOrientationDown;
}
[videolayerInstruction setTransform:videoAssetTrack.preferredTransform atTime:kCMTimeZero];
[videolayerInstruction setOpacity:0.0 atTime:asset.duration];

// 3.3 - Add instructions
mainInstruction.layerInstructions = [NSArray arrayWithObjects:videolayerInstruction,nil];

AVMutableVideoComposition *mainCompositionInst = [AVMutableVideoComposition videoComposition];

CGSize naturalSize;
if(isVideoAssetPortrait_){
    naturalSize = CGSizeMake(videoAssetTrack.naturalSize.height, videoAssetTrack.naturalSize.width);
} else {
    naturalSize = videoAssetTrack.naturalSize;
}
```



```
    renderHeight = naturalSize.height;
    mainCompositionInst.renderSize = CGSizeMake(renderWidth, renderHeight);
    mainCompositionInst.instructions = [NSArray arrayWithObject:mainInstruction];
```

视频绿边

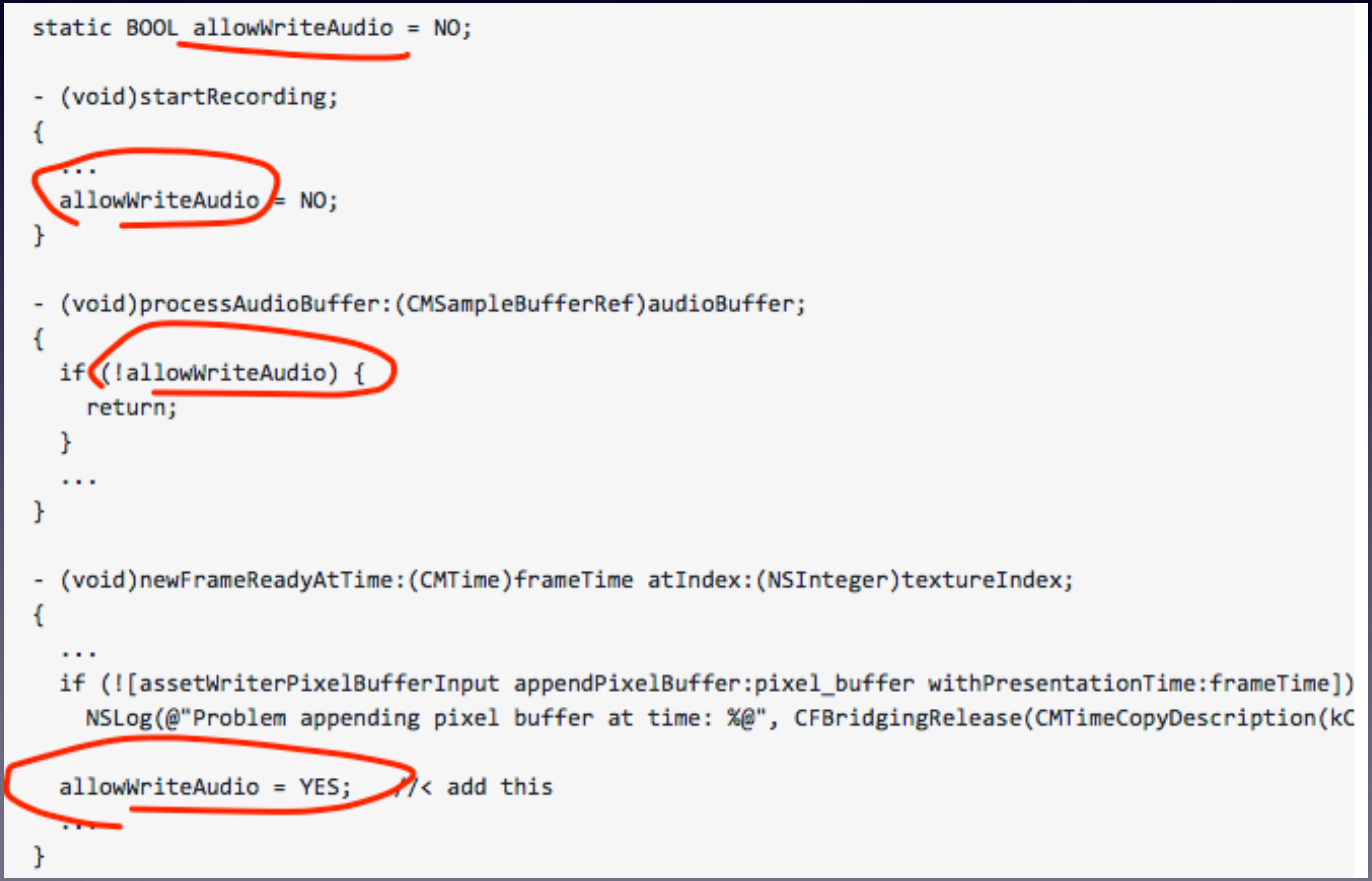
视频输出尺寸与设置尺寸不同，差一像素。



