# SwitchButton组件功能介绍

## 功能介绍：

SwitchButton组件是一个可以切换开关状态的组件，可通过点击和滑动实现开关。

## 模拟器上运行效果：

关：

开：

# SwitchButton使用方法

## 新建工程，增加组件Har包依赖

在应用模块中添加HAR，先将switchbutton-debug.har复制到entry\libs目录下，再在build.gradle中添加依赖。

dependencies **{** implementation fileTree(dir: 'libs', include: ['\*.jar', '\*.har'])  
 compile project(path: ':switchbutton')  
**}**

## 修改主页面的布局文件

修改主页面的布局文件ability\_main.xml，增加组件**com.isoftstone.switchbutton.SwitchButton**

修改后代码如下：

其中app开头的为自定义属性，xml文件中必须定义，否则控件无法正常运行

<?xml version="1.0" encoding="utf-8"?>  
  
 <DirectionalLayout  
 xmlns:ohos="http://schemas.huawei.com/res/ohos"  
 xmlns:app="http://schemas.huawei.com/apk/res/ohos"  
 ohos:height="match\_parent"  
 ohos:width="match\_parent"  
 ohos:left\_margin="20vp"  
 ohos:layout\_alignment="horizontal\_center"  
 ohos:orientation="vertical">  
  
 <com.isoftstone.switchbutton.SwitchButton  
 ohos:id="$+id:switchBtn1"  
 ohos:top\_margin="100vp"  
 ohos:height="match\_parent"  
 ohos:width="match\_parent"  
 app:SwitchButton\_ThumbDrawable\_on = "$graphic:button\_thumb\_on"  
 app:SwitchButton\_ThumbDrawable\_off = "$graphic:button\_thumb\_off"  
 app:SwitchButton\_BackDrawable\_on = "$graphic:button\_back\_on"  
 app:SwitchButton\_BackDrawable\_off = "$graphic:button\_back\_off"  
 app:SwitchButton\_BackHeight = "200"  
 app:SwitchButton\_BackWidth = "300"  
 app:SwitchButton\_kswThumbMargin = "100"  
 app:SwitchButton\_kswThumbWidth = "100"  
 app:SwitchButton\_kswThumbHeight = "90"  
 app:SwitchButton\_kswThumbRadius = "50"  
 app:SwitchButton\_kswBackRadius = "50"  
 app:SwitchButton\_kswThumbRangeRatio = "2.2"  
 app:SwitchButton\_tintColor = "$color:thumb\_on"  
 app:SwitchButton\_kswTextOn = "ON"  
 app:SwitchButton\_kswTextOff = "OFF"  
 ohos:layout\_alignment="horizontal\_center"/>  
  
 </DirectionalLayout>

## 添加视图资源



需要定义4个状态，分别为轨迹的开关状态，滑块的开关状态，文件名参考2.2中定义

<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:ohos="http://schemas.huawei.com/res/ohos"  
 ohos:shape="oval">  
 <solid  
 ohos:color="#FFFFFF"/>  
</shape>

