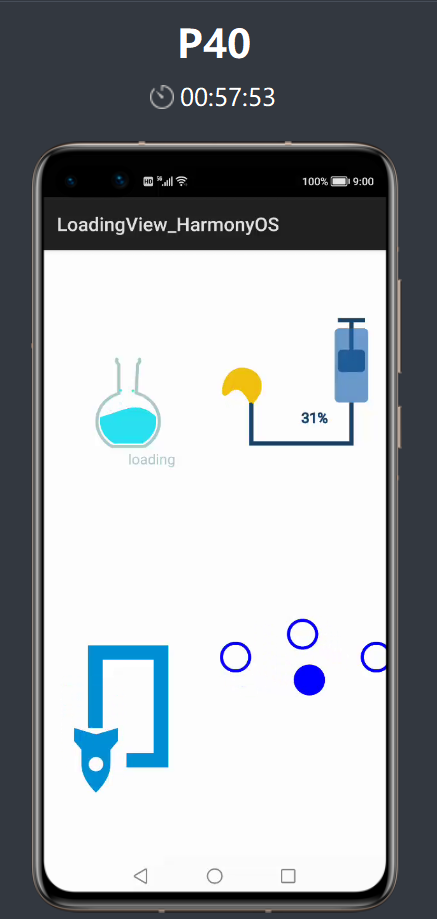
# LoadingView组件功能介绍

## 功能介绍：

LoadingView组件是一个可以显示加载动画的三方组件，目前支持4种风格的显示。

## 模拟器上运行效果：



# **LoadingView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，只需要将precentpositionlayout.har和loadingview.har复制到entry\libs目录下即可（由于build.gradle中已经依赖的libs目录下的\*.har，因此不需要在做修改）。

## **修改主页面的布局文件**

修改主页面的布局文件ability\_main.xml，将跟组件容器修改为**com.isoftstone.precentpositionlayout.PrecentPositionLayout**，然后在增加4个**com.isoftstone.loadingview.LoadingView**组件，分别位于屏幕的左上，左下，右上，右下的位置，长度和宽度都占屏幕的50%。修改后代码如下：

<?xml version**="1.0"** encoding**="utf-8"**?>  
<com.isoftstone.precentpositionlayout.PrecentPositionLayout  
 xmlns:ohos**="http://schemas.huawei.com/res/ohos"** ohos:height**="match\_parent"** ohos:width**="match\_parent"** ohos:orientation**="vertical"** ohos:id **= "$+id:layout\_main"**>  
  
 <com.isoftstone.loadingview.LoadingView  
 ohos:id**="$+id:text\_helloworld"** ohos:height**="500"** ohos:width**="500"** ohos:left\_margin**="0"** ohos:top\_margin**="0"** />  
  
 <com.isoftstone.loadingview.LoadingView  
 ohos:id**="$+id:text\_helloworld2"** ohos:height**="500"** ohos:width**="500"** ohos:left\_margin**="500"** ohos:top\_margin**="0"** />  
  
 <com.isoftstone.loadingview.LoadingView  
 ohos:id**="$+id:text\_helloworld3"** ohos:height**="500"** ohos:width**="500"** ohos:left\_margin**="0"** ohos:top\_margin**="500"** />  
  
 <com.isoftstone.loadingview.LoadingView  
 ohos:id**="$+id:text\_helloworld4"** ohos:height**="500"** ohos:width**="500"** ohos:left\_margin**="500"** ohos:top\_margin**="500"** />  
  