```
AVCaptureVideoDataOutput *output = [[[self captureSession] outputs] firstObject];
NSDictionary* outputSettings = [output videoSettings];
long width  = [[outputSettings objectForKey:@"Width"]  longValue];
long height = [[outputSettings objectForKey:@"Height"] longValue];

GPUImageMovieWriter *writer = [[GPUImageMovieWriter alloc] initWithMovieURL:outputURL
                                                           size:CGSizeMake(height, width)];
writer.encodingLiveVideo = YES;
```

第一帧黑屏

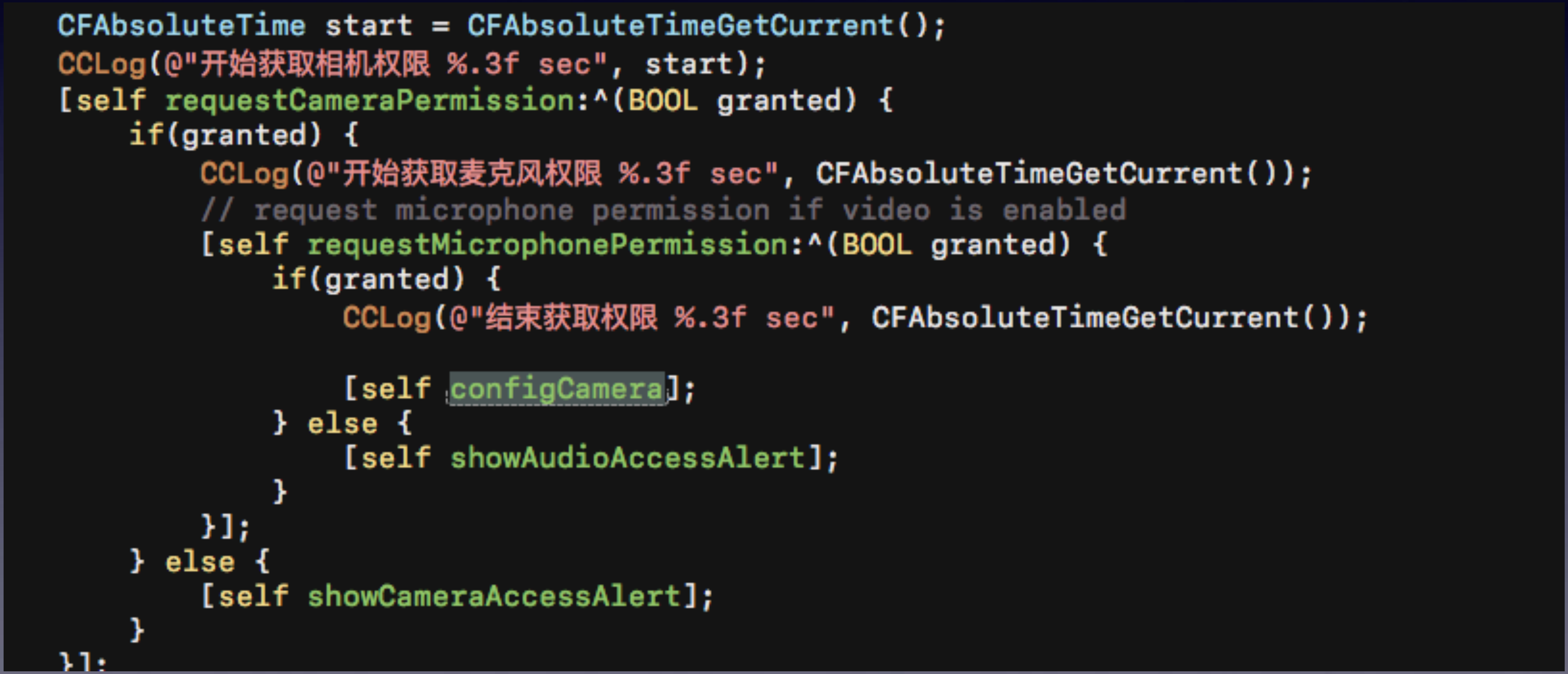
音频先到，视频未到