## 修改MainAbilitySlince的UI加载代码

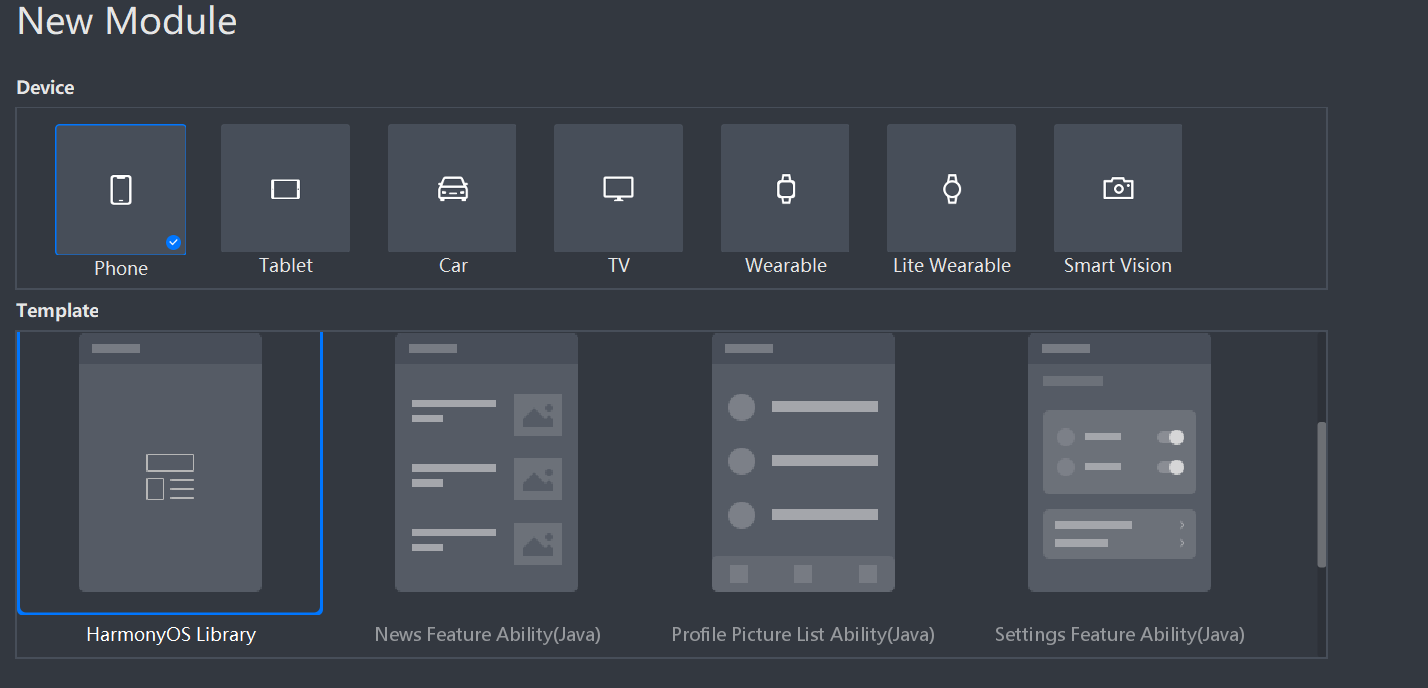
在MainAbilitySlince类的onStart函数中，增加如下代码。

public void onStart(Intent intent) {  
 super.onStart(intent);  
 super.setUIContent(ResourceTable.*Layout\_ability\_switchButton*);  
 SwitchButton button = (SwitchButton) findComponentById(ResourceTable.*Id\_switchBtn1*);  
 }

# SwitchButton开发实现

## 新建一个Module

新建一个Module，类型选择HarmonyOS Library，模块名为loadingview，如图



## 新建一个SwitchButton类

新建一个SwitchButton类，继承自AbsButton类，实现AbsButton.CheckedStateChangedListener和Component.DrawTask

### 重写onDraw

代码如下：

public void onDraw(Component component, Canvas canvas){  
 float progress;  
 HiLog.*info*(*label*,"onDraw：isCheck"+isCheck);  
 HiLog.*info*(*label*,"onDraw：isOnToOff"+isOnToOff);  
 HiLog.*info*(*label*,"onDraw：mProgress："+mProgress);  
 if(isOnToOff){  
 progress = mProgress;  
 }else{  
 progress = mProgress;  
 }  
 HiLog.*info*(*label*,"onDraw：progress："+progress);  
 HiLog.*info*(*label*,"onDraw");  
 if (!mReady) {  
 HiLog.*info*(*label*,"onDraw:setup");  
 setup();  
 }  
 if (!mReady) {  
 HiLog.*info*(*label*,"onDraw:return");  
 return;  
 }  
  
 // fade back  
 if (mIsBackUseDrawable) {  
 if (mFadeBack && mCurrentBackDrawable != null && mNextBackDrawable != null) {  
 // fix #75, 70%A + 30%B != 30%B + 70%A, order matters when mix two layer of different alpha.  
 // So make sure the order of on/off layers never change during slide from one endpoint to another.  
 Element below = isChecked() ? mCurrentBackDrawable : mNextBackDrawable;  
 Element above = isChecked() ? mNextBackDrawable : mCurrentBackDrawable;  
  
 int alpha = (int) (255 \* getProgress());  
 below.setAlpha(alpha);  
 below.drawToCanvas(canvas);  
 alpha = 255 - alpha;  
 above.setAlpha(alpha);  
 above.drawToCanvas(canvas);  
 } else {  
 HiLog.*info*(*label*,"onDraw:619");  
 mBackDrawable.setAlpha(255);  
 mBackDrawable.drawToCanvas(canvas);  
 }  
 } else {  
 if (mFadeBack) {  
 HiLog.*info*(*label*,"onDraw:624");  
 int alpha;  
 int colorAlpha;  
  
