User Generated Short Video SDK

Advanced Features and Special Effects

Product Documentation

Contents

Advanced Features and Special Effects
TikTok-like Special Effects
iOS

Last updated：2021-09-15 17:15:24

Special Effect Filter

You can add multiple special effect filters for your videos. Currently, 11 filters are supported, all of which allow you to set the start and end time for display in the video. If multiple filters are set at the same point in time, the SDK will display the last set one.

You can set a special effect as follows:

```
- (void) startEffect:(TXEffectType)type startTime:(float)startTime;
- (void) stopEffect:(TXEffectType)type endTime:(float)endTime;
// The special effect type (`type` parameter) is defined in the `TXEffectType` co
nstant:
typedef NS_ENUM(NSInteger,TXEffectType)
{
TXEffectType_ROCK_LIGHT, // Dynamic light-wave
TXEffectType_DARK_DRAEM, // Dark dream
TXEffectType_SOUL_OUT, // Soul out
TXEffectType_SCREEN_SPLIT,// Screen split
TXEffectType_WIN_SHADOW, // Window blinds
TXEffectType_GHOST_SHADOW,// Ghost shadow
TXEffectType_PHANTOM, // Phantom
TXEffectType_GHOST, // Ghost
TXEffectType_LIGHTNING, // Lightening
TXEffectType_MIRROR, // Mirror
TXEffectType_ILLUSION, // Illusion
};
- (void) deleteLastEffect;
- (void) deleteAllEffect;
```

You can call `deleteLastEffect()` to delete the last set special effect filter.
You can call `deleteAllEffect()` to delete all set special effect filters:

Demo:
Use the first special effect filter between the first and second seconds, use the second special effect filter between the third and fourth seconds, and delete the special effect filter set between the third and fourth seconds:

```
// Use the first special effect filter between the first and second seconds
[_ugcEdit startEffect:TXEffectType_SOUL_OUT startTime:1.0];
[_ugcEdit stopEffect:TXEffectType_SOUL_OUT startTime:2.0)];
// Use the second special effect filter between the third and fourth seconds
[_ugcEdit startEffect:TXEffectType_SPLIT_SCREEN startTime:3.0];
[_ugcEdit stopEffect:TXEffectType_SPLIT_SCREEN startTime:4.0];
// Delete the special effect filter set between the third and fourth seconds
[_ugcEdit deleteLastEffect];
```

Slow/Fast Motions

You can change the playback speed of multiple video segments by setting slow/fast playback as follows:

```
- (void) setSpeedList:(NSArray *)speedList;
// The `TXSpeed` parameters are as follows:
@interface TXSpeed: NSObject
@property (nonatomic, assign) CGFloat startTime; // Speed change start time in s
@property (nonatomic, assign) CGFloat endTime; // Speed change end time in s
@property (nonatomic, assign) TXSpeedLevel speedLevel; // Speed change level
@end
Currently, multiple speed change levels are supported, which are defined in the `
TXSpeedLevel` constant:
typedef NS_ENUM(NSInteger, TXSpeedLevel) {
SPEED_LEVEL_SLOWEST, // Ultra-slow
SPEED_LEVEL_SLOW, // Slow
SPEED_LEVEL_NOMAL, // Normal
SPEED_LEVEL_FAST, // Fast
SPEED_LEVEL_FASTEST, // Ultra-fast
};
```

Demo:

```
// The SDK supports speed change of multiple video segments. This demo only shows
slow playback of one video segment.
TXSpeed *speed =[[TXSpeed alloc] init];
speed.startTime = 1.0;
speed.endTime = 3.0;
speed.speedLevel = SPEED_LEVEL_SLOW;
[_ugcEdit setSpeedList:@[speed]];
```

Reverse Playback

You can reverse a video as follows:

```
- (void) setReverse:(BOOL)isReverse;
```

Demo:

```
[_ugcEdit setReverse:YES];
```

Video Segment Loop

You can loop a video segment, but the audio will not be looped.

Set the video segment for loop as follows:

```
- (void) setRepeatPlay:(NSArray *)repeatList;
// The `TXRepeat` parameters are as follows:
@interface TXRepeat: NSObject
@property (nonatomic, assign) CGFloat startTime; // Loop start time in s
@property (nonatomic, assign) CGFloat endTime; // Loop end time in s
@property (nonatomic, assign) int repeatTimes; // Number of repeats
@end
```

Demo:

```
TXRepeat *repeat = [[TXRepeat alloc] init];
repeat.startTime = 1.0;
repeat.endTime = 3.0;
repeat.repeatTimes = 3; // Number of repeats
[_ugcEdit setRepeatPlay:@[repeat]];
```

Android

Last updated：2020-09-01 14:52:06

Special Effect Filter

You can add multiple special effect filters for your videos. Currently, 11 filters are supported, all of which allow you to set the start and end time for display in the video. If multiple filters are set at the same point in time, the SDK will display the last set one.

Set the special effect filter:

```
/**
* Set the start time of the special effect filter
* @param type Special effect filter type
* @param startTime Start time of special effect filter in ms
*/
public void startEffect(int type, long startTime);
/**
* Set the end time of the special effect filter
* @param type Special effect filter type
* @param endTime End time of special effect filter in ms
*/
public void stopEffect(int type, long endTime);
```

Parameter description: @param type: special effect filter type, which is defined in the `TXVideoEditConstants` constant:

```
public static final int TXEffectType_SOUL_OUT = 0; // Filter 1
public static final int TXEffectType_SPLIT_SCREEN = 1; // Filter 2
public static final int TXEffectType_DARK_DRAEM = 2; // Filter 3
public static final int TXEffectType_ROCK_LIGHT = 3; // Filter 4
public static final int TXEffectType_WIN_SHADDOW = 4; // Filter 5
public static final int TXEffectType_GHOST_SHADDOW = 5; // Filter 6
public static final int TXEffectType_PHANTOM_SHADDOW = 6; // Filter 7
public static final int TXEffectType_GHOST = 7; // Filter 8
public static final int TXEffectType_LIGHTNING = 8; // Filter 9
public static final int TXEffectType_MIRROR = 9; // Filter 10
public static final int TXEffectType_ILLUSION = 10; // Filter 11
```

Delete the last set special effect filter:

```
public void deleteLastEffect();
```

Delete all set special effect filters:

```
public void deleteAllEffect();
```

Below is a complete sample:

Use the first special effect filter between the first and second seconds, use the second special effect filter between the third and fourth seconds, and delete the special effect filter set between the third and fourth seconds:

```
// Use the first special effect filter between the first and second seconds
mTXVideoEditer.startEffect(TXVideoEditConstants.TXEffectType_SOUL_OUT, 1000);
mTXVideoEditer.stopEffect(TXVideoEditConstants.TXEffectType_SOUL_OUT, 2000);
// Use the second special effect filter between the third and fourth seconds
mTXVideoEditer.startEffect(TXVideoEditConstants.TXEffectType_SPLIT_SCREEN, 3000);
mTXVideoEditer.stopEffect(TXVideoEditConstants.TXEffectType_SPLIT_SCREEN, 4000);
// Delete the special effect filter set between the third and fourth seconds
mTXVideoEditer.deleteLastEffect();
```

Slow/Fast Motions

You can change the playback speed of multiple video segments by setting slow/fast playback as follows:

```
public void setSpeedList(List speedList);

// The `TXSpeed` parameters are as follows:
public final static class TXSpeed {
public int speedLevel; // Speed change level
public long startTime; // Start time
public long endTime; // End time
}

// Currently, multiple speed change levels are supported, which are defined in th
e `TXVideoEditConstants` constant:
SPEED_LEVEL_SLOWEST - Ultra-slow
SPEED_LEVEL_SLOW - Slow
SPEED_LEVEL_NORMAL - Normal
SPEED_LEVEL_FAST - Fast
SPEED_LEVEL_FASTEST - Ultra-fast
```

Below is a complete sample:

```
List<TXVideoEditConstants.TXSpeed> list = new ArrayList<>();
TXVideoEditConstants.TXSpeed speed1 = new TXVideoEditConstants.TXSpeed();
```

```
speed1.startTime = 0;
speed1.endTime = 1000;
speed1.speedLevel = TXVideoEditConstants.SPEED_LEVEL_SLOW; // Slow
list.add(speed1);

TXVideoEditConstants.TXSpeed speed2 = new TXVideoEditConstants.TXSpeed();
speed2.startTime = 1000;
speed2.endTime = 2000;
speed2.speedLevel = TXVideoEditConstants.SPEED_LEVEL_SLOWEST; // Ultra-slow
list.add(speed2);

TXVideoEditConstants.TXSpeed speed3 = new TXVideoEditConstants.TXSpeed();
speed3.startTime = 2000;
speed3.endTime = 3000;
speed3.speedLevel = TXVideoEditConstants.SPEED_LEVEL_SLOW; // Slow
list.add(speed3);

mTXVideoEditer.setSpeedList(list);
```

Reverse Playback

You can reverse video playback. Specifically, you can call `setReverse(true)` / `setReverse(false)` to start/stop reverse playback.

> ⚠ **Note：**
>
> The legacy version of **setTXVideoReverseListener()** (below SDK 4.5) needs to be manually called for the first listening, while the new version can take effect without being called.

Demo:

```
mTXVideoEditer.setTXVideoReverseListener(mTxVideoReverseListener);
mTXVideoEditer.setReverse(true);
```

Video Segment Loop

You can loop a video segment, but the audio will not be looped. Currently, Android supports loop of only one video segment thrice.
You can call `setRepeatPlay(null)` to cancel the video segment loop set previously.

