

FULiveDemoMac

FULiveDemo 是集成了 Faceunity 面部跟踪、美颜、Animoji、道具贴纸、AR面具、换脸、背景分割以及视频滤镜功能的 Mac 版 Demo。

注：第一运行Demo会报缺少证书的 error ,如果您已拥有我司颁发的证书，将证书替换到工程中重新运行即可。如您还没有我司颁发的证书，可以查看[这里](#)获取证书

SDK v5.5.0 更新

更新内容

- 智能美肤性能优化提升
- 表情跟踪针对细微表情优化
- 修复MAC上rgb/bgr问题

软件需求

一、支持平台

macOS 10.6以上系统

二、开发环境

Xcode 8或更高版本

SDK集成

一、通过cocoapods集成

含有深度学习的版本：

```
pod 'Nama-macOS', '5.5.0' #注意此版本目前为dev版
```

不含深度学习的版本（lite版）：

```
pod 'Nama-macOS-lite', '5.5.0' #注意此版本目前为dev版
```

接下来执行：

```
pod install
```

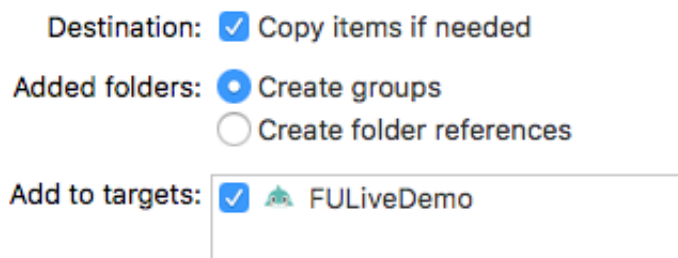
如果提示无法找到该版本，请尝试执行以下指令后再试：

```
pod repo update 或 pod setup
```






二、通过 github 下载集成

含有深度学习的版本：[FaceUnity-SDK-Mac-v5.5.0-dev.zip](#) 不含深度学习的版本（lite版）：[FaceUnity-SDK-Mac-v5.5.0-dev-lite.zip](#)

下载完成并解压后将库文件夹拖入到工程中，并勾选上 Copy items if needed，如图：



然后在Build Phases → Link Binary With Libraries 中添加依赖库，这里需要添加 OpenGL.framework、Accelerate.framework、CoreMedia.framework、AVFoundation.framework、libc++.tbd 这几个依赖库，如图：

Name	Status
 CoreMedia.framework	Required ⚡
 OpenGL.framework	Required ⚡
 Accelerate.framework	Required ⚡
 AVFoundation.framework	Required ⚡
 libc++.tbd	Required ⚡

文件说明

一、头文件

- authpack.h 证书文件，一般由我司通过邮箱发送给使用者
- funama.h C接口头文件
- FURenderer.h OC接口头文件

二、库文件

- libnama.a 人脸跟踪及道具绘制核心静态库

三、数据文件

- v3.bundle 初始化必须的二进制文件

- face_beautification.bundle 我司美颜相关的二进制文件
- anim_model.bundle 表情优化相关二进制文件
- ardata_ex.bundle 高精度模型相关二进制文件
- fxaa.bundle 3D道具去锯齿二进制文件
- items/*.bundle 该文件夹位于 FULiveDemo 的字文件夹中，这些 .bundle 文件是我司制作的特效贴纸文件，自定义特效贴纸制作的文档和工具请联系我司获取。

注：这些数据文件都是二进制数据，与扩展名无关。实际在app中使用时，打包在程序内或者从网络接口下载这些数据都是可行的，只要在相应的函数接口传入正确的文件路径即可。

导入证书

您需要拥有我司颁发的证书才能使用我们的SDK的功能，获取证书方法：

- 1、拨打电话 **0571-89774660**
- 2、发送邮件至 marketing@faceunity.com 进行咨询。

iOS端发放的证书为包含在authpack.h中的g_auth_package数组，如果您已经获取到鉴权证书，将authpack.h导入工程中即可。根据应用需求，鉴权数据也可以在运行时提供(如网络下载)，不过要注意证书泄露风险，防止证书被滥用。

