链滴

Lottie 源码浅探

作者：woyehua

原文链接：https://ld246.com/article/1541638315689

来源网站：链滴

前置知识：

Lottie对动画的变换主要是通过Matrix实现， 因此需要了解Matrix相关知识，可以参考下面的博客：

https://blog.csdn.net/pathuang68/article/details/6991867

一、动画Json传入方法：

Json动画传入的方式定义在LottieComposition.Factory中，分别为：

1. fromAssetFileName：从assets中加载，参数filename指assets中json的名称

2. fromRawFile：从res/raw中加载，参数resId就是raw中json的id

3. fromInputStream：从InputStream中加载

4. fromJsonString：从Json字串中加载

5. fromJsonReader：从JsonReader中加载（如果需要解析的是JsonObject，可以通过new Json eader(newStringReader(jsonObject))的方式加载，不过这种方式不推荐）

二、解析（Lottie中的解析方法都位于com.airbnb.lottie.parser下）：

1. 动画json传入后，最终会调用到LottieComposition.Factory.fromJsonReader方法，该方法会解析事件委托给AsyncCompositionLoader异步线程，而AsyncCompositionLoader会调用LottieC mposition.Factory.fromJsonSync方法，该方法调用LottieCompositionParser.parse方法开始进行画json的解析，并将解析结果生成LottieComposition。

2. AsyncCompositionLoader中会调用LottieCompositionParser.parse方法进行具体解析，在Lo tieCompositionParser.parse方法中：

1) layers会调用parseLayers，parseLayers调用LayerParser.parse解析

2) assets会调用parseAssets解析

3) fonts会调用parseFonts，parseFonts调用FontParser.parse解析

4) chars会调用parseChars，parseChars调用FontCharacterParser.parse解析

5) w会解析成width

6) h会解析成height

7) ip会解析成startFrame

8) op会解析成endFrame

9) fr会解析成frameRate（动画速率）

10) v会解析成version（插件版本），进行版本支持性校验

3. LottieCompositionParser会根据上面的解析结果生成LottieComposition：

1) Rect bounds：通过scaledWidth（width与屏幕密度的乘积）、scaledHeight（height与屏密度的乘积）确定

2) startFrame、endFrame、frameRate就是上一步解析的值

3) layers、layerMap是layers中解析的值

4) precomps是assets中解析的值

5) images是assets中解析出的图片的值

6) characters是chars中解析的值

7) fonts是fonts中解析的值

三、Layer具体解析：

LayerParser解析layers中的内容，其中关键是ks，ks的内容包含了动画用到的一些值，ks的解析是通AnimatableTransformParser.parse方法进行，LayerParser.parse会通过上面解析的数据最终生成Laer：

1) nm：解析为layerName

2) ind：解析为layerId

3) refId：解析为refId

4) ty：解析为layerType（附录1）

5) parent：解析为parentId

6) sw：解析为solidWidth

7) sh：解析为solidHeight

8) sc：解析为solidColor

9) ks：解析为transform

10) tt：解析为mattType

11) masksProperties：数组，里面数据会通过MaskParser.parse解析成Mask并存入masks中

12) shapes：数组，里面数据会通过ContentModelParser.parse解析成ContentModel并存入shape中

13) t：文本，会进一步解析：

(1) d：通过AnimatableValueParser.parseDocumentData解析成text

(2) a：通过AnimatableTextPropertiesParser.parse方法解析成textProperties

14) ef：Lottie中目前不支持这种方式的效果，如果要用，需要将这种效果之直接添加到shape的contents中

15) sr：解析成timeStretch

16) st：解析成startFrame

17) w：解析成preCompWidth

18) h：解析成preCompHeight

19) ip：解析成inFrame

20) op：解析成outFrame

21) tm：解析成timeRemapping

22) cl：解析成cl

AnimatableTransformParser解析ks内容，包括”a”（位置信息）、”p”（位移）、”s”（缩放、”rz”（控制3d图层，暂时不支持）、”r”、”o”、”so”、”eo”，其实质是解析上面各字下配置的k的值（解析方法为KeyframesParser.parse），并根据k的值生成对应的AnimatableValue而k值用到的几个主要解析方式是：

1) PathParser：用于解析k数组，解析成PointF，用于生成AnimatablePathValue。

2) FloatParser：用于解析k数字，解析成Float，用于生成AnimatableFloatValue。

3) IntegerParser：用于解析k数字，解析成Integer，用于生成AnimatableIntegerValue。

4) ScaleXYParser：用于解析k数组，解析成ScaleXY，用于生成AnimatableScaleValue。

AnimatableTransformParser中具体字段解析：

1) a（位置信息）：调用AnimatablePathValueParser.parse解析k中配置的值，并将生成的结果值给anchorPoint

2) p（位移信息）：调用AnimatablePathValueParser.parseSplitPath解析，并将解析生成的结赋值给postion

3) s（缩放信息）：调用AnimatableValueParser.parseScale解析，并将解析结果赋值给scale

4)　rz：3D图层，不支持

5)　r（翻转信息）：调用AnimatableValueParser.parseFloat解析，并将解析结果赋值给rotation

6)　o（不透明度）：调用AnimatableValueParser.parseFloat方法解析，并将解析结果赋值给opacty

7)　so（开始时不透明度）：调用AnimatableValueParser.parseFloat方法解析，并将解析结果赋给startOpacity

8)　eo（结束时不透明度）：调用AnimatableValueParser.parseFloat方法解析，并将解析结果赋给endOpacity

四、AnimatablePathValueParser、AnimatableValueParser重要方法解析：

1.　AnimatablePathValueParser中：

1)　parse：若待解析的JsonReader中的值是数组，将调用PathKeyframeParser.parse方法解析并解析结果存入keyframes（List<Keyframe<PointF>>）中，之后会调用KeyframesParser.setEndFrmes将keyframes中Keyframe的startFrame与endFrame依次串联起来；若不是数组，就直接调用JsnUtils.jsonToPoint方法，并将解析结果生成的Keyframe存入keyframes中。最后根据keyframes生AnimatablePathValue。