Set the video segment for loop as follows:

```
public void setRepeatPlay(List repeatList);

// The `TXRepeat` parameters are as follows:
public final static class TXRepeat {
public long startTime; // Loop start time in ms
public long endTime; // Loop end time in ms
public int repeatTimes; // Number of repeats
}
```

Demo:

```
long currentPts = mVideoProgressController.getCurrentTimeMs();

List repeatList = new ArrayList<>();
TXVideoEditConstants.TXRepeat repeat = new TXVideoEditConstants.TXRepeat();
repeat.startTime = currentPts;
repeat.endTime = currentPts + DEAULT_DURATION_MS;
repeat.repeatTimes = 3; // Currently, a video segment can be repeated only for th
rice.
repeatList.add(repeat); // Currently, only one video segment can be looped.
mTXVideoEditer.setRepeatPlay(repeatList);
```

Stickers and Subtitles
iOS

Last updated：2020-09-01 14:52:07

Static Sticker

```
- (void) setPasterList:(NSArray *)pasterList;

// The `TXPaster` parameters are as follows:
@interface TXPaster: NSObject
@property (nonatomic, strong) UIImage* pasterImage; // Sticker image
@property (nonatomic, assign) CGRect frame; // Sticker frame (please note that th
e frame coordinates here are relative to the rendering view)
@property (nonatomic, assign) CGFloat startTime; // Sticker start time in s
@property (nonatomic, assign) CGFloat endTime; // Sticker end time in s
@end
```

Animated Sticker

```
- (void) setAnimatedPasterList:(NSArray *)animatedPasterList;

// The `TXAnimatedPaster` parameters are as follows:
@interface TXAnimatedPaster: NSObject
@property (nonatomic, strong) NSString* animatedPasterpath; // Animated image fil
e path
@property (nonatomic, assign) CGRect frame; // Animated image frame (please note
that the frame coordinates here are relative to the rendering view)
@property (nonatomic, assign) CGFloat rotateAngle; // Animated image rotation ang
le. Value range: 0-360
@property (nonatomic, assign) CGFloat startTime; // Animated image start time in
s
@property (nonatomic, assign) CGFloat endTime; // Animated image end time in s
@end
```

Demo:

```
- (void)setVideoPasters:(NSArray*)videoPasterInfos
{
```

```
NSMutableArray* animatePasters = [NSMutableArray new];
NSMutableArray* staticPasters = [NSMutableArray new];
for (VideoPasterInfo* pasterInfo in videoPasterInfos) {
if (pasterInfo.pasterInfoType == PasterInfoType_Animate) {
TXAnimatedPaster* paster = [TXAnimatedPaster new];
paster.startTime = pasterInfo.startTime;
paster.endTime = pasterInfo.endTime;
paster.frame = [pasterInfo.pasterView pasterFrameOnView:_videoPreview];
paster.rotateAngle = pasterInfo.pasterView.rotateAngle * 180 / M_PI;
paster.animatedPasterpath = pasterInfo.path;
[animatePasters addObject:paster];
}
else if (pasterInfo.pasterInfoType == PasterInfoType_static){
TXPaster *paster = [TXPaster new];
paster.startTime = pasterInfo.startTime;
paster.endTime = pasterInfo.endTime;
paster.frame = [pasterInfo.pasterView pasterFrameOnView:_videoPreview];
paster.pasterImage = pasterInfo.pasterView.staticImage;
[staticPasters addObject:paster];
}
}
[_ugcEditer setAnimatedPasterList:animatePasters];
[_ugcEditer setPasterList:staticPasters];
}
```

Custom Animated Sticker

Adding an animated sticker is actually to arrange **a series of images** in **a certain sequence** at **a certain interval** and insert them to the video to implement an animated sticker effect.

How do I customize a sticker?
Here, an animated sticker in the SDK demo is used as an example:

```
{
"name":"glass", // Sticker name
"count":6, // Number of stickers
"period":480, // Playback period: time taken for playing back the sticker once in
ms
"width":444, // Sticker width
"height":256, // Sticker height
"keyframe":6, // Key image, which can represent the animated sticker
"frameArray": [ // Set of all images
{"picture":"glass0"},
{"picture":"glass1"},
```

```
{"picture":"glass2"},
{"picture":"glass3"},
{"picture":"glass4"},
{"picture":"glass5"}
]
}
```

The SDK will get the corresponding `config.json` of the animated sticker and display it in the format defined in the .json file.

Note：

As this encapsulation format is required by the SDK, please describe animated stickers strictly in this format.

Adding Subtitles

1. Bubble subtitles

Video subtitling is supported. You can add subtitles to each frame of a video and set the start and end time to display each subtitle. All subtitles form a subtitle list, which can be passed to the SDK, and the SDK will automatically add the subtitles to the video at the corresponding points in time.

You can set subtitles as follows:

```
- (void) setSubtitleList:(NSArray *)subtitleList;

The `TXSubtitle` parameters are as follows:
@interface TXSubtitle: NSObject
@property (nonatomic, strong) UIImage* titleImage; // Subtitle image (here, you n
eed to convert the text loading control to an image)
@property (nonatomic, assign) CGRect frame; // Subtitle frame (please note that t
he frame coordinates here are relative to the rendering view)
@property (nonatomic, assign) CGFloat startTime; // Subtitle start time in s
@property (nonatomic, assign) CGFloat endTime; // Subtitle end time in s
@end
```

- titleImage: subtitle image. If controls like `UILabel` are used by the upper layer, please convert the control to `UIImage` first. For detailed directions, please see the sample code of the demo.
- frame: subtitle frame (please note that the frame is relative to the frame of the rendering view passed in during `initWithPreview`). For more information, please see the sample code of the demo.
- startTime: subtitle start time.
- endTime: subtitle end time.

As the subtitle UI logic is complicated, a complete method is provided at the demo layer. We recommend you directly implement subtitling as instructed in the demo, which greatly reduces your integration costs.

Demo:

```
@interface VideoTextInfo : NSObject
@property (nonatomic, strong) VideoTextFiled* textField;
@property (nonatomic, assign) CGFloat startTime; //in seconds
@property (nonatomic, assign) CGFloat endTime;
@end

videoTextInfos = @[VideoTextInfo1, VideoTextInfo2 ...];

for (VideoTextInfo* textInfo in videoTextInfos) {
TXSubtitle* subtitle = [TXSubtitle new];
subtitle.titleImage = textInfo.textField.textImage; //UILabel (UIView) -> UIImage
subtitle.frame = [textInfo.textField textFrameOnView:_videoPreview]; // Calculate
the coordinates relative to the rendering view
subtitle.startTime = textInfo.startTime; // Subtitle start time
subtitle.endTime = textInfo.endTime; // Subtitle end time
[subtitles addObject:subtitle]; // Add the subtitle list
}

[_ugcEditer setSubtitleList:subtitles]; // Set the subtitle list
```

2. How do I customize bubble subtitles?

Parameters required by bubble subtitles

- Font area dimensions: top, left, right, bottom
- Default font size
- Width and height

> **Note：**
>
> The parameters above are all in px.

Encapsulation format

As bubble subtitles contain many parameters, we recommend you encapsulate relevant parameters at the demo layer. For example, the Tencent Cloud demo uses the JSON format for encapsulation:

```
{
"name":"boom", // Bubble subtitle name
```

```
"width": 376, // Width
"height": 335, // Height
"textTop":121, // Top margin of text area
"textLeft":66, // Left margin of text area
"textRight":69, // Right margin of text area
"textBottom":123, // Bottom margin of text area
"textSize":40 // Font size
}
```

Note：

You can determine the encapsulation format by yourself, which is optional for the SDK.

Overlong subtitle

If a subtitle is too long, how do I arrange the layout to make the subtitle neatly fit in the bubble?

An automatic layout control is provided in the demo. If a subtitle is too long, the control will automatically decrease the font size until the entire subtitle can fit in the bubble.

You can also modify the relevant control source code to meet your unique business needs.

Android

Last updated：2020-09-01 14:52:08

Static Sticker

You can set a static sticker as follows:

```
public void setPasterList(List pasterList);

// The `TXPaster` parameters are as follows:
public final static class TXPaster {
public Bitmap pasterImage; // Sticker image
public TXRect frame; // Sticker frame (please note that the frame coordinates her
e are relative to the rendering view)
public long startTime; // Sticker start time in ms
public long endTime; // Sticker end time in ms
}
```

Animated Sticker

You can set an animated sticker as follows:

```
public void setAnimatedPasterList(List animatedPasterList);

// The `TXAnimatedPaster` parameters are as follows:
public final static class TXAnimatedPaster {
public String animatedPasterPathFolder; // Address of animated sticker image
public TXRect frame; // Animated sticker frame (please note that the frame coordi
nates here are relative to the rendering view)
public long startTime; // Animated sticker start time in ms
public long endTime; // Animated sticker end time in ms
public float rotation;
}
```

