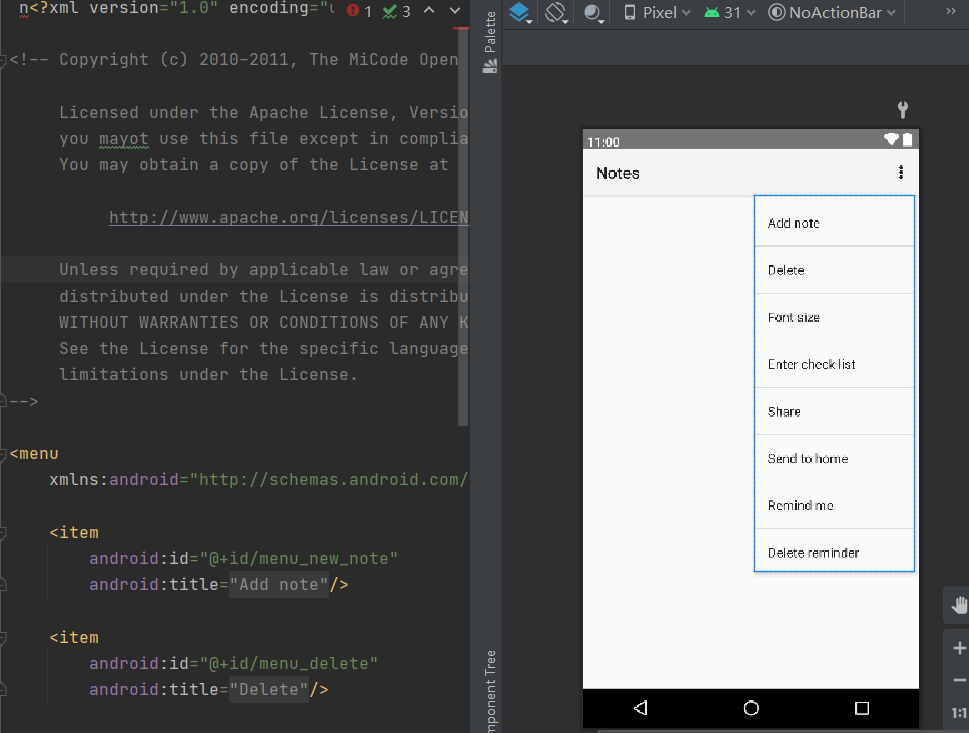
**小米便签维护：改变字体颜色**

20181105509 欧者

小米便签中字体能改变大小但是不能改变颜色，于是我打算设计一个改变便签颜色的功能。

1. **步骤**

首先找到小米便签功能汇总的包。

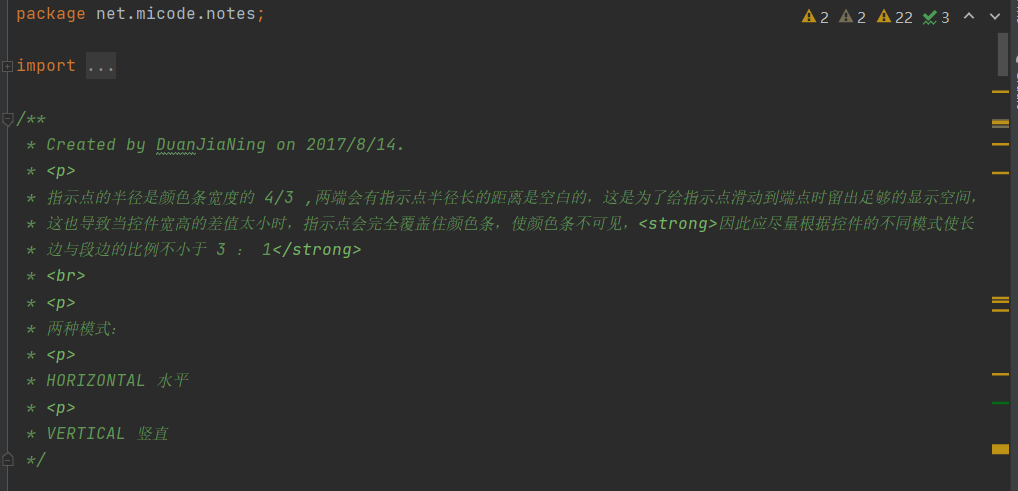


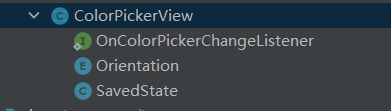
想办法在这部分代码里面加入我想要的功能：字体变换。

仿照原有代码加入改变字体颜色的变量后显示有错误，找到如下包：