初始化

首先在代码中引入 FURenderer.h 头文件

```
#import "FURenderer.h"
```

然后执行初始化

```
NSString *v3Path = [[NSBundle mainBundle] pathForResource:@"v3"
ofType:@"bundle"];

[[FURenderer shareRenderer] setupWithDataPath:v3Path
authPackage:g_auth_package authSize:sizeof(g_auth_package)
shouldCreateContext:YES];
```

注：app启动后只需要setup一次FURenderer即可，其中 `g_auth_package` 密钥数组声明在authpack.h 中。

至此，工程的配置及 SDK 的初始化工作已全部完成，下面就可以通过我们的 SDK 进行视频处理了！

接口说明：

```
- (void)setupWithDataPath:(NSString *)v3path
    authPackage:(void *)package
    authSize:(int)size
    shouldCreateContext:(BOOL)create;
```

参数说明：

`v3path` v3.bundle 文件路径

`package` 内存指针，指向鉴权数据的内容。如果是用包含 `authpack.h` 的方法在编译时提供鉴权数据，则这里可以写为 `g_auth_package`

`size` 鉴权数据的长度，以字节为单位。如果鉴权数据提供的是 `authpack.h` 中的 `g_auth_package`，这里可写作 `sizeof(g_auth_package)`

`create` 如果设置为YES，我们会在内部创建并持有一个context，这种情况下工程中必须要使用OC层接口

视频处理

处理视频之前，首先需要创建道具句柄，然后将视频图像数据及道具句柄一同传入我们的绘制接口，处理完成之后道具中的特效就被绘制到图像中了。

一、创建道具：

创建道具接口：

```
+ (int)itemWithContentsOfFile:(NSString *)path
```

参数说明：

`path` 道具路径

返回值：

`int` 道具句柄

示例：

```
NSString *path = [[NSBundle mainBundle] pathForResource:@"tiara"
ofType:@"bundle"];

int itemHandle = [FURenderer itemWithContentsOfFile:path];
```

在实际应用中有时需要同时使用多个道具，我们的图像处理接口接受的参数是一个包含多个道具句柄的int数组，所以我们需要将创建一个int数组来保存这些道具句柄。下面我们将创建一个花环道具的句柄并保存在int数组的第0位，示例如下：

```
int items[3];

NSString *path = [[NSBundle mainBundle] pathForResource:@"tiara"
ofType:@"bundle"];

int itemHandle = [FURenderer itemWithContentsOfFile:path];

items[0] = itemHandle;
```

二、视频图像处理：

将上一步创建的包含一个贴纸道具句柄的items数组传入视频图像处理接口，同时传入需要被处理的图像，即可为图像添加特效贴纸，示例如下：

```
CVPixelBufferRef pixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer);

[[FURenderer shareRenderer] renderPixelBuffer:pixelBuffer
withFrameId:frameID items:items itemCount:sizeof(items)/sizeof(int)
flipx:YES];

frameID += 1;
```

图像处理接口说明：

```
- (CVPixelBufferRef)renderPixelBuffer:(CVPixelBufferRef)pixelBuffer
withFrameId:(int)frameid
items:(int*)items
itemCount:(int)itemCount
flipx:(BOOL)flip;
```

参数说明：

`pixelBuffer` 图像数据，支持的格式为：BGRA、YUV420SP

`frameid` 当前处理的视频帧序号，每次处理完对其进行加1操作，不加1将无法驱动道具中的特效动画

`items` 包含多个道具句柄的int数组

`itemCount` 句柄数组中包含的句柄个数

`flip` 道具镜像使能，如果设置为YES可以将道具做镜像操作

返回值：

`CVPixelBufferRef` 被处理过的的图像数据

三、道具销毁与切换：

销毁单个道具：

```
/**
 销毁单个道具
 */
+ (void)destroyItem:(int)item;
```

参数说明：

`item` 要销毁的道具句柄

该接口将释放传入的句柄所对应的资源，为保证编程的严谨性，在执行完该操作后请将该句柄置为0。示例如下：

```
if (items[0] != 0) {
    [FURenderer destroyItem:items[0]];
}
items[0] = 0;
```