 // fix #75  
 int belowColor = isChecked() ? mCurrBackColor : mNextBackColor;  
 int aboveColor = isChecked() ? mNextBackColor : mCurrBackColor;  
 HiLog.*info*(*label*,"onDraw:mCurrBackColor:"+mCurrBackColor);  
 HiLog.*info*(*label*,"onDraw:mNextBackColor:"+mNextBackColor);  
 HiLog.*info*(*label*,"onDraw:belowColor:"+belowColor);  
 HiLog.*info*(*label*,"onDraw:aboveColor:"+aboveColor);  
 // 当前背景色  
 alpha = (int) (255 \* getProgress());  
 colorAlpha = Color.*alpha*(belowColor);  
 colorAlpha = colorAlpha \* alpha / 255;  
 RgbColor rgbColor = new RgbColor(belowColor);  
// mPaint.setARGB(colorAlpha, Color.red(belowColor), Color.green(belowColor), Color.blue(belowColor));  
 Color color = new Color(Color.*argb*(colorAlpha, rgbColor.getRed(), rgbColor.getGreen(), rgbColor.getBlue()));  
// Color color = new Color(belowColor);  
// HiLog.info(label,"onDraw:color:"+color.getValue());  
 mPaint.setColor(color);  
 HiLog.*info*(*label*,"onDraw:color:"+color);  
 canvas.drawRoundRect(mBackRectF, mBackRadius, mBackRadius, mPaint);  
  
 // next back  
 alpha = 255 - alpha;  
 colorAlpha = Color.*alpha*(aboveColor);  
 colorAlpha = colorAlpha \* alpha / 255;  
// mPaint.setARGB(colorAlpha, Color.red(aboveColor), Color.green(aboveColor), Color.blue(aboveColor));  
 RgbColor rgbColor2 = new RgbColor(aboveColor);  
 Color color2 = new Color(Color.*argb*(colorAlpha, rgbColor2.getRed(), rgbColor2.getGreen(), rgbColor2.getBlue()));  
// Color color2 = new Color(aboveColor);  
 mPaint.setColor(color2);  
 HiLog.*info*(*label*,"onDraw:color2:"+color2);  
 canvas.drawRoundRect(mBackRectF, mBackRadius, mBackRadius, mPaint);  
  
 mPaint.setAlpha(255);  
 } else {  
 HiLog.*info*(*label*,"onDraw:653");  
 mPaint.setColor(new Color(mCurrBackColor));  
 canvas.drawRoundRect(mBackRectF, mBackRadius, mBackRadius, mPaint);  
 }  
 }  
  
 //text  
 Layout switchText =progress > 0.5 ? mOnLayout : mOffLayout;  
 HiLog.*info*(*label*,"onDraw:switchText:"+switchText.getHeight()+",width:"+switchText.getWidth());  
 RectFloat textRectF = progress > 0.5 ? mTextOnRectF : mTextOffRectF;  
 if (switchText != null && textRectF != null) {  
 int alpha = (int) (255 \* (progress >= 0.75 ? progress \* 4 - 3 : progress < 0.25 ? 1 - progress \* 4 : 0));  
 int textColor = progress > 0.5 ? mOnTextColor : mOffTextColor;  
 int colorAlpha = Color.*alpha*(textColor);  
 colorAlpha = colorAlpha \* alpha / 255;  
  
 RgbColor textColorRgb = new RgbColor(textColor);  
 Color color2 = new Color(Color.*argb*(colorAlpha, textColorRgb.getRed(), textColorRgb.getGreen(), textColorRgb.getBlue()));  
 mTextPaint.setColor(color2);  
  
 canvas.save();  
 canvas.translate(textRectF.left, textRectF.top);  
 switchText.drawText(canvas);  
 HiLog.*info*(*label*,switchText.toString());  
 canvas.restore();  
 }  
  
 // thumb滑块  
 mPresentThumbRectF.modify(mThumbRectF);  
 HiLog.*info*(*label*,"onDraw:设置mPresentThumbRectF:");  
 setOffsetOfRectF(mPresentThumbRectF,progress \* mSafeRectF.getWidth(), 0);  
 HiLog.*info*(*label*,"onDraw:设置mPresentThumbRectF成功:");  
 if (mIsThumbUseDrawable) {  
 HiLog.*info*(*label*,"onDraw:mThumbDrawable:");  
 mThumbDrawable.setBounds((int) mPresentThumbRectF.left, (int) mPresentThumbRectF.top, ceil(mPresentThumbRectF.right), ceil(mPresentThumbRectF.bottom));  
 mThumbDrawable.drawToCanvas(canvas);  
 } else {  
 mPaint.setColor(new Color(mCurrThumbColor));  
 canvas.drawRoundRect(mPresentThumbRectF, mThumbRadius, mThumbRadius, mPaint);  
 HiLog.*info*(*label*,"onDraw:drawRoundRect:");  
 }  
 }