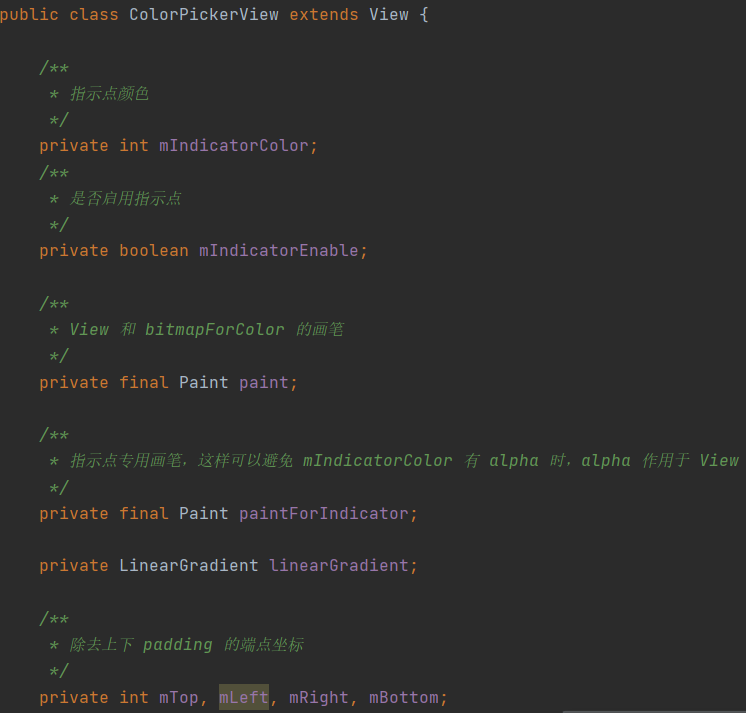
发现小米便签本身就有颜色板块，我想要将此包化为己用，改变一些变量后发现还是报错。





于是我自己创建了一个包。

定义了一个函数，绘制了一个长方体和一个圆形，将长方体一定长度定义为某种颜色，当拖动或点击圆形到某一长度时改变字体颜色。

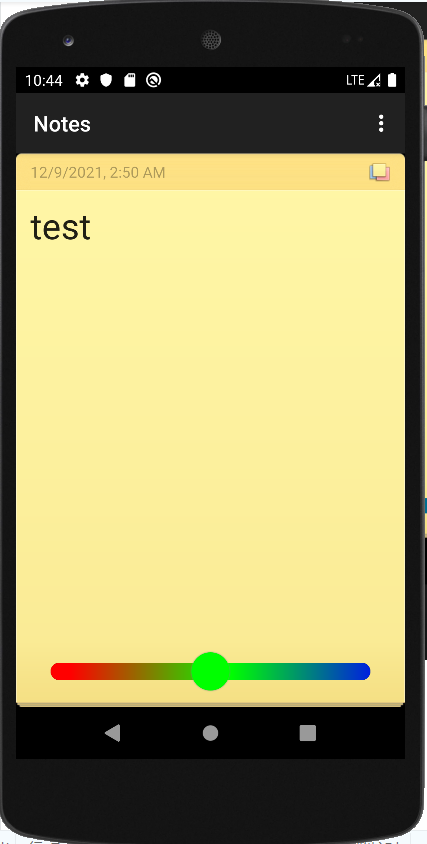


实现函数如下：

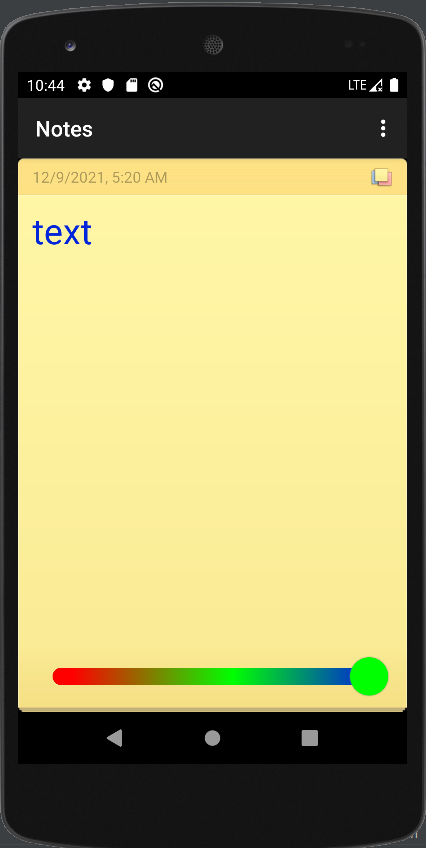
public interface OnColorPickerChangeListener {  
  