销毁全部道具：

```
/**
 销毁所有道具
 */
+ (void)destroyAllItems;
```

该接口可以销毁全部道具句柄所对应的资源,同样在执行完该接口后请将所有句柄都置为0。示例如下：

```
[FURenderer destroyAllItems];

for (int i = 0; i < sizeof(items) / sizeof(int); i++) {
    items[i] = 0;
}
```

道具切换：

如果需要切换句柄数组中某一位的句柄时，需要先创建一个新的道具句柄，并将该句柄替换到句柄数组中需要被替换的位置上，最后再把被替换的句柄销毁掉。下面以替换句柄数组的第0位为例进行说明：

```

// 先创建再释放可以有效缓解切换道具卡顿问题
NSString *path = [[NSBundle mainBundle]
pathForResource:_demoBar.selectedItem ofType:@"bundle"];
int itemHandle = [FURenderer itemWithContentsOfFile:path];

if (items[0] != 0) {
    [FURenderer destroyItem:items[0]];
}

items[0] = itemHandle;

```

注意：如果这里先销毁了老的道具，再创建新的道具可能会出现道具不连续的现象，即短时间没有道具的现象。

视频美颜

美颜处理

视频美颜配置方法与视频加特效道具类似，首先创建美颜道具句柄，并保存在上面提到的items数组的items[1]中,示例如下：

```

- (void)loadFilter
{
    NSString *path = [[NSBundle mainBundle]
pathForResource:@"face_beautification" ofType:@"bundle"];
    items[1] = [FURenderer itemWithContentsOfFile:path];
}

```

在处理视频时，将包含美颜道具句柄的items数组传入视频图像处理接口，同时传入需要被处理的图像，即可为图像添加美颜效果，示例如下：

```

CVPixelBufferRef pixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer);

[[FURenderer shareRenderer] renderPixelBuffer:pixelBuffer
withFrameId:frameID items:items itemCount:sizeof(items)/sizeof(int)
flipx:YES];

frameID += 1;

```

美颜参数设置

美颜道具主要包含七个模块的内容：滤镜、美白、红润、磨皮、亮眼、美牙、美型。每个模块都有默认效果，它们可以调节的参数如下。

一、滤镜

目前版本中提供以下滤镜：

普通滤镜：

```
"origin", "delta", "electric", "slowlived", "tokyo", "warm"
```

美颜滤镜：

```
"ziran", "danya", "fennen", "qingxin", "hongrun"
```

其中 "origin" 为原图滤镜，其他滤镜属于风格化滤镜及美颜滤镜，美颜滤镜具有一定美颜、增白、亮唇等功能。滤镜由参数 filter_name 指定。切换滤镜时，通过 fultemSetParams 设置美颜道具的参数，如下：

```
// Set item parameters - filter
[FURenderer itemSetParam:items[1] withName:@"filter_name" value:@"origin"];
```

另外滤镜开放了滤镜强度接口，可以通过参数 filter_level 来控制当前滤镜程度。该参数的取值范围为 [0, 1]，0 为无效果，1.0 为默认效果。客户端需要针对每个滤镜记录用户的选择的 filter_level，当切换滤镜时，设置该参数。

二、美白和红润

美白

通过参数 color_level 来控制美白程度。该参数的推荐取值范围为 0~1，0 为无效果，0.5 为默认效果，大于 1 为继续增强效果。

设置参数的例子代码如下：

```
// Set item parameters - whiten
[FURenderer itemSetParam:items[1] withName:@"color_level" value:@(0.5)];
```

红润

通过参数 red_level 来控制红润程度。该参数的推荐取值范围为 0~1，0 为无效果，0.5 为默认效果，大于 1 为继续增强效果。

```
// Set item parameters - red
[FURenderer itemSetParam:items[1] withName:@"red_level" value:@(0.5)];
```

注：新增的美颜滤镜如 “shaonv” 滤镜本身能够美白肤色，提亮红唇，开启该滤镜时，适当减弱独立的美白红润功能。

三、磨皮

新版美颜中，控制磨皮的参数有五个：blur_level, skin_detect, nonshin_blur_scale, heavy_blur, blur_blend_ratio。