### 重写OnTouchEvent

public boolean onTouchEvent(Component component, TouchEvent touchEvent) {  
 {  
 int action = touchEvent.getAction();  
 MmiPoint point = touchEvent.getPointerPosition(touchEvent.getIndex());  
 float deltaX = point.getX() - mStartX;  
 float deltaY = point.getY() - mStartY;  
  
 switch (action) {  
 case TouchEvent.*PRIMARY\_POINT\_DOWN*:  
 mStartX = point.getX();  
 mStartY = point.getY();  
 mLastX = mStartX;  
 setPressState(true);  
 break;  
  
 case TouchEvent.*POINT\_MOVE*:  
 float x = point.getX();  
 drawableStateChanged(isCheck?ComponentState.*COMPONENT\_STATE\_CHECKED*:ComponentState.*COMPONENT\_STATE\_EMPTY*);  
 setProgress(getProgress() + (x - mLastX) / mSafeRectF.getWidth());  
 mLastX = x;  
 if (!mCatch && (Math.*abs*(deltaX) > mTouchSlop / 2f || Math.*abs*(deltaY) > mTouchSlop / 2f)) {  
 if (deltaY == 0 || Math.*abs*(deltaX) > Math.*abs*(deltaY)) {  
 catchView();  
 } else if (Math.*abs*(deltaY) > Math.*abs*(deltaX)) {  
 return false;  
 }  
 }  
 break;  
  
 case TouchEvent.*CANCEL*:  
 case TouchEvent.*PRIMARY\_POINT\_UP*:  
 mCatch = false;  
 float time = touchEvent.getOccurredTime() - touchEvent.getStartTime();  
 if (Math.*abs*(deltaX) < mTouchSlop && Math.*abs*(deltaY) < mTouchSlop && time < mClickTimeout) {  
 HiLog.*info*(*label*,"Log\_onTouchEvent simulateClick");  
 simulateClick();  
 } else {  
 boolean nextStatus = getStatusBasedOnPos();  
 if (nextStatus != isChecked()) {  
 setChecked(nextStatus);  
 }  
 else {  
 setProgress(nextStatus?1:0);  
 }  
 }  
 if (isPressed()) {  
 if (mUnsetPressedState == null) {  
 mUnsetPressedState = new UnsetPressedState();  
 }  
 }  
 break;  
  
 default:  
 break;  
 }  
 return true;  
 }  
}

### 动画实现（有待进一步解决）

为了实现动画，需要定义一个AnimatorValue，并设置动画侦听回调函数，代码如下：

由于动画无法设置数值变化，该动画和滑动效果的组件重绘回发生冲突（滑动效果会设置progress值，根据progress值重绘控件，当绘制进行一半时，滑动结束会修改button状态，开启动画连续实现后半程的绘制，但目前动画默认从0-1，不支持中间值设置，button会还原重新绘制），暂时关闭，有待解决

// 动画  
private AnimatorValue animatorValue;

mProgressAnimator = new AnimatorValue();

mProgressAnimator.setDuration(DEFAULT\_ANIMATION\_DURATION);

mProgressAnimator.setCurveType(Animator.CurveType.LINEAR);

// mProgressAnimator.setInterpolator(new AccelerateDecelerateInterpolator());

mProgressAnimator.setValueUpdateListener(new AnimatorValue.ValueUpdateListener(){

@Override

public void onUpdate(AnimatorValue animatorValue, float v) {

setProgress(v);

HiLog.info(label, "mProgressAnimator onUpdate:"+v);

}

});  
  
 // 启动动画

protected void animateToState(boolean checked) {

if (mProgressAnimator == null) {

return;

}

if (mProgressAnimator.isRunning()) {

mProgressAnimator.cancel();

}

mProgressAnimator.setDuration(mAnimationDuration);

mProgressAnimator.start();

}  
}

## 编译HAR包

利用Gradle可以将HarmonyOS Library库模块构建为HAR包，构建HAR包的方法如下：

在Gradle构建任务中，双击PackageDebugHar或PackageReleaseHar任务，构建Debug类型或Release类型的HAR。

待构建任务完成后，可以在工程目录中的switchbutton> bulid > outputs > har目录中，获取生成的HAR包。

