iOS Video Conference

Example of video conference client for iOS

This example can be used to participate in video conference for three participants on Web Call Server and allows to publish WebRTC stream.

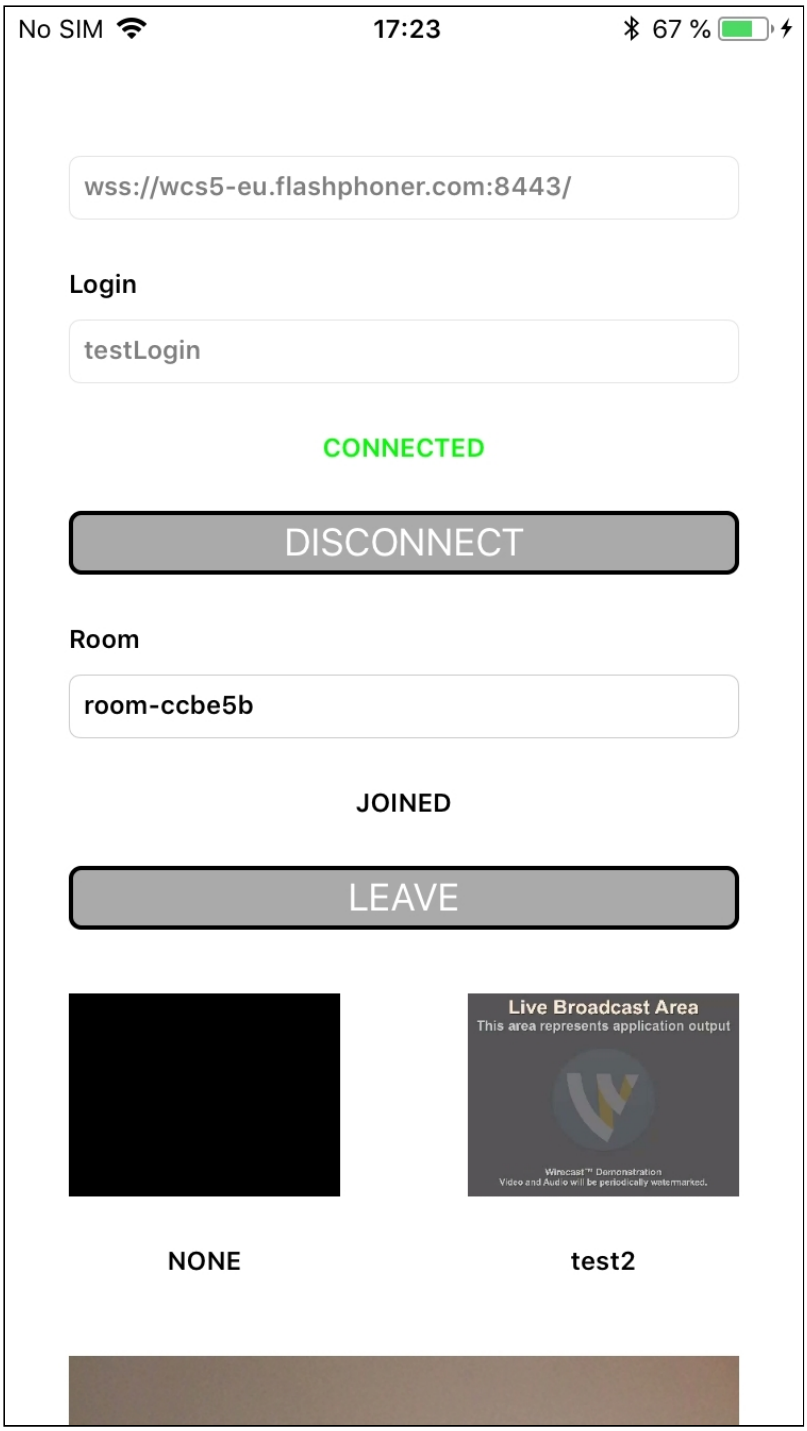
On the screenshot below the participant is connected, publishing a stream and playing streams from the other two participants.

Input fields required for connecting to WCS server and joining conference

- `WCS URL` is the address of the WCS server
- `Login` is the username
- `Room` is the name of conference room

Three videos are played

- video from the camera of this participant - the lower one
- videos from the other participants



Analyzing the example code

To analyze the code, let's take Conference example, which is available here.

class for the main view of the application: `ViewController` (header file ViewController.h; implementation file ViewController.m).

1. Import of API

code

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

2. Connection to the server

`FPWCSApi2.createRoomManager` code

`FPWCSApi2RoomManagerOptions` object with the following parameters is passed to `createRoomManager` method

- URL of WCS server
- username

```
- (void)connect {
    FPWCSApi2RoomManagerOptions *options = [[FPWCSApi2RoomManagerOptions
alloc] init];
    options.urlServer = _connectUrl.text;
    options.username = _connectLogin.input.text;
    NSError *error;
    roomManager = [FPWCSApi2 createRoomManager:options error:&error];
    ...
}
```

3. Joining a conference

`FPWCSApi2RoomManager.join` code

`FPWCSApi2RoomOptions` object with the name of the conference room is passed to the `join` method

```
FPWCSApi2RoomOptions * options = [[FPWCSApi2RoomOptions alloc] init];
options.name = _joinRoomName.input.text;
room = [roomManager join:options];
```

4. Receiving the event describing chat room state

`FPWCSApi2Room.onStateCallback` code

On this event:

- the size of the collection of `Participant` objects returned by method `getParticipants` is determined to get the number of already connected participants
- if the maximum allowed number of participants had already been reached, the user leaves the room

- otherwise, appropriate changes in the interface are done and playback of video stream published by the other participants is started

```
[room onStateCallback:^(FPWCSApi2Room *room) {
    NSDictionary *participants = [room getParticipants];
    if ([participants count] >= 3) {
        [room leave:nil];
        _joinStatus.text = @"Room is full";
        [self changeViewState:_joinButton enabled:YES];
        return;

    }
    NSString *chatState = @"participants: ";
    for (NSString* key in participants) {
        FPWCSApi2RoomParticipant *participant = [participants
valueForKey:key];
        ParticipantView *pv = [freeViews pop];
        [busyViews setValue:pv forKey:[participant getName]];
        [participant play:pv.display];
        pv.login.text = [participant getName];
        chatState = [NSString stringWithFormat:@"%@%@, ", chatState,
[participant getName]];
    }
    ...
}];
```

5. Video stream publishing

`FPWCSApi2Room.publish` code

The following stream options is passed to `publish` method:

- view to display video

```
- (void)publishButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        [room unpublish];
    } else {
        publishStream = [room publish:_localDisplay];
        ...
    }
}
```

6. Receiving the event notifying that other participant joined to the room

`FPWCSApi2Room.kFPWCSRoomParticipantEventJoined` code

```
[room on:kFPWCSRoomParticipantEventJoined participantCallback:^(FPWCSApi2Room
*room, FPWCSApi2RoomParticipant *participant) {
    ParticipantView *pv = [freeViews pop];
    if (pv) {
```

```
        pv.login.text = [participant getName];
        _messageHistory.text = [NSString stringWithFormat:@"%@\n%@ - %@",
_messageHistory.text, participant.getName, @"joined"];
        [busyViews setValue:pv forKey:[participant getName]];
    }
}];
```

7. Receiving the event notifying that other participant published video stream

`FPWCSApi2Room.kFPWCSRoomParticipantEventPublished`,
`FPWCSApi2RoomParticipant.play` code

On this event, video stream from other participant playback is started

```
FPWCSApi2Room kFPWCSRoomParticipantEventPublished participantCallback,
FPWCSApi2RoomParticipant play код

[room on:kFPWCSRoomParticipantEventPublished
participantCallback:^(FPWCSApi2Room *room, FPWCSApi2RoomParticipant
*participant) {
    ParticipantView *pv = [busyViews valueForKey:[participant getName]];
    if (pv) {
        [participant play:pv.display];
    }
}];
```

8. Receiving the event notifying that other participant sent a message

`FPWCSApi2Room.onMessageCallback` code

```
[room onMessageCallback:^(FPWCSApi2Room *room, FPWCSApi2RoomMessage *message)
{
    _messageHistory.text = [NSString stringWithFormat:@"%@\n%@ - %@",
_messageHistory.text, message.from, message.text];
}];
```

9. Sending a message to other room participants

`FPWCSApi2RoomParticipant.sendMessage` code

The message text is passed to the metod

```
- (void)sendButton:(UIButton *)button {
    for (NSString *name in [room getParticipants]) {
        FPWCSApi2RoomParticipant *participant = [room getParticipants][name];
        [participant sendMessage:_messageBody.text];
    }
    _messageHistory.text = [NSString stringWithFormat:@"%@\n%@ - %@",
_messageHistory.text, _connectLogin.input.text, _messageBody.text];
```

```
    _messageBody.text = @"";
}
```

10. Mute/unmute audio and video for stream published

`FPWCSApi2Stream.unmuteAudio`, `FPWCSApi2Stream.muteAudio`, `FPWCSApi2Stream.unmuteVideo`, `FPWCSApi2Stream.muteVideo` code

```
- (void)muteAudioChanged:(id)sender {
    if (publishStream) {
        if (_muteAudio.control.isOn) {
            [publishStream muteAudio];
        } else {
            [publishStream unmuteAudio];
        }
    }
}

- (void)muteVideoChanged:(id)sender {
    if (publishStream) {
        if (_muteVideo.control.isOn) {
            [publishStream muteVideo];
        } else {
            [publishStream unmuteVideo];
        }
    }
}
```

11. Stop stream publishing

`FPWCSApi2Room.unpublish` code

```
- (void)publishButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        [room unpublish];
    } else {
        ...
    }
}
```

12. Leaving chat room

`FPWCSApi2Room.leave` код

Server REST hook app response handler function is passed to the method.

```
if ([button.titleLabel.text isEqualToString:@"LEAVE"]) {
    if (room) {
        FPWCSApi2DataHandler *handler = [[FPWCSApi2DataHandler alloc] init];
```

```
        handler.onAccepted = ^(FPWCSApi2Session *session, FPWCSApi2Data
*data){
            [self onUnpublished];
            [self onLeaved];
        };
        handler.onRejected = ^(FPWCSApi2Session *session, FPWCSApi2Data
*data){
            [self onUnpublished];
            [self onLeaved];
        };
        [room leave:handler];
        room = nil;
    }
}
```

13. Disconnection

`FPWCSApi2RoomManager.disconnect` code

```
- (void)connectButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"DISCONNECT"]) {
        if (roomManager) {
            [roomManager disconnect];
        }
        ...
    }
}
```