`blur_level` 指定磨皮程度。该参数的推荐取值范围为[0, 6], 0为无效果, 对应7个不同的磨皮程度。

`skin_detect` 指定是否开启皮肤检测, 开启后, 将自动检测是否皮肤, 是皮肤的区域将直接根据 `blur_level`指定的磨皮程度进行磨皮, 非皮肤区域将减轻磨皮导致模糊的效果。该参数的推荐取值为0-1, 0为无效果, 1为开启皮肤检测, 默认不开启。

`nonshin_blur_scale` 指定开启皮肤检测后, 非皮肤区域减轻磨皮导致模糊的程度。该参数范围是[0.0,1.0], 0表示不磨皮, 1表示完全磨皮, 默认值为0.45。调整该参数需要先开启 `skin_detect`。

新增朦胧美肤:

`heavy_blur` 指定是否开启朦胧美肤功能。大于1开启朦胧美肤功能。

`blur_blend_ratio` 指定磨皮结果和原图融合率。该参数的推荐取值范围为0-1。

注意: 朦胧美肤使用了比较强的模糊算法, 优点是会把皮肤磨得更加光滑, 瑕疵更少, 而且性能比老版磨皮性能更好, 缺点是会降低一些清晰度。另外开启朦胧美肤后`blur_level`, `skin_detect`两个参数继续有效, 而 `nonshin_blur_scale` 参数对朦胧美肤无效

设置参数的例子代码如下:

```
// Set item parameters - blur
[FURenderer itemSetParam:items[1] withName:@"heavy_blur" value:@(1)];
[FURenderer itemSetParam:items[1] withName:@"skin_detect" value:@(1)];
[FURenderer itemSetParam:items[1] withName:@"blur_level" value:@(6.0)];
[FURenderer itemSetParam:items[1] withName:@"blur_blend_ratio"
value:@(0.5)];
[FURenderer itemSetParam:items[1] withName:@"nonshin_blur_scale"
value:@(0.45)];
```

四、亮眼

使眼睛区域的纹理变得更加清晰, 眼眸更加明亮。可通过参数 `eye_bright` 来控制亮眼程度。该参数的推荐取值范围为0~1, 0为关闭该功能, 0到1效果逐渐增强。

设置参数的例子代码如下:

```
// Set item parameters - eye_bright
[FURenderer itemSetParam:items[1] withName:@"eye_bright" value:@(0.5)];
```

五、美牙

使牙齿区域变得更亮更白。可通过参数 `tooth_whiten` 来控制美牙程度。该参数的推荐取值范围为0~1, 0为关闭该功能, 0到1效果逐渐增强。

设置参数的例子代码如下:

```
// Set item parameters - tooth_whiten
[FURenderer itemSetParam:items[1] withName:@"tooth_whiten" value:@(0.5)];
```

六、美型

1、基本美型

美型支持四种基本美型：女神、网红、自然、默认，一种高级美型：自定义。由参数 face_shape 指定：默认（3）、女神（0）、网红（1）、自然（2）、自定义（4）。

```
// Set item parameters - shaping
[FURenderer itemSetParam:items[1] withName:@"face_shape" value:@(3.0)];
```

在上述四种基本美型及一种高级美型的基础上，我们提供了以下三个参数：face_shape_level、eye_enlarging、cheek_thinning。

参数 face_shape_level 用以控制变化到指定基础脸型的程度。该参数的取值范围为[0, 1]。0为无效果，即关闭美型，1为指定脸型。

若要关闭美型，可将 face_shape_level 设置为0。

```
// Set item parameters - shaping level
[FURenderer itemSetParam:items[1] withName:@"face_shape_level"
value:@(1.0)];
```

参数 eye_enlarging 用以控制眼睛大小。此参数受参数 face_shape_level 影响。该参数的推荐取值范围为[0, 1]。大于1为继续增强效果。

```
// Set item parameters - eye enlarging level
[FURenderer itemSetParam:items[1] withName:@"eye_enlarging" value:@(1.0)];
```

