

iOS GPUImageDemo

Пример приложения с захватом видео с использованием библиотеки GPUImage

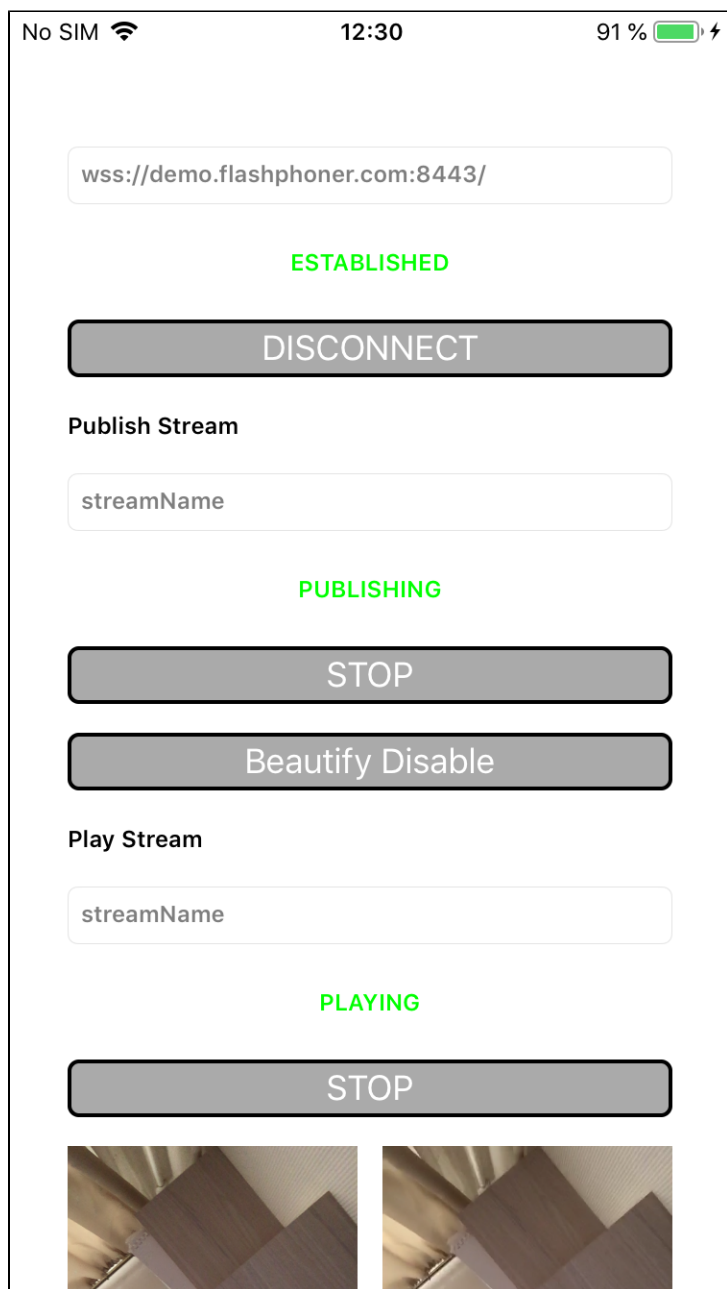
Данное приложение демонстрирует возможность захвата видео из кастомного источника. В качестве источника используется фильтр-бьютификатор на основе библиотеки GPUImage. Приложение работает только с версией SDK [2.6.1](#) и новее.

На скриншоте представлен пример публикации потока с фильтром бьютификации

Поля ввода

- 'WCS URL', где demo.flashphoner.com - адрес WCS-сервера
- 'Publish Stream' - для имени публикуемого потока
- 'Play Stream' - для имени воспроизводимого потока

Кнопка Beautify Enable/Disable включает и отключает фильтр бьютификации (на скриншоте фильтр включен)



Работа с кодом примера

Для разбора кода возьмем версию примера GPUImageDemo, которая доступна для скачивания [здесь](#).

Класс для основного вида приложения: ViewController (заголовочный файл [ViewController.h](#); файл имплементации [ViewController.m](#)).

1. Импорт API

[code](#)

```
#import <FPWCSApi2/FPWCSApi2.h>
```

2. Создание сессии и подключение к серверу.

FPWCSApi2 createSession, FPWCSApi2Session connect

[code](#)

В параметрах сессии указываются:

- URL WCS-сервера
- имя серверного приложения defaultApp

```
- (FPWCSApi2Session *)connect {
    FPWCSApi2SessionOptions *options = [[FPWCSApi2SessionOptions alloc] init];
    options.urlServer = _connectUrl.text;
    options.appKey = @"defaultApp";
    NSError *error;
    FPWCSApi2Session *session = [FPWCSApi2 createSession:options error:&error];
    ...
    [session connect];
    return session;
}
```

3. Подготовка захвата из кастомного источника

[code](#)

Источник инициализируется один раз при первом входе в функцию publishStream

```
- (FPWCSApi2Stream *)publishStream {
    FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = _localStreamName.text;

    if (!self.videoCamera) {
        self.videoCamera = [[GPUImageVideoCamera alloc] initWithSessionPreset:AVCaptureSessionPreset640x480
        cameraPosition:AVCaptureDevicePositionFront];
        self.videoCamera.outputImageOrientation = UIInterfaceOrientationPortrait;
        self.videoCamera.horizontallyMirrorFrontFacingCamera = YES;

        self.rawDataOutput = [[GPUImageRawDataOutput alloc] initWithImageSize:CGSizeMake(480, 640)
        resultsInBGRAFormat:YES];
        self.videoCapturer = [[GPUImageVideoCapturer alloc] init];

        [self.rawDataOutput setNewFrameAvailableBlock:^(
            [_videoCapturer processNewFrame:_rawDataOutput];
        )];
        [self.videoCamera addTarget:_rawDataOutput];
        [self.videoCamera addTarget:_localDisplay];
    }
    ...
}
```

