# **鸿蒙开源第三方组件——进度轮ProgressWheel**

## **前言**

基于安卓平台的进度轮组件ProgressWheel（https://github.com/Alford087/ProgressWheel），实现了鸿蒙化迁移和重构，代码已经开源到（https://gitee.com/isrc\_ohos/progress-wheel\_ohos），欢迎各位下载使用并提出宝贵意见！

## 背景

 进度轮是UI界面中常见的组件，通常用于向用户显示某个耗时操作完成的百分比，例如：加载状态、下载进度、刷新网页等。进度轮可以动态的显示操作进度，避免用户误以为程序失去响应，从而更好地提高用户界面的友好性。

## 组件功能展示

基于鸿蒙系统，通过自定义控件属性的方式实现了进度轮组件，该组件支持进度轮的旋转、进度增加两种功能。

### 旋转

点击“Start spinning”按钮，此时进度轮会开始旋转，在旋转过程中按钮上的“Start spinning”变成“Stop spinning”，点击“Stop spinning”用户可以随时停止旋转，效果如图1所示。进度轮旋转功能主要用于展示服务器正在加载数据的状态，此时的作用和加载动画库AVLoadingIndicatorView类似。



图1 进度轮旋转

### 进度增加

点击“Increment”按钮，进度轮会定量增加进度，进度值会实时显示在进度轮的中间，效果如图2所示，进度增加功能主要用于展示服务器加载数据的进度。



图2 按钮控制进度增加

## Sample解析

在Sample中向用户提供了5个场景，分别是：（1）进度轮旋转、（2）按钮控制进度增加、（3）原生进度条控制进度增加、（4）背景改变、（5）样式改变。其中（1）、（2）两种场景较为简单，均为按钮触发，调用ProgressWheel类的开始旋转、进度增加方法即可，在Library解析部分会详解解释。此处重点介绍（3）、（4）、（5）三种场景。

### 原生进度条控制进度增加



图3 原生进度条控制进度增加

原生进度条是指鸿蒙系统的基本组件slider，它也可以用于显示内容加载或操作处理的进度，此处我们通过拖动原生进度条来改变进度轮的进度值，并将进度值实时显示。效果如图3所示，代码实现如下：

@Override  
**public void** onProgressUpdated(Slider seekBar, **int** i, **boolean** b){

*//原生进度条和进度轮换算，100代表原生进度条的进度最大值，360代表进度轮的进度最大值*  
 **double** progress = 360.0 \* (seekBar.getProgress() / 100.0);

*//进度轮进度设置*

wheel.setProgress((int) progress);  
}

### 背景改变



图4 进度轮背景改变

使用Random 类产生随机数，特定处理后作为背景像素点。点击“Random bg”按钮，背景像素点显示，进度轮的背景会发生随机变化。效果如图4所示。代码如下：

*//背景改变*

**private static void** randomBg(ProgressWheel wheel) {

*//随机产生背景元素*

Random random = **new** Random();  
 **int** firstColour = random.nextInt();*//随机数获取*

**int** secondColour = random.nextInt();  
 **int** patternSize = (1 + random.nextInt(3)) \* 8;*//随机数处理*  
  **int** patternChange = (1 + random.nextInt(3)) \* 8;  
 **int**[] pixels = new int[patternSize];  
 **for** (**int** i = 0; i < patternSize; i++) {  
 pixels[i] = (i > patternChange) ? firstColour : secondColour;*//得到像素点*  
 }  
 PixelMap.InitializationOptions options=new PixelMap.InitializationOptions();  
 options.size=**new** Size(1,patternSize);  
 options.pixelFormat=PixelFormat.ARGB\_8888;

*//设置背景元素*  
 wheel.setRimShader(**new** PixelMapShader(  
 **new** PixelMapHolder(PixelMap.create(pixels, options)),  
 Shader.TileMode.REPEAT\_TILEMODE,  
 Shader.TileMode.REPEAT\_TILEMODE), Paint.ShaderType.RADIAL\_SHADER);  
}