Demo:

```
List animatedPasterList = new ArrayList<>();
List pasterList = new ArrayList<>();
for (int i = 0; i < mTCLayerViewGroup.getChildCount(); i++) {
PasterOperationView view = (PasterOperationView) mTCLayerViewGroup.getOperationVi
```

```
ew(i);
TXVideoEditConstants.TXRect rect = new TXVideoEditConstants.TXRect();
rect.x = view.getImageX();
rect.y = view.getImageY();
rect.width = view.getImageWidth();
TXCLog.i(TAG, "addPasterListVideo, adjustPasterRect, paster x y = " + rect.x +
"," + rect.y);

int childType = view.getChildType();
if (childType == PasterOperationView.TYPE_CHILD_VIEW_ANIMATED_PASTER) {
TXVideoEditConstants.TXAnimatedPaster txAnimatedPaster = new TXVideoEditConstant
s.TXAnimatedPaster();

txAnimatedPaster.animatedPasterPathFolder = mAnimatedPasterSDcardFolder + view.ge
tPasterName() + File.separator;
txAnimatedPaster.startTime = view.getStartTime();
txAnimatedPaster.endTime = view.getEndTime();
txAnimatedPaster.frame = rect;
txAnimatedPaster.rotation = view.getImageRotate();

animatedPasterList.add(txAnimatedPaster);
TXCLog.i(TAG, "addPasterListVideo, txAnimatedPaster startTimeMs, endTime is : " +
txAnimatedPaster.startTime + ", " + txAnimatedPaster.endTime);
} else if (childType == PasterOperationView.TYPE_CHILD_VIEW_PASTER) {
TXVideoEditConstants.TXPaster txPaster = new TXVideoEditConstants.TXPaster();

txPaster.pasterImage = view.getRotateBitmap();
txPaster.startTime = view.getStartTime();
txPaster.endTime = view.getEndTime();
txPaster.frame = rect;

pasterList.add(txPaster);
TXCLog.i(TAG, "addPasterListVideo, txPaster startTimeMs, endTime is : " + txPaste
r.startTime + ", " + txPaster.endTime);
}
}

mTXVideoEditer.setAnimatedPasterList(animatedPasterList); // Set an animated stic
ker
mTXVideoEditer.setPasterList(pasterList); // Set a static sticker
```

Custom Animated Sticker

Adding an animated sticker is actually to arrange **a series of images** in **a certain sequence** at **a certain interval** and insert them to the video to implement an animated sticker effect.

Encapsulation format

Here, an animated sticker in the demo is used as an example:

```
{
"name":"glass", // Sticker name
"count":6, // Number of stickers
"period":480, // Playback period: time taken for playing back the sticker once in ms
"width":444, // Sticker width
"height":256, // Sticker height
"keyframe":6, // Key image, which can represent the animated sticker
"frameArray": [ // Set of all images
{"picture":"glass0"},
{"picture":"glass1"},
{"picture":"glass2"},
{"picture":"glass3"},
{"picture":"glass4"},
{"picture":"glass5"}
]
}
```

The SDK will get the corresponding `config.json` of the animated sticker and display it in the format defined in the .json file.

Note：

As this encapsulation format is required by the SDK, please describe animated stickers strictly in this format.

Adding Subtitles

1. Bubble subtitles

Bubble video subtitling is supported. You can add subtitles to each frame of a video and set the start and end time to display each subtitle. All subtitles form a subtitle list, which can be passed to the SDK, and the SDK will automatically add the subtitles to the video at the corresponding points in time.

You can set bubble subtitles as follows:

```
public void setSubtitleList(List subtitleList);
```

```
// The `TXSubtitle` parameters are as follows:
public final static class TXSubtitle {
public Bitmap titleImage; // Subtitle image
public TXRect frame; // Subtitle frame
public long startTime; // Subtitle start time in ms
public long endTime; // Subtitle end time in ms
}

public final static class TXRect {
public float x;
public float y;
public float width;
}
```

- titleImage: subtitle image. If controls like `TextView` are used by the upper layer, please convert the control to `Bitmap` first. For detailed directions, please see the sample code of the demo.
- frame: subtitle frame (please note that the frame is relative to the frame of the rendering view passed in during `initWithPreview`). For more information, please see the sample code of the demo.
- startTime: subtitle start time.
- endTime: subtitle end time.

As the subtitle UI logic is complicated, a complete method is provided at the demo layer. We recommend you directly implement subtitling as instructed in the demo, which greatly reduces your integration costs.

Demo:

```
mSubtitleList.clear();
for (int i = 0; i < mWordInfoList.size(); i++) {
TCWordOperationView view = mOperationViewGroup.getOperationView(i);
TXVideoEditConstants.TXSubtitle subTitle = new TXVideoEditConstants.TXSubtitle();
subTitle.titleImage = view.getRotateBitmap(); // Get `Bitmap`
TXVideoEditConstants.TXRect rect = new TXVideoEditConstants.TXRect();
rect.x = view.getImageX(); // Get the X coordinate relative to the parent view
rect.y = view.getImageY(); // Get the Y coordinate relative to the parent view
rect.width = view.getImageWidth(); // Image width
subTitle.frame = rect;
subTitle.startTime = mWordInfoList.get(i).getStartTime(); // Set the start time
subTitle.endTime = mWordInfoList.get(i).getEndTime(); // Set the end time
mSubtitleList.add(subTitle);
}
mTXVideoEditer.setSubtitleList(mSubtitleList); // Set the subtitle list
```

2. How do I customize bubble subtitles?

Parameters required by bubble subtitles

- Font area dimensions: top, left, right, bottom
- Default font size
- Width and height

> **Note：**
>
> The parameters above are all in px.

Encapsulation format

As bubble subtitles contain many parameters, we recommend you encapsulate relevant parameters at the demo layer. For example, the Tencent Cloud demo uses the JSON format for encapsulation:

```
{
"name":"boom", // Bubble subtitle name
"width": 376, // Width
"height": 335, // Height
"textTop":121, // Top margin of text area
"textLeft":66, // Left margin of text area
"textRight":69, // Right margin of text area
"textBottom":123, // Bottom margin of text area
"textSize":40 // Font size
}
```

> **Note：**
>
> You can determine the encapsulation format by yourself, which is optional for the SDK.

Overlong subtitle

If a subtitle is too long, how do I arrange the layout to make the subtitle neatly fit in the bubble?

An automatic layout control is provided in the demo. If a subtitle is too long, the control will automatically decrease the font size until the entire subtitle can fit in the bubble.

You can also modify the relevant control source code to meet your unique business needs.

Video Karaoke
iOS

Last updated：2020-09-01 14:58:16

This document describes how to implement basic duet features from scratch.

Process Overview

1. Place two views on the page, one for playback, and the other for shoot.
2. Place a button and progress bar for shoot and progress display, respectively.
3. Stop shoot after the video in the same duration as that of the source video has been shot.
4. Compose the shot video with the source video side by side.
5. Preview the composed video.

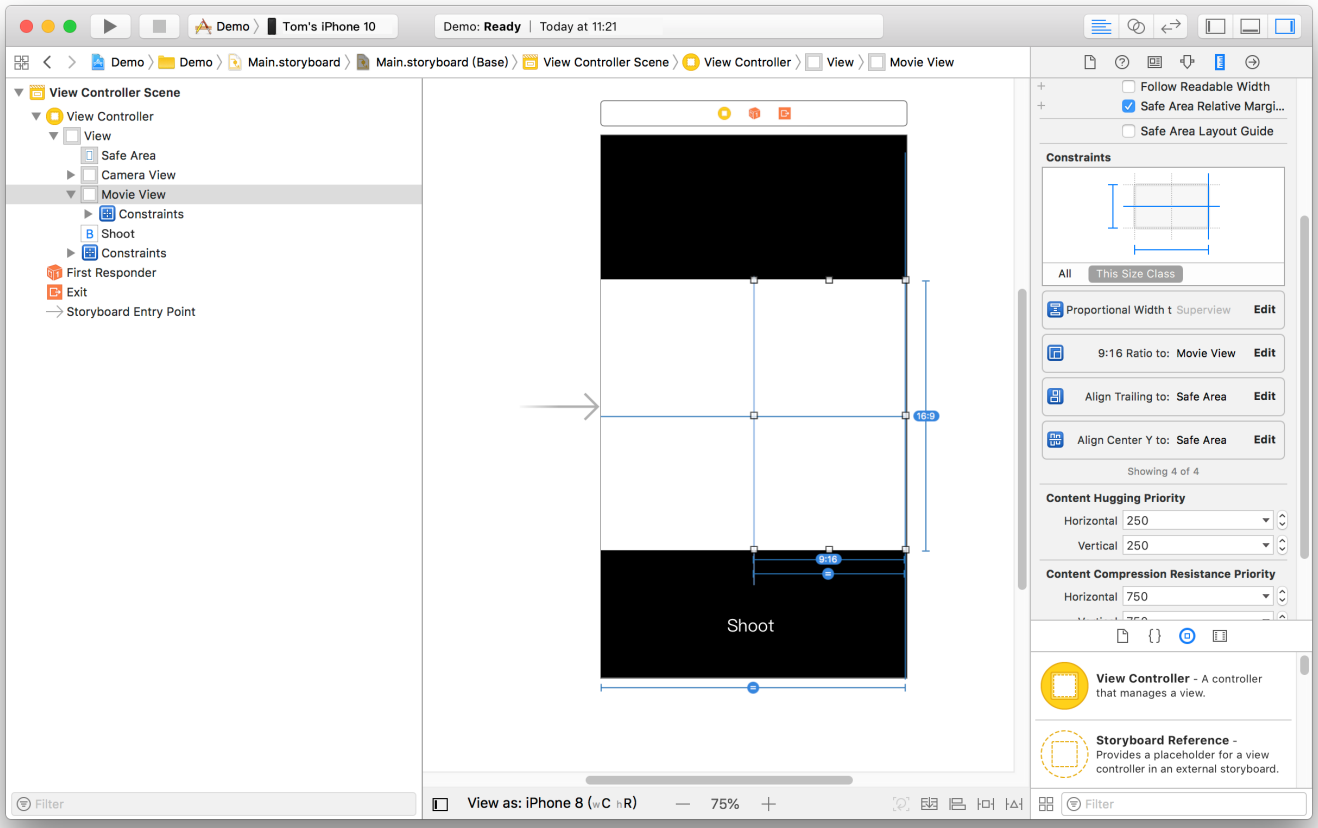
UI Construction

Create a project first. Open Xcode, select "File" > "New" > "Project", and name the project to create it. The project is named "Demo" in this example. To shoot a video, the camera and mic permissions are required. Add the following items to `Info` :

```
Privacy - Microphone Usage Description
Privacy - Camera Usage Description
```