 */\*\*  
 \* 选取的颜色值改变时回调  
 \*  
 \** ***@param*** *picker ColorPickerView  
 \** ***@param*** *color 颜色  
 \*/* void onColorChanged(ColorPickerView picker, int color);  
  
 */\*\*  
 \* 开始颜色选取  
 \*  
 \** ***@param*** *picker ColorPickerView  
 \*/* void onStartTrackingTouch(ColorPickerView picker);  
  
 */\*\*  
 \* 停止颜色选取  
 \*  
 \** ***@param*** *picker ColorPickerView  
 \*/* void onStopTrackingTouch(ColorPickerView picker);  
 }  
  
  
 @Override  
 protected Parcelable onSaveInstanceState() {  
 Parcelable parcelable = super.onSaveInstanceState();  
 SavedState ss = new SavedState(parcelable);  
 ss.selX = curX;  
 ss.selY = curY;  
 ss.color = bitmapForColor;  
 if (mIndicatorEnable) {  
 ss.indicator = bitmapForIndicator;  
 }  
 return ss;  
 }  
  
 @Override  
 protected void onRestoreInstanceState(Parcelable state) {  
 if (!(state instanceof SavedState)) {  
 super.onRestoreInstanceState(state);  
 return;  
 }  
 SavedState ss = (SavedState) state;  
 super.onRestoreInstanceState(ss.getSuperState());  
  
 curX = ss.selX;  
 curY = ss.selY;  
 colors = ss.colors;  
  
 bitmapForColor = ss.color;  
 if (mIndicatorEnable) {  
 bitmapForIndicator = ss.indicator;  
 needReDrawIndicator = true;  
 }  
 needReDrawColorTable = true;  
  
 }  
  
 private class SavedState extends BaseSavedState {  
 int selX, selY;  
 int[] colors;  
 Bitmap color;  
 Bitmap indicator = null;  
  
 SavedState(Parcelable source) {  
 super(source);  
 }  
  
 @Override  
 public void writeToParcel(Parcel out, int flags) {  
 super.writeToParcel(out, flags);  
 out.writeInt(selX);  
 out.writeInt(selY);  
 out.writeParcelable(color, flags);  
 out.writeIntArray(colors);  
 if (indicator != null) {  
 out.writeParcelable(indicator, flags);  
 }  
 }  
 }  
  
 public void setPosition(int x, int y) {  
 if (inBoundOfColorTable(x, y)) {  
 curX = x;  
 curY = y;  
 if (mIndicatorEnable) {  
 needReDrawIndicator = true;  
 }  
 invalidate();  
 }  
 }  
  
  
 */\*\*  
 \* 显示默认的颜色选择器  
 \*/* public void showDefaultColorTable() {  
 setColors(createDefaultColorTable());  
 }  
  
 public int getIndicatorColor() {  
 return mIndicatorColor;  
 }  
  
 public void setIndicatorColor(int color) {  
 this.mIndicatorColor = color;  
 needReDrawIndicator = true;  
 invalidate();  
 }  
  
 public void setOrientation(Orientation orientation) {  
 this.orientation = orientation;  
 needReDrawIndicator = true;  
 needReDrawColorTable = true;  
 requestLayout();  
 }  
}

1. **演示**

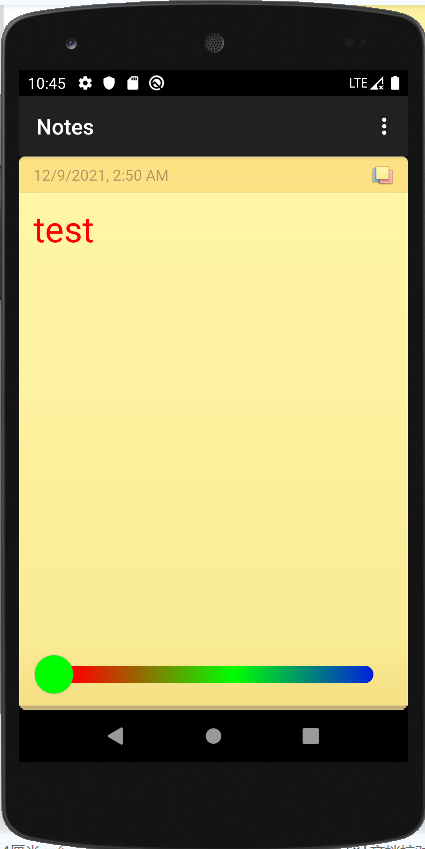
**初始页面：**



右阈值：



左阈值;



1. **不足分析**
2. **可选颜色太少。**
3. **颜色改变后不能存储。**