### 样式改变



图5 进度轮样式改变

通过自定义进度轮的长度、宽度、背景等来设计不同的样式，点击“A different style”按钮触发样式改变，效果如图5所示，代码如下：

*//样式改变*

**private static void** styleRandom(ProgressWheel wheel, Context ctx) {  
 wheel.setRimShader(**null**, Paint.ShaderType.RADIAL\_SHADER);  
 wheel.setRimColor(0xFFFFFFFF);  
 wheel.setCircleColor(0x00000000);*//内圆颜色*  
 wheel.setBarColor(0xFF000000);*//进度轮体颜色*  
 wheel.setContourColor(0xFFFFFFFF);*//外圆颜色*  
 wheel.setBarWidth(pxFromDp(ctx, 8));*//宽度*

wheel.setBarLength(pxFromDp(ctx, 100));*//长度*

wheel.setSpinSpeed(2);*//旋转速度*

wheel.setDelayMillis(3);*//间隔时间*  
}

## Library解析

### 1.功能实现

1.声明setupBounds()、setupPaints()方法，用canvas绘制进度轮，设定内圆、外圆、条纹等、文字等属性。文字用于显示进度轮的属性值，不局限于显示当前进度。

**public** ProgressWheel(Context context) {  
 **super**(context);DrawTask task = (component, canvas) -> {  
 *//初始化元素边界* setupBounds();  
 *//初始化绘制属性* setupPaints();  
 *//绘制内圆* canvas.drawArc(**innerCircleBounds**, **new** Arc(360, 360, **false**), **circlePaint**);  
 *//绘制外圆* canvas.drawArc(**circleBounds**, **new** Arc(360, 360, **false**), **rimPaint**);canvas.drawArc(**circleOuterContour**, **new** Arc(360, 360, **false**), **contourPaint**);  
 *//绘制条纹* **if** (**isSpinning**) {  
 canvas.drawArc(**circleBounds**, **new** Arc(**progress** - 90, **barLength**, **false**), **barPaint**);  
 } **else** {  
 canvas.drawArc(**circleBounds**, **new** Arc(-90, **progress**, **false**), **barPaint**);  
 }  
  *//设置文字于圆心处显示*

**float** textHeight = **textPaint**.descent() - **textPaint**.ascent();  
 **float** verticalTextOffset = (textHeight / 2) - **textPaint**.descent();  
 **for** (String line : **splitText**) {  
 **float** horizontalTextOffset = **textPaint**.measureText(line) / 2;  
 canvas.drawText(  
 **textPaint**,  
 line,  
 (**float**) component.getWidth() / 2 - horizontalTextOffset,  
 (**float**) component.getHeight() / 2 + verticalTextOffset);  
 }

*//旋转时在不同的位置画进度条*

**if** (**isSpinning**) {  
 scheduleRedraw();  
 }  
 };  
 addDrawTask(task);  
}

2.进度轮旋转

该功能只提供给用户进度轮旋转的展示形式，不提供当前线程的量化进度。

1）进度轮开始旋转方法。

**public void** startSpinning() {  
 **isSpinning** = **true**;*//设置当前为旋转状态*

**pinHandler**.sendEvent(0);*//每隔一定时间重新画进度,来达到旋转的效果*  
}

2）进度轮停止旋转方法。

**public void** stopSpinning() {  
 **isSpinning** = **false**;*//设置当前为停止状态*  
 **progress** = 0;*//进度清零*

invalidate();

}

1. 进度增加

该模式在旋转时提供当前的量化进度数据，用户可以清晰地了解当前的线程进度，是一种对用户更友好的交互模式。

**public void** incrementProgress(**int** amount) {

**isSpinning** = **false**;*//增加进度时进度轮不旋转*  
 **progress**+= amount;*//定量增加*  
 **if** (**progress** > 360){  
 **progress** %= 360;*//超过360会自动重置*  
 }  
 invalidate();  
}

### 2.移植方法

本组件在移植时大部分采用API替换的方法，少数方法需要重写，如处理进度轮旋转的时候重写spinHandler()方法，该方法的功能是：进度轮旋转时在不同的像素位置绘制进度条，移动的位置超过360度则置为0度，重新旋转。代码如下：

*//每次绘制要移动的像素数目*

**private float spinSpeed** = 2f; *//绘制过程的时间间隔*

**private int delayMillis** = 100;

**private** EventHandler **spinHandler** = **new** EventHandler(EventRunner.getMainEventRunner())  
{@Override  
 **public void** processEvent(InnerEvent msg)  
 {  
 invalidate();  
 **if** (**isSpinning**)  
 {

*//更新画进度的位置*

**progress** += **spinSpeed**;

*//要移动的像素数目超过360则重置* **if** (**progress** > 360)  
 {  
 **progress** = 0;  
 }  
 **spinHandler**.sendEvent(0, **delayMillis**);  
 }  
 **super**.processEvent(msg);  
 }  
};

## 项目贡献人

刘磊 郑森文 朱伟 陈美汝 张馨心