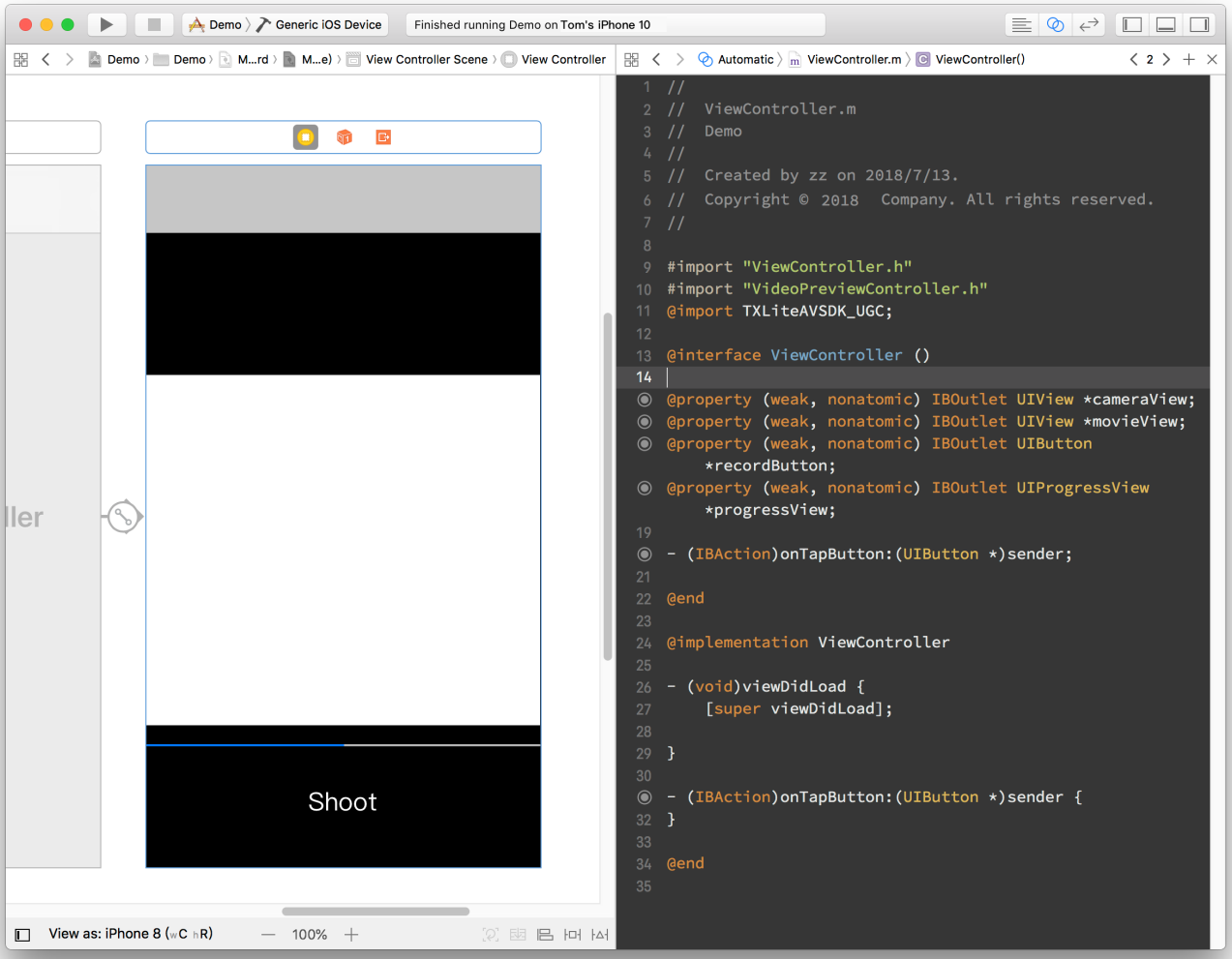
You can enter desired values for the two items, such as "Shooting Video"

Configure a simple shoot page. Open `Main.storyboard` , drag two `UIView` objects into it, configure their width to 0.5 time of the `superview` , and set their aspect ratio to 16:9.



Add the progress bar, bind the page to `IBOutlet` in `ViewController.m`, and set the button `IBAction`. As the preview page needs to be redirected to after shoot, a navigation controller is required. Click the "VC" icon in yellow, select "Editor" > "Embed In" in the menu, and click "Navigation Controller" to add a layer of "Navigation

Controller" onto the "ViewController". At this point, the basic UI has been constructed.



Sample Code

The duet feature mainly uses three other features: playback, shoot, and composition of the shot and source videos, which correspond to the `TXVideoEditer`, `TXUGCRecord`, and `TXVideoJoiner` SDK classes, respectively.

The SDK license needs to be configured before this feature can be used. Open `AppDelegate.m` and add the following code to it:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
[TXUGCBase setLicenceURL:@"<License URL>" key:@"<License key>"];
```

```
return YES;
}
```

Here, you need to apply for the license parameters in the UGSV Console. Once submitted, your application will be generally approved very soon, and the relevant information will be displayed on the page.

1. First, implement the declaration and initialization.
   Open `ViewContorller.m` , import the SDK, and declare the instances of the three classes above. As video playback, shoot, and composition are all async operations here, you need to listen on their events by adding the declaration for implementing the three protocols: `TXVideoJoinerListener` , `TXUGCRecordListener` , and `TXVideoPreviewListener` . After the declaration is added, the code will be as follows:

```
#import "ViewController.h"
@import TXLiteAVSDK_UGC;

@interface ViewController () <TXVideoJoinerListener, TXUGCRecordListener, TXVid
eoPreviewListener>
{
TXVideoEditer *_editor;
TXUGCRecord *_recorder;
TXVideoJoiner *_joiner;

TXVideoInfo *_videoInfo;

NSString *_recordPath;
NSString *_resultPath;
}

@property (weak, nonatomic) IBOutlet UIView *cameraView;
@property (weak, nonatomic) IBOutlet UIView *movieView;
@property (weak, nonatomic) IBOutlet UIButton *recordButton;
@property (weak, nonatomic) IBOutlet UIProgressView *progressView;

- (IBAction)onTapButton:(UIButton *)sender;
@end
```

   After preparing the member variables and API implementation declaration, initialize the member variables above in `viewDidLoad` .

```
- (void)viewDidLoad {
[super viewDidLoad];
// Here, place a .mp4 video file or a .mov video shot on the phone in the proje
ct
```

```
NSString *mp4Path = [[NSBundle mainBundle] pathForResource:@"demo" ofType:@"mp
4"];
_videoInfo = [TXVideoInfoReader getVideoInfo:mp4Path];
TXAudioSampleRate audioSampleRate = AUDIO_SAMPLERATE_48000;
if (_videoInfo.audioSampleRate == 8000) {
audioSampleRate = AUDIO_SAMPLERATE_8000;
}else if (_videoInfo.audioSampleRate == 16000){
audioSampleRate = AUDIO_SAMPLERATE_16000;
}else if (_videoInfo.audioSampleRate == 32000){
audioSampleRate = AUDIO_SAMPLERATE_32000;
}else if (_videoInfo.audioSampleRate == 44100){
audioSampleRate = AUDIO_SAMPLERATE_44100;
}else if (_videoInfo.audioSampleRate == 48000){
audioSampleRate = AUDIO_SAMPLERATE_48000;
}

// Set the video storage path
_recordPath = [NSTemporaryDirectory() stringByAppendingPathComponent:@"record.m
p4"];
_resultPath = [NSTemporaryDirectory() stringByAppendingPathComponent:@"result.m
p4"];

// Initialize the player
TXPreviewParam *param = [[TXPreviewParam alloc] init];
param.videoView = self.movieView;
param.renderMode = RENDER_MODE_FILL_EDGE;
_editor = [[TXVideoEditer alloc] initWithPreview:param];
[_editor setVideoPath:mp4Path];
_editor.previewDelegate = self;

// Initialize the shoot parameters
_recorder = [TXUGCRecord shareInstance];
TXUGCCustomConfig *recordConfig = [[TXUGCCustomConfig alloc] init];
recordConfig.videoResolution = VIDEO_RESOLUTION_720_1280;
// The frame rates of the shot video and source video must be the same; otherwi
se, the audio and video may be out of sync
// Note: the frame rate of the duet video obtained here is the average frame ra
te and may be a decimal, which needs to be rounded
recordConfig.videoFPS = (int)(_videoInfo.fps + 0.5);
// The audio sample rates of the shot video and source video must be the same;
otherwise, the audio and video may be out of sync
recordConfig.audioSampleRate = audioSampleRate;
recordConfig.videoBitratePIN = 9600;
recordConfig.maxDuration = _videoInfo.duration;
_recorder.recordDelegate = self;

// Enable camera preview
```

```
[_recorder startCameraCustom:recordConfig preview:self.cameraView];

// Compose videos
_joiner = [[TXVideoJoiner alloc] initWithPreview:nil];
_joiner.joinerDelegate = self;
[_joiner setVideoPathList:@[_recordPath, mp4Path]];
}
```

2. Next, implement the shoot feature. You only need to respond to the user click of the button to call the SDK method. For the sake of convenience, the button is reused here to display the current status, and the logic of displaying the progress is added to the progress bar.