参数 cheek_thinning 用以控制脸大小。此参数受参数 face_shape_level 影响。该参数的推荐取值范围为[0, 1]。大于1为继续增强效果。

```
// Set item parameters - cheek thinning level
[FURenderer itemSetParam:items[1] withName:@"cheek_thinning" value:@(1.0)];
```

2、高级美型

精细脸型调整功能

新增优化瘦脸、大眼的效果，增加额头调整、下巴调整、瘦鼻、嘴型调整4项美颜变形，将 face_shape 设为4即可开启精细脸型调整功能，FULiveDemo中可以在脸型中选择自定义来开启精细脸型调整功能

使用方法：

- 加载face_beautification.bundle
- 调整如下参数 face_shape: 4, // 4为开启高级美型模式，0~3为基本美型

瘦脸

优化瘦脸变形效果，比之前更加自然

使用方法：

- 加载face_beautification.bundle
- 调整如下参数 face_shape: 4, // 4为开启高级美型模式，0~3为基本美型
cheek_thinning: 0.0, // 使用了原有参数cheek_thinning控制瘦脸，范围0 - 1

大眼

优化大眼变形效果，比之前更加自然

使用方法：

- 加载face_beautification.bundle
- 调整如下参数 face_shape: 4, // 4为开启高级美型模式，0~3为基本美型
eye_enlarging: 0.0, // 使用了原有参数eye_enlarging控制大眼，范围0 - 1

额头调整

新增加的一款美颜变形，可以调整额头大小

使用方法：

- 加载face_beautification.bundle
- 调整如下参数 face_shape: 4, // 4为开启高级美型模式，0~3为基本美型
intensity_forehead: 0.5, // 大于0.5 变大，小于0.5变小

下巴调整

新增加的一款美颜变形，可以调整下巴大小

使用方法：

- 加载face_beautification.bundle
- 调整如下参数 face_shape: 4, // 4为开启高级美型模式，0~3为基本美型
intensity_chin: 0.5, // 大于0.5 变大，小于0.5变小

瘦鼻

新增加的一款美颜变形，可以进行瘦鼻操作

使用方法：

- 加载face_beautification.bundle
- 调整如下参数
face_shape: 4, // 4为开启高级美型模式，0~3为基本美型 intensity_nose: 0.0, // 0为正常大小，大于0开始瘦鼻，范围0 - 1

嘴型调整

新增加的一款美颜变形，可以调整嘴型大小

使用方法：

- 加载face_beautification.bundle
- 调整如下参数 face_shape: 4, // 4为开启高级美型模式, 0~3为基本美型 intensity_mouth: 0.5, // 大于0.5变大, 小于0.5变小

七、美颜美型突变过渡效果

使美颜变形过度的更自然, 避免突变效果, 可通过参数 change_frames 来控制渐变所需要的帧数, 0 渐变关闭, 大于0开启渐变, 值为渐变所需要的帧数。

设置参数的例子代码如下:

```
// Set item parameters - change_frames
[FURenderer itemSetParam:items[1] withName:@"change_frames" value:@(10)];
```

八、平台相关

PC及MAC端的美颜, 使用前必须将参数 is_opengl_es 设置为 0, 移动端无需此操作:

```
// Set item parameters
[FURenderer itemSetParam:items[1] withName:@"is_opengl_es" value:@(0)];
```

手势识别

目前我们的手势识别功能也是以道具的形式进行加载的。一个手势识别的道具中包含了要识别的手势、识别到该手势时触发的动效、及控制脚本。加载该道具的过程和加载普通道具、美颜道具的方法一致。

线上例子中 heart_v2.bundle 为爱心手势演示道具。将其作为道具加载进行绘制即可启用手势识别功能。手势识别道具可以和普通道具及美颜共存, 类似美颜将手势道具句柄保存在items句柄数组即可。

自定义手势道具的流程和2D道具制作一致, 具体打包的细节可以联系我司技术支持。

注: 新版手势道具中部分道具需要使用非lite版SDK才能正常使用

3D绘制抗锯齿功能

高效全屏抗锯齿, 使得3D绘制效果更加平滑。

使用方法:

- 加载fxaa.bundle, 随新版本SDK提供
- 绘制时将fxaa.bundle放在道具数组最后一个

照片驱动功能

针对照片进行精确的人脸重建, 然后支持实时表情驱动, 预置表情播放。可以用于实时应用, 也可以用于生成表情包等。

该功能的资源有两种方式生成方式：

- 使用FUEditor v4.3.0以上版本离线制作道具
- 利用相芯提供的云服务在线上传照片生成道具 在线云服务的方式请联系技术支持获取更多细节。

使用方法：

- 直接加载对应的道具
- 需要带有照片驱动权限的证书

人脸夸张变形功能

新增了5款夸张变形。

使用方法：

- 直接加载对应的道具
- 需要带有照片驱动权限的证书

音乐节奏滤镜

效果详见FULiveDemo，道具可以通过FUEditor进行制作（v4.2.1及以上）。

优化表情校准功能

- 被动校准：该种模式下会在整个用户使用过程中逐渐进行表情校准，用户对该过程没有明显感觉。该种校准的强度相比主动校准较弱。

使用方法：

- 调用 `fuSetExpressionCalibration` 接口控制表情校准功能的开关及不同模式，参数为0时关闭表情校准，2为被动校准。

注：优化后的SDK只支持被动校准功能，即fuSetExpressionCalibration接口只支持0（关闭）或2（被动校准）这两个数字，设置为1时将不再有效果。

接口说明

一、OC层API参考文档

[Nama API OC Reference](#)

二、C层API参考文档

[Nama API reference](#)

鉴权

我们的系统通过标准TLS证书进行鉴权。客户在使用时先从发证机构申请证书，之后将证书数据写在客户端代码中，客户端运行时发回我司服务器进行验证。在证书有效期内，可以正常使用库函数所提供的各种功能。没有证书或者证书失效等鉴权失败的情况会限制库函数的功能，在开始运行一段时间后自动终止。

证书类型分为**两种**，分别为**发证机构证书**和**终端用户证书**。

- 发证机构证书

适用对象：此类证书适合需批量生成终端证书的机构或公司，比如软件代理商，大客户等。

发证机构的二级CA证书必须由我司颁发，具体流程如下。

1. 机构生成私钥 机构调用以下命令在本地生成私钥 CERT_NAME.key，其中 CERT_NAME 为机构名称。

```
openssl ecparam -name prime256v1 -genkey -out CERT_NAME.key
```

1. 机构根据私钥生成证书签发请求 机构根据本地生成的私钥，调用以下命令生成证书签发请求 CERT_NAME.csr。在生成证书签发请求的过程中注意在 Common Name 字段中填写机构的正式名称。

```
openssl req -new -sha256 -key CERT_NAME.key -out CERT_NAME.csr
```

1. 将证书签发请求发回我司颁发机构证书

之后发证机构就可以独立进行终端用户的证书发行工作，不再需要我司的配合。

如果需要在终端用户证书有效期内终止证书，可以由机构自行用OpenSSL吊销，然后生成pem格式的吊销列表文件发给我们。例如如果要吊销先前误发的 "bad_client.crt"，可以如下操作：

```
openssl ca -config ca.conf -revoke bad_client.crt -keyfile CERT_NAME.key -cert CERT_NAME.crt
openssl ca -config ca.conf -gencrl -keyfile CERT_NAME.key -cert CERT_NAME.crt -out CERT_NAME.crl.pem
```

然后将生成的 CERT_NAME.crl.pem 发回给我司。

- 终端用户证书

适用对象：直接的终端证书使用者。比如，直接客户或个人等。

终端用户由我司或者其他发证机构颁发证书，并通过我司的证书工具生成一个代码头文件交给用户。该文件中是一个常量数组，内容是加密之后的证书数据，形式如下。

```
static char g_auth_package[]={ ... }
```

用户在库环境初始化时，需要提供该数组进行鉴权，具体参考 fuSetup 接口。没有证书、证书失效、网络连接失败等情况下，会造成鉴权失败，在控制台或者Android平台的log里面打出 "not authenticated" 信息，并在运行一段时间后停止渲染道具。

任何其他关于授权问题，请email: support@faceunity.com