</com.isoftstone.precentpositionlayout.PrecentPositionLayout>

## **修改MainAbilitySlince的UI加载代码**

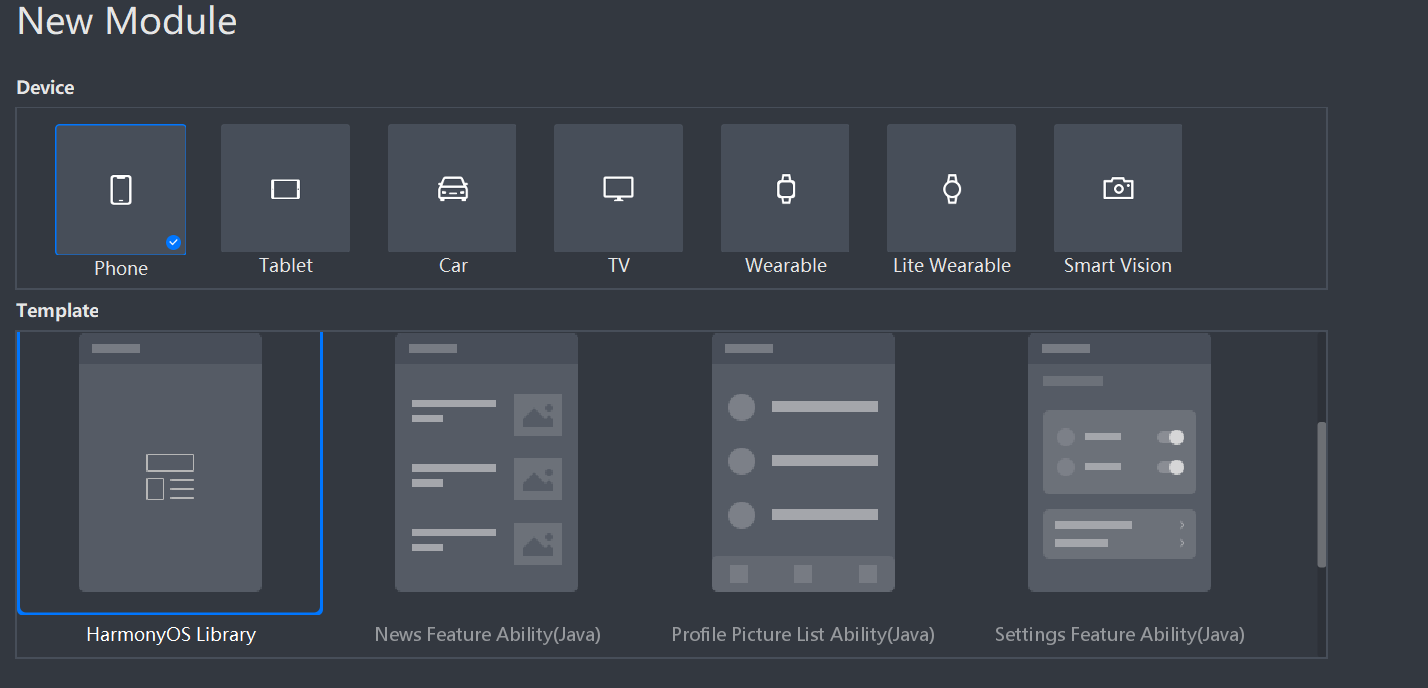
在MainAbilitySlince类的onStart函数中，增加如下代码。

**@Override**public void onStart(Intent intent) {  
 super.onStart(intent);  
 super.setUIContent(ResourceTable.*Layout\_ability\_main*);  
  
 PrecentPositionLayout precentPositionLayout = (PrecentPositionLayout)findComponentById(ResourceTable.*Id\_layout\_main*);  
 precentPositionLayout.AutoSize();  
  
 LoadingView loadingView1 = (LoadingView)findComponentById(ResourceTable.*Id\_text\_helloworld*);  
 loadingView1.SetType(LoadingView.LoadingViewType.*WATER*);  
 loadingView1.addDrawTask(loadingView1);  
  
 LoadingView loadingView2 = (LoadingView)findComponentById(ResourceTable.*Id\_text\_helloworld2*);  
 loadingView2.SetType(LoadingView.LoadingViewType.*BALLOON*);  
 loadingView2.addDrawTask(loadingView2);  
  
 LoadingView loadingView3 = (LoadingView)findComponentById(ResourceTable.*Id\_text\_helloworld3*);  
 loadingView3.SetType(LoadingView.LoadingViewType.*FISH*);  
 loadingView3.addDrawTask(loadingView3);  
  
 LoadingView loadingView4 = (LoadingView)findComponentById(ResourceTable.*Id\_text\_helloworld4*);  
 loadingView4.SetType(LoadingView.LoadingViewType.*CIRCLE*);  
 loadingView4.addDrawTask(loadingView4);  
  
}

# **LoadingView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LoadingView类**

新建一个LoadingView类，继承自Component类，实现Component.DrawTast接口，并重写onDraw方法，代码如下：

// 绘制类  
private LoadingRander loadingRander;

**@Override**public void onDraw(Component component, Canvas canvas) {  
 // 获取组件的大小，进行绘制  
 DimensFloat pt = getComponentSize();  
 Rect rect = new Rect(0,0,pt.getSizeXToInt(),pt.getSizeYToInt());  
 loadingRander.draw(canvas, rect);  
}

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：

// 动画  
private AnimatorValue animatorValue;

// 动画侦听函数  
private final AnimatorValue.ValueUpdateListener mAnimatorUpdateListener  
 = new AnimatorValue.ValueUpdateListener() {  
 **@Override** public void onUpdate(AnimatorValue animatorValue, float v) {  
 if (loadingRander != null) {  
 loadingRander.setProgess(v);  
 }  
 invalidate();  
 }  
};  
  
private void init() {  
 // 启动动画

animatorValue = new AnimatorValue();  
 animatorValue.setCurveType(Animator.CurveType.*LINEAR*);  
 animatorValue.setDelay(100);  
 animatorValue.setLoopedCount(Animator.*INFINITE*);  
 animatorValue.setDuration(2000);  
 animatorValue.setValueUpdateListener(mAnimatorUpdateListener);  
 animatorValue.start();  
}

增加一个设置类型的函数SetType，代码如下

public enum LoadingViewType {  
 // 支持的类型  
 *WATER*, *BALLOON*, *FISH*, *CIRCLE*;  
}

// 设置动画的类型  
public boolean SetType(LoadingViewType type) {  
 switch (type) {  
 case *WATER*:  
 loadingRander = new LoadingRanderWatter();  
 break;  
 case *BALLOON*:  
 loadingRander = new LoadingRanderBalloon();  
 break;  
 case *FISH*:  
 loadingRander = new LoadingRanderFish();  
 break;  
 case *CIRCLE*:  
 loadingRander = new LoadingRanderCircle();  
 break;  
 default:  
 return false;  
 }  
 return true;  
}

## **新建一个LoadingRander类**

LoadingRander是动画绘制的基类，它有两个对外接口，代码如下。

public class LoadingRander {  
 protected float mProgress;  
 protected float mWidth;  
 protected float mHeight;  
 protected float mTextSize;  
  
 public LoadingRander() {  
 }  
 // 设置进度  
 public void setProgess(float progress) {  
 mProgress = progress;  
 }  
 // 绘制  
 protected void draw(Canvas canvas, Rect bounds) {  
 return;  
 }  
}

## **新建四个LoadingRander的子类**

分别完成四种风格动画的绘制（重写基类的draw函数）。

## **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loadingview> bulid > outputs > har目录中，获取生成的HAR包。