```
- (IBAction)onTapButton:(UIButton *)sender {
[_editor startPlayFromTime:0 toTime:_videoInfo.duration];
if ([_recorder startRecord:_recordPath coverPath:[_recordPath stringByAppending
String:@".png"]] != 0) {
NSLog(@"Failed to start the camera");
}
[sender setTitle:@"Shooting" forState:UIControlStateNormal];
sender.enabled = NO;
}

#pragma mark TXVideoPreviewListener
-(void) onPreviewProgress:(CGFloat)time
{
self.progressView.progress = time / _videoInfo.duration;
}
```

3. After shoot, implement the composition. You need to specify the positions of the two videos in the output video. Here, the left and right positions are set.

```
-(void)onRecordComplete:(TXUGCRecordResult*)result;
{
NSLog(@"Shoot is completed and composition is started");
[self.recordButton setTitle:@"Composing..." forState:UIControlStateNormal];

// Get the width and height of the shot video
TXVideoInfo *videoInfo = [TXVideoInfoReader getVideoInfo:_recordPath];
CGFloat width = videoInfo.width;
CGFloat height = videoInfo.height;

// Place the shot and source videos on the left and right, respectively
CGRect recordScreen = CGRectMake(0, 0, width, height);
```

```
CGRect playScreen = CGRectMake(width, 0, width, height);
[_joiner setSplitScreenList:@[[NSValue valueWithCGRect:recordScreen],[NSValue v
alueWithCGRect:playScreen]] canvasWidth:width * 2 canvasHeight:height];
[_joiner splitJoinVideo:VIDEO_COMPRESSED_720P videoOutputPath:_resultPath];
}
```

4. Implement the delegation method of the composition progress to display the progress on the progress bar.

```
-(void) onJoinProgress:(float)progress
{
NSLog(@"Composing videos %d%%",(int)(progress * 100));
self.progressView.progress = progress;
}
```

5. Implement the delegation method of the composition completion and switch to the preview page.

```
#pragma mark TXVideoJoinerListener
-(void) onJoinComplete:(TXJoinerResult *)result
{
NSLog(@"Video composition completed");
VideoPreviewController *controller = [[VideoPreviewController alloc] initWithVi
deoPath:_resultPath];
[self.navigationController pushViewController:controller animated:YES];
}
```

At this point, the implementation is completed. The code of the video preview `VideoPreviewController` mentioned above is as follows:

- `VideoPreviewController.h`

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

@interface VideoPreviewController : UIViewController
- (instancetype)initWithVideoPath:(NSString *)path;
@end
```

- `VideoPreviewController.m:`

```
@import TXLiteAVSDK_UGC;
```

```
@interface VideoPreviewController () <TXVideoPreviewListener>
{
TXVideoEditer *_editor;
}
@property (strong, nonatomic) NSString *videoPath;
@end

@implementation VideoPreviewController

- (instancetype)initWithVideoPath:(NSString *)path {
if (self = [super initWithNibName:nil bundle:nil]) {
self.videoPath = path;
}
return self;
}

- (void)viewDidLoad {
[super viewDidLoad];
TXPreviewParam *param = [[TXPreviewParam alloc] init];
param.videoView = self.view;
param.renderMode = RENDER_MODE_FILL_EDGE;

_editor = [[TXVideoEditer alloc] initWithPreview:param];
_editor.previewDelegate = self;
[_editor setVideoPath:self.videoPath];
[_editor startPlayFromTime:0 toTime:[TXVideoInfoReader getVideoInfo:self.videoP
ath].duration];
}

-(void) onPreviewFinished
{
[_editor startPlayFromTime:0 toTime:[TXVideoInfoReader getVideoInfo:self.videoP
ath].duration];
}
@end
```

At this point, all basic duet features have been implemented. For the demo with more features, please see Source Code of Full-Featured UGSV Application Demo.

Android

Last updated：2020-09-01 14:57:25

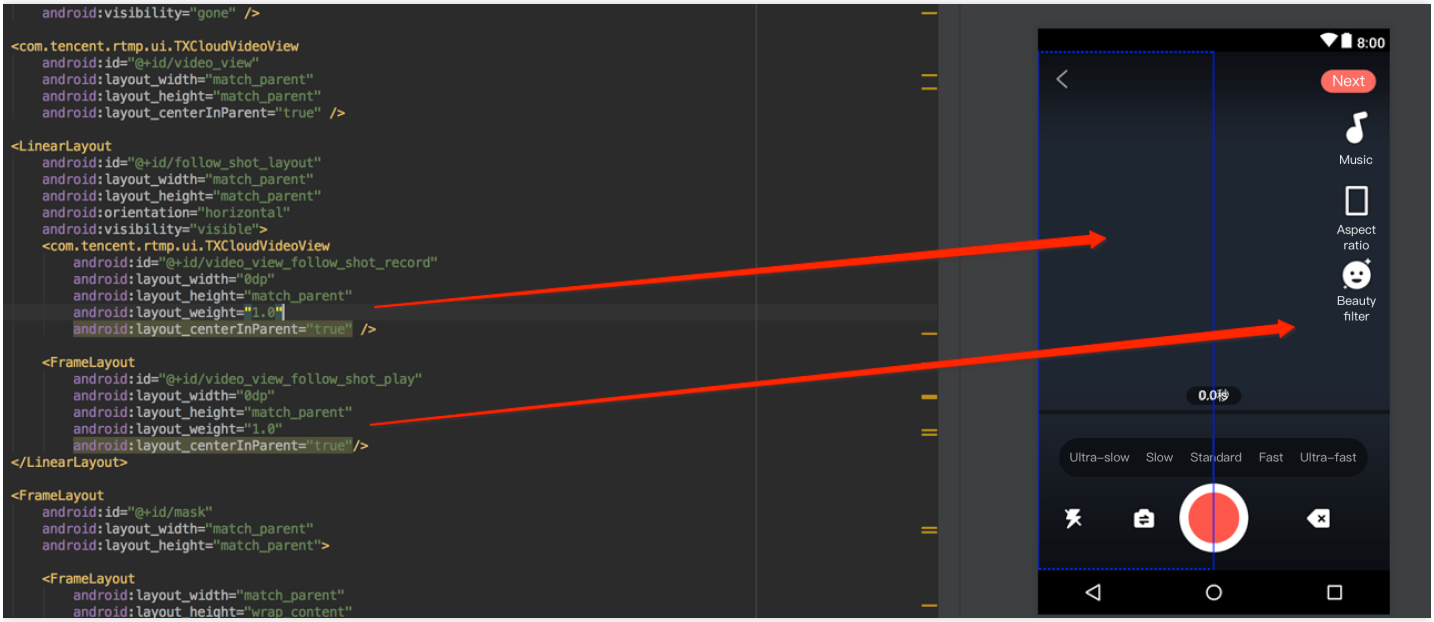
This document describes how to implement basic duet features.

Process Overview

1. Place two views on the page, one for playback, and the other for shoot.
2. Place a button and progress bar for shoot and progress display, respectively.
3. Stop shoot after the video in the same duration as that of the source video has been shot.
4. Compose the shot video with the source video side by side.
5. Preview the composed video.

UI Construction

In `activity_video_record.xml` of the shoot page `TCVideoRecordActivity`, create two views: the left one is the shoot page, and the right one is the playback page.



Sample Code

The duet feature mainly uses three other features: playback, shoot, and composition of the shot and source videos, which correspond to the `TXVideoEditer`, `TXUGCRecord`, and `TXVideoJoiner` SDK classes, respectively. You can also use `TXVodPlayer` for playback.

1. In the video list on the UGSV application homepage, select a video to enter the playback page `TCVodPlayerActivity`. Then, click "Duet" in the bottom-right corner.
   The video will be first downloaded onto the local SD card, the video information such as audio sample rate and frame rate (in fps) will be obtained, and then the shoot page will be displayed.

2. Enter the shoot page `TCVideoRecordActivity` for duet. Please pay attention to the following:
   - The maximum length of the shoot progress bar should be the length of the source video progress bar.
   - The frame rates of the shot video and source video must be the same; otherwise, the audio and video may be out of sync.
   - The audio sample rates of the shot video and source video must be the same; otherwise, the audio and video may be out of sync.
   - The rendering mode set for shoot is fit mode, where the video image can be proportionally scaled at the aspect ratio of 9:16.
   - You need to set audio mute for shoot on Android; otherwise, the source and shot videos' audios will be mixed.

```
// Shoot page
mVideoView = mVideoViewFollowShotRecord;
// Played back video
mFollowShotVideoPath = intent.getStringExtra(TCConstants.VIDEO_EDITER_PATH);
mFollowShotVideoDuration = (int)(intent.getFloatExtra(TCConstants.VIDEO_RECORD_
DURATION, 0) * 1000);
initPlayer();
// The maximum length of the shoot progress bar should be the length of the sou
rce video. The frame rate (`fps`) of the source video should also be used for t
he shot video
mMaxDuration = (int)mFollowShotVideoDuration;
mFollowShotVideoFps = intent.getIntExtra(TCConstants.RECORD_CONFIG_FPS, 20);
mFollowShotAudioSampleRateType = intent.getIntExtra(TCConstants.VIDEO_RECORD_AU
DIO_SAMPLE_RATE_TYPE, TXRecordCommon.AUDIO_SAMPLERATE_48000);
// Initialize the duet API
mTXVideoJoiner = new TXVideoJoiner(this);
mTXVideoJoiner.setVideoJoinerListener(this);
```

```
// Initialize the player. Here, `TXVideoEditer` is used. You can also use `TXVo
dPlayer`
mTXVideoEditer = new TXVideoEditer(this);
mTXVideoEditer.setVideoPath(mFollowShotVideoPath);
TXVideoEditConstants.TXPreviewParam param = new TXVideoEditConstants.TXPreviewP
```

```
aram();
param.videoView = mVideoViewPlay;
param.renderMode = TXVideoEditConstants.PREVIEW_RENDER_MODE_FILL_EDGE;
mTXVideoEditer.initWithPreview(param);
```

```
customConfig.videoFps = mFollowShotVideoFps;
customConfig.audioSampleRate = mFollowShotAudioSampleRateType; // The audio sam
ple rate of the shot video must be the same as that of the source video
customConfig.needEdit = false;
mTXCameraRecord.setVideoRenderMode(TXRecordCommon.VIDEO_RENDER_MODE_ADJUST_RESO
LUTION); // Set the rendering mode to fit mode
mTXCameraRecord.setMute(true); // The duet audio played back by the speaker wil
l not be recorded, because if it is recorded by mic, the source and shot video
s' audios will be mixed
```