4. Публикация потока

FPWCSApi2Session createStream, FPWCSApi2Stream publish

[code](#)

Методу `createStream` передаются параметры:

- имя публикуемого потока
- вид для локального отображения
- источник захвата видео

Если вызов `publish` был успешным, запускается захват видео при помощи функции `startCameraCapture`

```
- (FPWCSPi2Stream *)publishStream {
    ...
    options.display = _localNativeDisplay;
    options.constraints = [[FPWCSPi2MediaConstraints alloc] initWithAudio:YES videoCapturer:self.
videoCapturer];

    NSError *error;
    FPWCSPi2Stream *stream = [session createStream:options error:&error];
    ...
    if(![stream publish:&error]) {
        UIAlertController * alert = [UIAlertController
            alertControllerWithTitle:@"Failed to publish"
            message:error.localizedDescription
            preferredStyle:UIAlertControllerStyleAlert];

        UIAlertAction* okButton = [UIAlertAction
            actionWithTitle:@"Ok"
            style:UIAlertActionStyleDefault
            handler:^(UIAlertAction * action) {
                [self onUnpublished];
            }];

        [alert addAction:okButton];
        [self presentViewController:alert animated:YES completion:nil];
    }

    [self.videoCamera startCameraCapture];

    return stream;
}
```

5. Включение и отключение фильтра бьютификации

[code](#)

```
- (void)beautify:(UIButton *)button {
    if (self.beautifyEnabled) {
        self.beautifyEnabled = NO;
        [_beautyButton setTitle:@"Beautify Enable" forState:UIControlStateNormal];
        [self.videoCamera removeAllTargets];
        [self.videoCamera addTarget:_rawDataOutput];
        [self.videoCamera addTarget:_localDisplay];
    }
    else {
        self.beautifyEnabled = YES;
        [_beautyButton setTitle:@"Beautify Disable" forState:UIControlStateNormal];
        [self.videoCamera removeAllTargets];
        GPUImageBeautifyFilter *beautifyFilter = [[GPUImageBeautifyFilter alloc] init];
        [self.videoCamera addTarget:beautifyFilter];
        [beautifyFilter addTarget:_localDisplay];
        [beautifyFilter addTarget:_rawDataOutput];
    }
}
```

6. Воспроизведение видеопотока.

`FPWCSPi2Session createStream, FPWCSPi2Stream play`

[code](#)

Методу `createStream` передаются параметры:

- имя воспроизводимого потока
- вид для отображения потока

```
- (FPWCSApi2Stream *)playStream {
    FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = _remoteStreamName.text;
    options.display = _remoteDisplay;
    NSError *error;
    FPWCSApi2Stream *stream = [session createStream:options error:nil];
    ...
    if(![stream play:&error]) {
        UIAlertController * alert = [UIAlertController
            alertControllerWithTitle:@"Failed to play"
            message:error.localizedDescription
            preferredStyle:UIAlertControllerStyleAlert];

        UIAlertAction* okButton = [UIAlertAction
            initWithTitle:@"Ok"
            style:UIAlertActionStyleDefault
            handler:^(UIAlertAction * action) {

            }]];

        [alert addAction:okButton];
        [self presentViewController:alert animated:YES completion:nil];
    }
    return stream;
}
```

7. Остановка воспроизведения видеопотока.

FPWCSApi2Stream stop

[code](#)

```
- (void)playButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        if ([FPWCSApi2 getSessions].count) {
            FPWCSApi2Stream *stream;
            for (FPWCSApi2Stream *s in [[FPWCSApi2 getSessions][0] getStreams]) {
                if ([s getName] isEqualToString:_remoteStreamName.text) {
                    stream = s;
                    break;
                }
            }
            if (!stream) {
                NSLog(@"Stop playing, nothing to stop");
                [self onStopped];
                return;
            }
            NSError *error;
            [stream stop:&error];
        } else {
            NSLog(@"Stop playing, no session");
            [self onStopped];
        }
        ...
    }
}
```

8. Остановка публикации видеопотока.

FPWCSApi2Stream stop

[code](#)

```

- (void)publishButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        if ([FPWCSApi2 getSessions].count) {
            FPWCSApi2Stream *stream;
            for (FPWCSApi2Stream *s in [[FPWCSApi2 getSessions][0] getStreams]) {
                if ([s getName] isEqualToString:_localStreamName.text) {
                    stream = s;
                    break;
                }
            }
            if (!stream) {
                NSLog(@"Stop publishing, nothing to stop");
                [self onUnpublished];
                return;
            }
            NSError *error;
            [stream stop:&error];
        } else {
            NSLog(@"Stop publishing, no session");
            [self onUnpublished];
        }
        ...
    }
}

```

9. Закрытие соединения.

FPWCSApi2Session disconnect

[code](#)

```

- (void)connectButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"DISCONNECT"]) {
        if ([FPWCSApi2 getSessions].count) {
            FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
            NSLog(@"Disconnect session with server %@", [session getServerUrl]);
            [session disconnect];
        } else {
            NSLog(@"Nothing to disconnect");
            [self onDisconnected];
        }
        ...
    }
}

```