```
static BOOL allowWriteAudio = NO;

- (void)startRecording;
{
  ...
  allowWriteAudio = NO;
}

- (void)processAudioBuffer:(CMSampleBufferRef)audioBuffer;
{
  if (!allowWriteAudio) {
    return;
  }
  ...
}

- (void)newFrameReadyAtTime:(CMTime)frameTime atIndex:(NSInteger)textureIndex;
{
  ...
  if (![assetWriterPixelBufferInput appendPixelBuffer:pixel_buffer withPresentationTime:frameTime])
    NSLog(@"Problem appending pixel buffer at time: %@", CFBridgingRelease(CMTimeCopyDescription(kC

  allowWriteAudio = YES;   //< add this
  ...
}
```

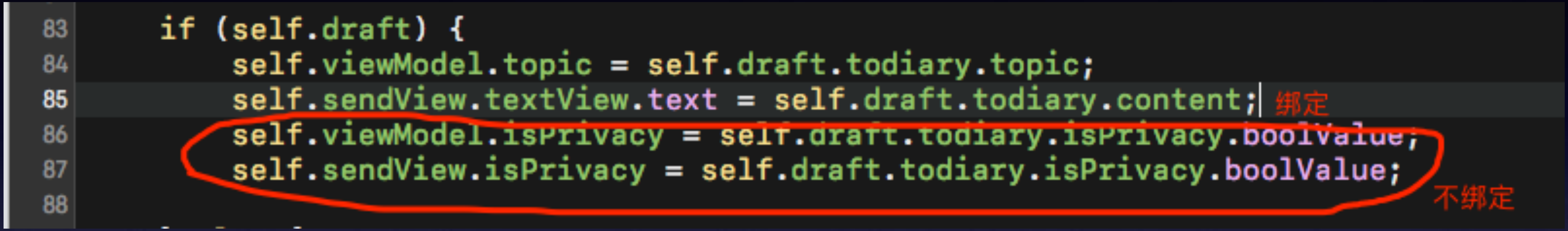
权限

每次都请求权限，直接获取权限会没有声音

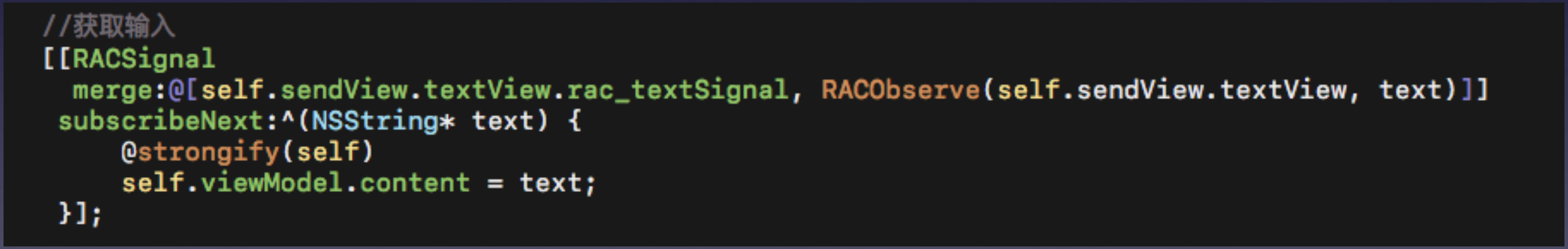


```
CFAbsoluteTime start = CFAbsoluteTimeGetCurrent();
CCLog(@"开始获取相机权限 %.3f sec", start);
[self requestCameraPermission:^(BOOL granted) {
    if(granted) {
        CCLog(@"开始获取麦克风权限 %.3f sec", CFAbsoluteTimeGetCurrent());
        // request microphone permission if video is enabled
        [self requestMicrophonePermission:^(BOOL granted) {
            if(granted) {
                CCLog(@"结束获取权限 %.3f sec", CFAbsoluteTimeGetCurrent());