3. Start shoot. When the maximum shoot duration is reached, the `onRecordComplete` callback will be returned to proceed with the composition. Here, you need to specify the positions of the two videos in the result.

```
private void prepareToJoiner(){
List<String> videoSourceList = new ArrayList<>();
videoSourceList.add(mRecordVideoPath);
videoSourceList.add(mFollowShotVideoPath);
mTXVideoJoiner.setVideoPathList(videoSourceList);
mFollowShotVideoOutputPath = getCustomVideoOutputPath("Follow_Shot_");
// Proportionally scale the video on the right by the width and height of the s
hot video on the left
int followVideoWidth;
int followVideoHeight;
if ((float) followVideoInfo.width / followVideoInfo.height >= (float)recordVide
oInfo.width / recordVideoInfo.height) {
followVideoWidth = recordVideoInfo.width;
followVideoHeight = (int) ((float)recordVideoInfo.width * followVideoInfo.heigh
t / followVideoInfo.width);
} else {
followVideoWidth = (int) ((float)recordVideoInfo.height * followVideoInfo.width
/ followVideoInfo.height);
followVideoHeight = recordVideoInfo.height;
}

TXVideoEditConstants.TXAbsoluteRect rect1 = new TXVideoEditConstants.TXAbsolute
Rect();
rect1.x = 0; // Top-left point position of the first video
rect1.y = 0;
rect1.width = recordVideoInfo.width; // Width and height of the first video
rect1.height = recordVideoInfo.height;
```

```
TXVideoEditConstants.TXAbsoluteRect rect2 = new TXVideoEditConstants.TXAbsolute
Rect();
rect2.x = rect1.x + rect1.width; // Top-left point position of the second video
rect2.y = (recordVideoInfo.height - followVideoHeight) / 2;
rect2.width = followVideoWidth; // Width and height of the second video
rect2.height = followVideoHeight;

List<TXVideoEditConstants.TXAbsoluteRect> list = new ArrayList<>();
list.add(rect1);
list.add(rect2);
mTXVideoJoiner.setSplitScreenList(list, recordVideoInfo.width + followVideoWidt
h, recordVideoInfo.height); // The second and third parameters: width and heigh
t of the video composition canvas
mTXVideoJoiner.splitJoinVideo(TXVideoEditConstants.VIDEO_COMPRESSED_540P, mFoll
owShotVideoOutputPath);
}
```

4. Listen on the composition callback. After `onJoinComplete`, redirect to the preview page for playback.

```
@Override
public void onJoinComplete(TXVideoEditConstants.TXJoinerResult result) {
mCompleteProgressDialog.dismiss();
if(result.retCode == TXVideoEditConstants.JOIN_RESULT_OK){
runOnUiThread(new Runnable() {
@Override
public void run() {
isReadyJoin = true;
startEditerPreview(mFollowShotVideoOutputPath);
if(mTXVideoEditer != null){
mTXVideoEditer.release();
mTXVideoEditer = null;
}
}
});
}else{
runOnUiThread(new Runnable() {
@Override
public void run() {
Toast.makeText(TCVideoRecordActivity.this, "Composition failed", Toast.LENGTH_S
HORT).show();
}
});
}
}
```

At this point, all basic duet features have been implemented. For the complete code, please see Source Code of Full-Featured UGSV Application Demo.

Image Transition Special Effects
iOS

Last updated：2020-09-01 14:52:09

The image editing feature is added starting from SDK 4.7. You can select a desired image to add effects such as transition animation, background music, and stickers.
The API functions are as follows:

```
/*
*pitureList: list of transition images, which must contain at least three images (note: we recommend you compress the images to 720p or lower (as shown in the demo); otherwise, the memory usage may be too high, causing editing exceptions).
*fps: frame rate of the video generated from the transition images in fps. Value range: 15-30.
* Returned values:
* 0: set successfully
* -1: failed to set. Please check whether the image list exists, the number of images is greater than or equal to 3, and the frame rate (`fps`) is normal
*/
- (int)setPictureList:(NSArray<UIImage *> *)pitureList fps:(int)fps;

/*
*transitionType: transition type. For more information, please see `TXTransitionType`
* Returned values:
* duration: transition video duration (note: the duration for the same image list may vary by transition animation. You can get the transition image duration here)
*/
- (void)setPictureTransition:(TXTransitionType)transitionType duration:(void(^)(CGFloat))duration;
```

- The `setPictureList` API is used to set the image list, which must contain at least three images. If too many images are set, the image size should be appropriate to avoid editing exceptions due to high memory usage.
- The `setPictureTransition` API is used to set the transition effect. Currently, six effects are available, and their durations may vary. You can get the transition duration through `duration` here.
- Pay attention to the API call sequence: call `setPictureList` first and then call `setPictureTransition` .
- Image editing currently does not support loop, reverse, fast/slow motions, and post-roll watermarking, but supports other video editing features. The call method is the same as that of video editing.

Android

Last updated：2020-09-01 14:52:09

The image editing feature is added starting from SDK 4.9. You can select a desired image to add effects such as transition animation, background music, and stickers.
The API functions are as follows:

```
/*
* bitmapList: list of transition images, which must contain at least three images (note: we recommend you compress the images to 720p or lower (as shown in the demo); otherwise, the memory usage may be too high, causing editing exceptions).
* fps: frame rate of the video generated from the transition images in fps. Value range: 15-30.
* Returned values:
* 0: set successfully
* -1: failed to set. Please check whether the image list exists
*/
public int setPictureList(List<Bitmap> bitmapList, int fps);

/*
* type: transition type. For more information, please see `TXVideoEditConstants`
* Returned values:
* duration: transition video duration (note: the duration for the same image list may vary by transition animation. You can get the transition image duration here)
*/
public long setPictureTransition(int type)
```

- Here, the `setPictureList` API is used to set the image list, which must contain at least three images. If too many images are set, the image size should be appropriate to avoid editing exceptions due to high memory usage.
- The `setPictureTransition` API is used to set the transition effect. Currently, six effects are available, and their durations may vary. You can get the transition duration through the returned value here.
- Pay attention to the API call sequence: call `setPictureList` first and then call `setPictureTransition`.
- Image editing currently does not support loop, reverse, and fast/slow motions, but supports other video editing features. The call method is the same as that of video editing.

Customizing Video Data
iOS

Last updated：2022-06-24 14:56:48

Callback for Pre-Processing for Shooting

```
/**
* Call back in the OpenGL thread, where captured images can be processed.
* @param texture Texture ID
* @param width Texture width
* @param height Texture height
* @return Texture returned to the SDK
* Note: The type of textures called back by the SDK is GL_TEXTURE_2D, which must
also be the texture type returned by the API. This callback follows the calling
of beauty filter APIs, and the texture format is GL_RGBA.
*/
- (GLuint)onPreProcessTexture:(GLuint)texture width:(CGFloat)width height:(CGFlo
at)height;

/**
* Call back in the OpenGL thread. You can release OpenGL resources here.
*/
- (void)onTextureDestoryed;
```

Callback for Pre-Processing for Video Editing

```
/**
Call back in the OpenGL thread, where captured images can be processed.
@param textureId Texture ID
@param width Texture width
@param height Texture height
@param timestamp Texture timestamp (ms)
@return Texture returned to the SDK
Note: The type of textures called back by the SDK is GL_TEXTURE_2D, which must a
lso be the texture type returned by the API. This callback follows the calling o
f beauty filter APIs, and the texture format is GL_RGBA.
Timestamp is the PTS of the current video frame and is measured in milliseconds.
You can customize beauty filter effects based on your needs.
```

```
*/
- (GLuint)onPreProcessTexture:(GLuint)texture width:(CGFloat)width height:(CGFloat)height timestamp:(UInt64)timestamp;

/**
* Call back in the OpenGL thread. You can release OpenGL resources here.
*/
- (void)onTextureDestoryed;
```

Android

Last updated：2020-09-01 14:52:09

Shoot Preprocessing Callback

```
public interface VideoCustomProcessListener {
/**
* Call back in the OpenGL thread. You can further process the captured video imag
e in the callback
* @param textureId Texture ID
* @param width Texture width
* @param height Texture height
* @return Texture ID returned to SDK. If no processing is required, simply return
the texture ID passed in.
* Note: the texture type called back by the SDK is `GLES20.GL_TEXTURE_2D`, which
must also be the texture type returned to the SDK by the API
*/
int onTextureCustomProcess(int textureId, int width, int height);

/**
* Call back face coordinates in Enterprise Edition
* @param points Normalized face coordinates. Every two coordinates indicate the `
X` and `Y` values of a point. The value range is [0.f,1.f]
*/
void onDetectFacePoints(float[] points);

/**
* Call back in the OpenGL thread. You can release the created OpenGL resources in
the callback
*/
void onTextureDestroyed();
}
```

Editing Preprocessing Callback

```
public interface TXVideoCustomProcessListener {
/**
* Call back in the OpenGL thread. You can further process the captured video imag
e in the callback
*
```

```
* @param textureId Texture ID
* @param width Texture width
* @param height Texture height
* @return Texture ID returned to SDK. If no processing is required, simply return
the texture ID passed in.
* <p>
* Note: the texture type called back by the SDK is `GLES20.GL_TEXTURE_2D`, which
must also be the texture type returned to the SDK by the API
*/
int onTextureCustomProcess(int textureId, int width, int height, long timestamp);

/**
* Call back in the OpenGL thread. You can release the created OpenGL resources in
the callback
*/
void onTextureDestroyed();
}
```

Video Porn Detection

Last updated：2020-09-01 14:56:48

In scenarios such as personal live streaming shoot and UGSV, the video content is unpredictable. To prevent non-compliant contents from being displayed on the VOD platform, you need to audit the uploaded videos first and transcode and distribute them after confirming that they are compliant. The Tencent Cloud UGSV solution supports AI-based porn detection in videos, which can automatically identify whether a video involves pornographic information.

Using AI-based Porn Detection

The AI-based porn detection feature can be used only after it is integrated into the video processing task flow. It depends on the video AI - content audit feature on the VOD backend, and the audit result will be sent to the application backend service as an event notification. VOD has a built-in task flow `QCVB_ProcessUGCFile` for UGSV porn detection scenarios. If you use this task flow and specify to perform AI-based porn detection, the porn detection task will be executed first, and whether to perform subsequent operations (such as transcoding, watermarking, and screencapturing) will be determined based on the porn detection result.

AI Template Overview

Template ID	Porn Processing	Sample Rate
10	End the task flow (operations such as transcoding, watermarking, and screencapturing will not be executed subsequently)	For videos whose duration is less than 500 seconds, sampling is performed once per second; for videos whose duration is greater than or equal to 500 seconds, sampling is performed once every 1% of the duration
20	Continue executing the task flow	None

AI-based Porn Detection Connection Sample

Step 1. Submit an AI-based porn detection task when generating an upload signature

```
/**
* Generate a signature that contains an AI-based porn detection task
*/
```

```
function getUploadSignature(req, res) {
res.json({
code: 0,
message: 'ok',
data: {
// Specify the template parameters and task flow during the upload
signature: gVodHelper.createFileUploadSignature({ procedure: 'QCVB_SimpleProcessF
ile({1,1,1,1})' })
}
});
}
```

Step 2. Get the AI-based porn detection result when getting the event notification

```
/**
* Get the AI-based porn detection result of the event
*/
function getAiReviewResult(event){
let data = event.eventContent.data;
if(data.aiReview){
return data.aiReview;
}
return {};
}
```

Custom Themes
iOS

Last updated：2022-11-15 11:20:02

`UGCKit` allows you to freely modify the preset text, colors, and icons.

Text

`UGCKit` offers two language packages by default: Simplified Chinese and American English. All their localized strings are stored in the default standard localized string format in `UGCKit/UGCKitResources/Localizable.strings`. You can replace the text to change the default strings or add new languages.

Below is an example of changing the Simplified Chinese animated effect names, which are in `UGCKit/UGCKitResources/zh-Hans.lproj/Localizable.strings`.

```
"UGCKit.Edit.VideoEffect.DynamicLightWave" = "Rock light";
"UGCKit.Edit.VideoEffect.DarkFantasy" = "Dark dream";
"UGCKit.Edit.VideoEffect.SoulOut" = "Soul out";
"UGCKit.Edit.VideoEffect.ScreenSplit" = "Split screen";
"UGCKit.Edit.VideoEffect.Shutter" = "Blinds";
"UGCKit.Edit.VideoEffect.GhostShadow" = "Shadow";
"UGCKit.Edit.VideoEffect.Phantom" = "Phantom";
"UGCKit.Edit.VideoEffect.Ghost" = "Ghost";
"UGCKit.Edit.VideoEffect.Lightning" = "Lightning";
"UGCKit.Edit.VideoEffect.Mirror" = "Mirror";
"UGCKit.Edit.VideoEffect.Illusion" = "Illusion";
```

To change "Illusion" to "Phantasm", you only need to modify the last line as follows:

```
"UGCKit.Edit.VideoEffect.Illusion" = "Phantasm";
```

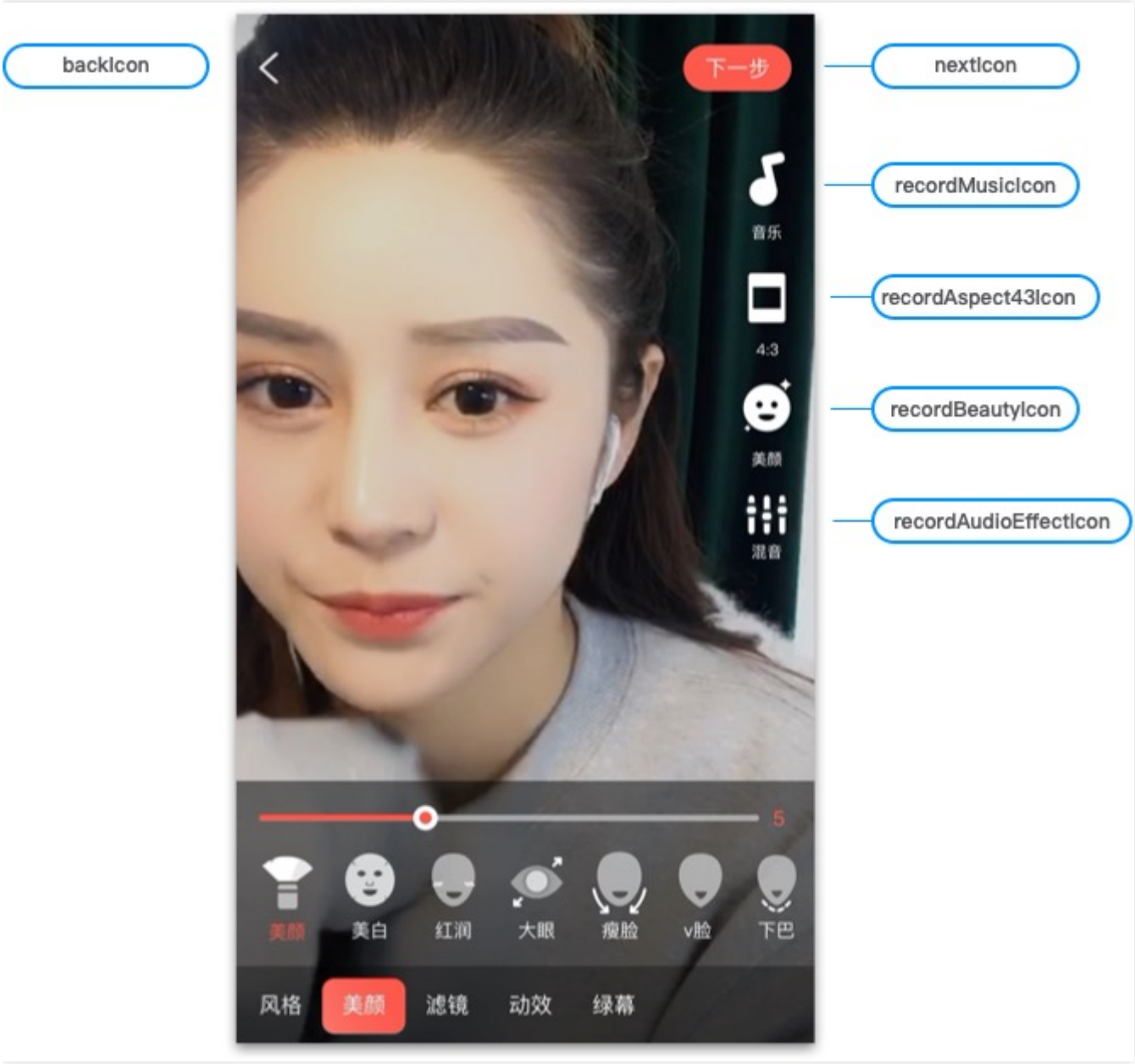
Color

The methods of getting the colors on all UIs of `UGCKit` are defined in the `UGCKitTheme` class. You can modify the attribute values to change colors. For the specific resource names, see the comments in `UGCKitTheme.h`.
Below is an example of changing the background color of the application UI:

```
UGCKitTheme *theme = [[UGCKitTheme alloc] init];
theme.backgroundColor = [UIColor whiteColor]; // Change the background color to w
hite
UGCKitEditViewController *editViewController = [[UKEditViewController alloc] init
WithMedia:media config:nil theme:theme]; // Use the custom theme to create a cont
roller
```

Icons

The icons on all UIs of `UGCKit` are in the `UGCKit.xcassets` resource file and can be replaced as needed. The specific icon filenames can be viewed in `UGCKitTheme.h` . An icon's resource name is the same as its attribute/method name, and all icons are defined in `UGCKitTheme.h` . Taking the recording UI as an example, the following names are the icons' attribute names in `UGCKitTheme` as well as the resource names in `UGCKit.xcassets` .



You can replace icons in `UGCKit` in two ways:

- Directly replace icons in `UGCKitTheme.xcassets` .
- Write code to assign values to objects in `UGCKitTheme` to change icons.