                [self configCamera];
            } else {
                [self showAudioAccessAlert];
            }
        }];
    } else {
        [self showCameraAccessAlert];
    }
}];
```

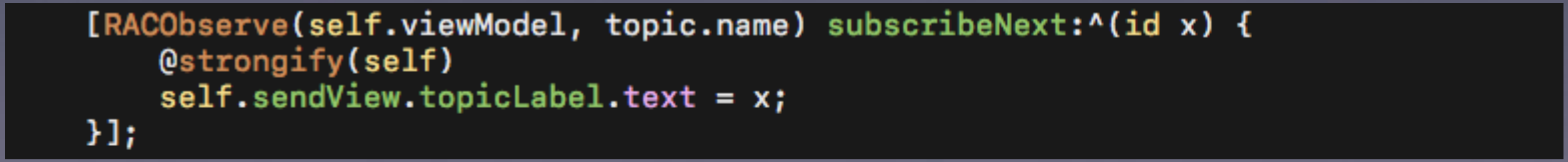
ViewModel



```
    if (self.draft) {
        self.viewModel.topic = self.draft.todiary.topic;
        self.sendView.textView.text = self.draft.todiary.content;
        self.viewModel.isPrivacy = self.draft.todiary.isPrivacy.boolValue;
        self.sendView.isPrivacy = self.draft.todiary.isPrivacy.boolValue;

```

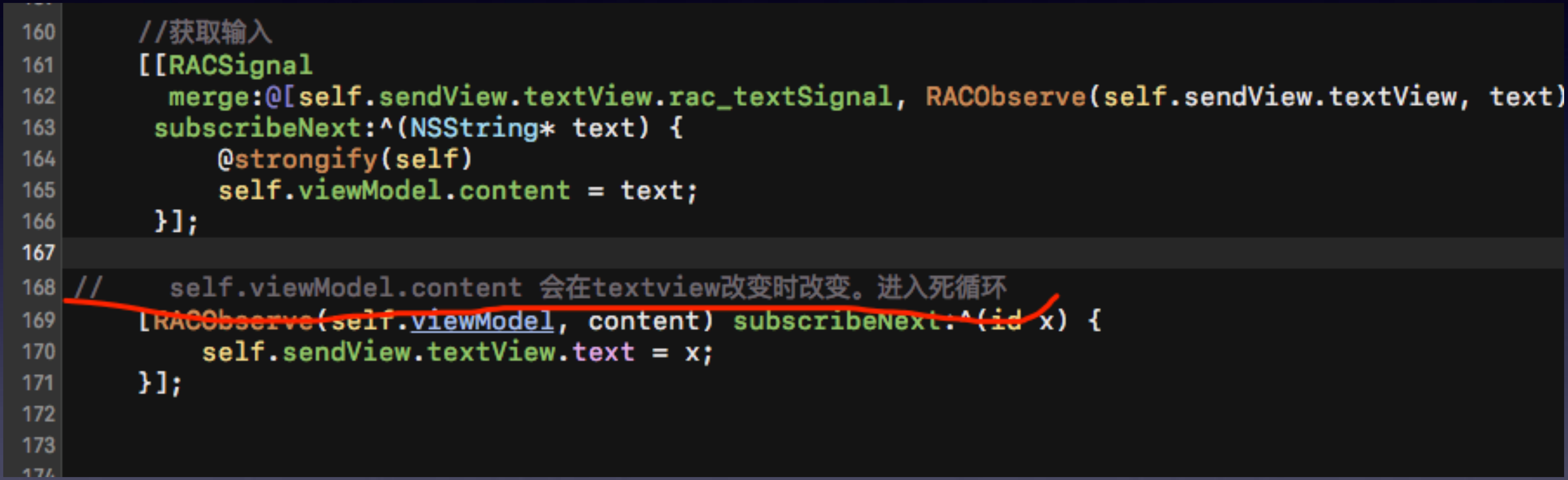


```
  //获取输入
  [[RACSignal
    merge:@[self.sendView.textView.rac_textSignal, RACObserve(self.sendView.textView, text)]]
   subscribeNext:^(NSString* text) {
       @strongify(self)
       self.viewModel.content = text;
   }];
```



```
    [RACObserve(self.viewModel, topic.name) subscribeNext:^(id x) {
        @strongify(self)
        self.sendView.topicLabel.text = x;
    }];
```

可输入也可展示的控件，绑定问题



```
    //获取输入
    [[RACSignal
      merge:@[self.sendView.textView.rac_textSignal, RACObserve(self.sendView.textView, text)
     subscribeNext:^(NSString* text) {
         @strongify(self)
         self.viewModel.content = text;
     }];

//    self.viewModel.content 会在textview改变时改变。进入死循环
    ~~[RACObserve(self.viewModel, content) subscribeNext:^(id~~ x) {
        self.sendView.textView.text = x;
    }];
```

谢谢