Below is the sample code for changing icons on the recording UI:

```
UGCKitTheme *theme = [[UGCKitTheme alloc] init];
theme.nextIcon = [UIImage imageName:@"myConfirmIcon"]; // Set the icon used to co
mplete recording
theme.recordMusicIcon = [UIImage imageName:@"myMusicIcon"]; // Set the icon of th
e music feature
theme.beautyPanelWhitnessIcon = [UIImage imageNamed:@"beauty_whitness"]; // Set t
```

```
he icon of the brightening filter
UGCKitRecordViewController *viewController = [[UGCKitRecordViewController alloc]
initWithConfig:nil theme:theme]; // Use the custom theme to create a controller
```

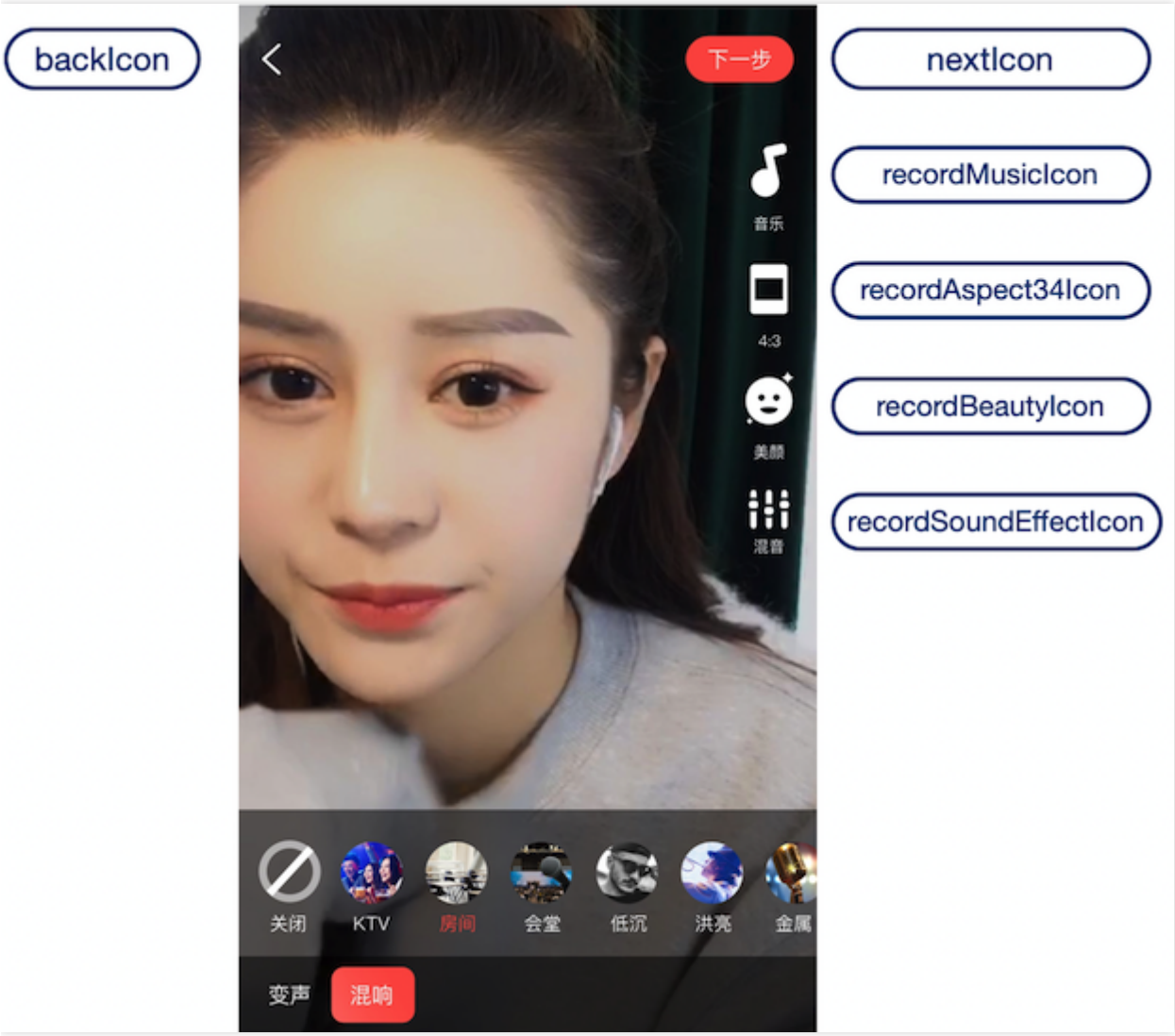
Android

Last updated：2022-11-14 18:25:34

`UGCKit` is a set of encapsulated interactive UIs, which is the UGSV demo app's theme by default. With simple modifications, you can customize your own theme and replace the icons, text, and colors.

Customizing the recording theme

The recording UI contains the right icon toolbar, the bottom toolbar, music panel, beauty filter panel, and sound effect panel.



1. Declare a `<style>` in `app/res/values/style.xml`, specify its parent theme as `RecordStyle`, and change the theme to the target theme. You can find all available themes in

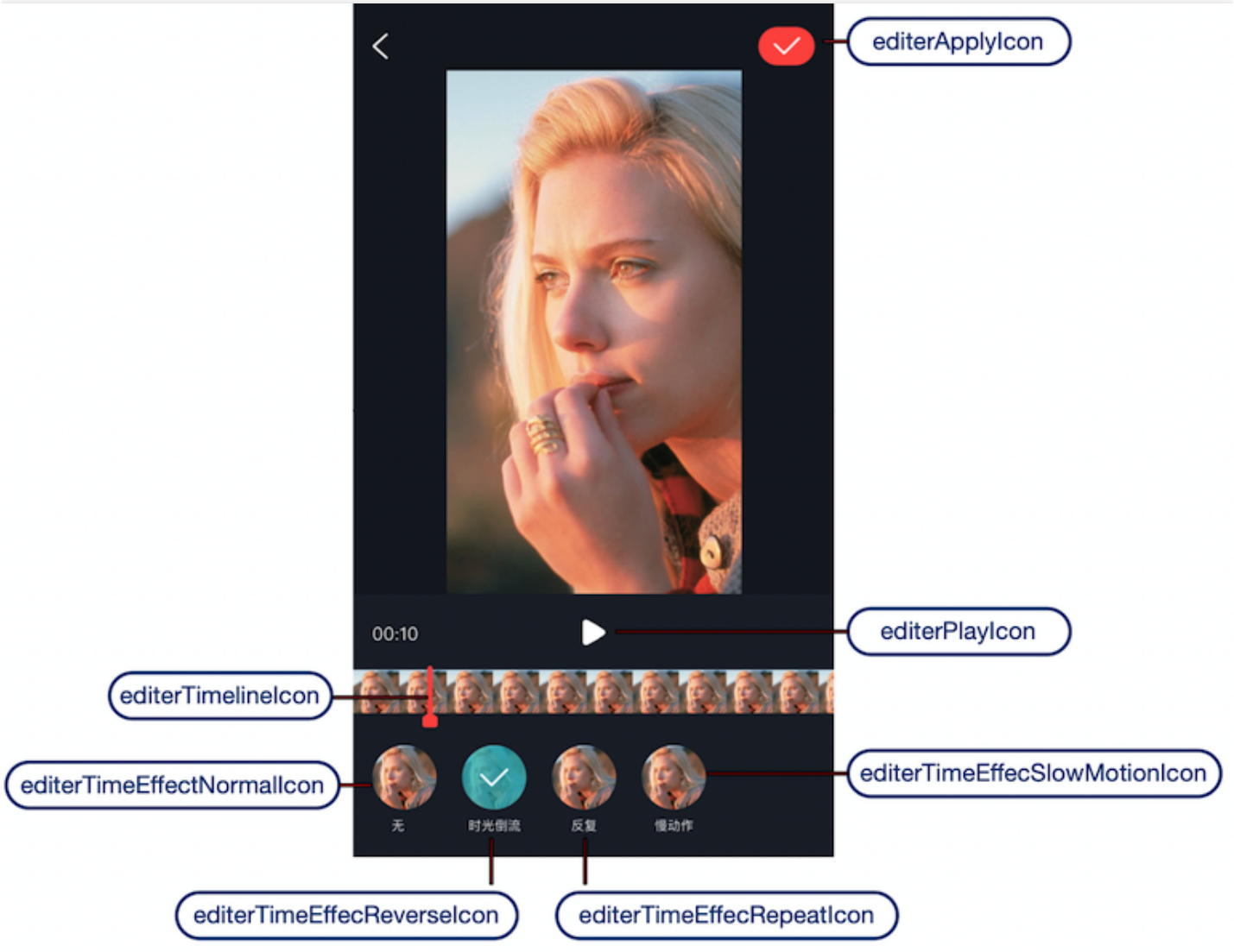
`app/ugckit/res/values.theme_style.xml` .

Below is the sample code for replacing the music and beauty filter icons on the recording UI:

2. Declare the custom theme in `AndroidManifest.xml` :

Customizing the editing theme

The editing UI contains the editing and clipping UI, animated effect, beauty filter, and special effect panel, and speed, filter, sticker, and bubble subtitles panels.



1. Declare a `<style>` in `app/res/values/style.xml` , specify its parent theme as `EditerStyle` , and change the theme to the target theme. You can find all available themes in `app/ugckit/res/values.theme_style.xml` .

Below is the sample code for replacing the playback and pause icons on the editing UI:

2. Declare the custom theme in `AndroidManifest.xml` :