2)　parseSplitPath：

(1)　k：调用AnimatablePathValueParser.parse解析，并将解析结果赋值给pathAnimation

(2)　x：若JsonReader值为String，不解析，并将hasExpressions置为true；若不是String，则调用nimatableValueParser.parseFloat解析，并将结果返回给xAnimation

(3)　y：若JsonReader值为String，不解析，并将hasExpressions置为true；若不是String，则调用nimatableValueParser.parseFloat解析，并将结果返回给yAnimation

如果pathAnimation不为空（k直接成功），则直接返回pathAnimation；否则返回通过xAnimation yAnimation生成的AnimatableSplitDimensPathValue。

2.　AnimatableValueParser中parseFloat、parseInteger、parsePoint等方法中最终会调用pars方法解析，而parse方法则调用KeyframesParser.parse解析，：

1)　parseFloat：调用parse并将FloatParser作为最终解析方法，解析结果生成AnimatableFloatVaue

2)　parseInteger：调用parse并将IntegerParser作为最终解析方法，解析结果生成AnimatableIntgerValue

3)　parsePoint：调用parse并将PointFParser作为最终解析方法，解析结果生成AnimatablePointalue

4)　parseScale：调用parse并将ScaleXYParser作为最终解析方法，解析结果生成AnimatableScalValue

5)　parseShapeData：调用parse并将ShapeDataParser作为最终解析方法，解析结果生成AnimatbleShapeValue

6)　parseDocumentData：调用parse并将DocumentDataParser作为最终解析方法，解析结果生

AnimatableTextFrame

7) parseColor：调用parse并将ColorParser作为最终解析方法，解析结果生成AnimatableColorValue

8) parseGradientColor（渐变色）：调用parse并将GradientColorParser作为最终解析方法，解结果生成AnimatableGradientColorValue

3. KeyframesParser.parse：

Parse方法中首先会判断JsonReader值是否为STRING类型，如果是就直接返回空数据集keyframes（List<Keyframe<T>>），若不是，则解析JsonReader中k中配置的值，k中的值基本是3种类型，分别数字数组、对象数组、对象：

1) 若k中值为数组，则判断数组中元素是否为数字，若是，则调用KeyframeParser.parse解析整数组（animated为false）；若不是数字，则调用KeyframeParser.parse依次解析数组中元素（animaed为true）

2) 若k中值不是数组，则调用KeyframeParser.parse解析整个数组（animated为false）

上面的解析结果都会存入keyframes中，最后会调用setEndFrames方法将keyframes中的Keyframe startFrame与endFrame串联，最后将keyframes返回给调用者

4. KeyframeParser：

1) parse：判断传入的animated值，若为true，则调用parseKeyframe；否则调用parseStaticVale

2) parseKeyframe方法是用于解析k中对象数组中的值：

(1) t：解析成startFrame

(2) s：调用传入的最终解析方法解析成stratValue

(3) e：调用传入的最终解析方法解析成endValue

(4) o：调用JsonUtils.jsonToPoint解析成cp1

(5) i：调用JsonUtils.JsonToPoint解析成cp2

(6) h：解析成hold（若h值为1则为true，否则为false）

(7) to：调用JsonUtils.jsonToPoint解析成pathCp1

(8) ti：调用JsonUtils.jsonToPoint解析成pathCp2

解析完后会对解析的值进行进一步操作：

(1) 若hold为ture，将endValue设置为startValue，并将interpolator设置为LINEAR_INTERPOLAOR；

(2) 若hold为false，并且cp1与cp2均不为空，则通过MiscUtils.clamp筛选出合适的cp1、cp2的值并通过PathInterpolatorCompat.create创建interpolator（其中有PathInterpolator缓存的逻辑，了解的可以看下）。

(3) 若为其他情况，则将interpolator设置为LINEAR_INTERPOLATOR

最后，通过解析的startValue、endValue、startFrame以及创建的interpolator生成Keyframe并返给调用者

3) parseStaticValue：调用传入的最终解析方法解析出value，并根据value生成Keyframe

5. FloatParser.parse：

调用JsonUtils.valueFromObject解析出float值并将结果与传入的scale相乘后返回

6. IntegerParser.parse：

调用调用JsonUtils.valueFromObject解析出float值并将结果与传入的scale相乘，再通过Math.roun取出对应整值并返回

7. PointFParser.parse：

如果reader中值是NUMBER类型，直接通过JsonReader.nextDouble与scale生成PointF；若reader数组或对象，则调用JsonUtils.jsonToPoint方法解析出PointF

8. ScaleXYParser.parse：

若是数组，就调用JsonReader.beginArray开始解析，否则就直接解析。解析JsonReader中连续的个double值，并根据这两个值与scale生成ScaleXY

9. ShapeDataParser.parse：

(1) c：解析成closed

(2) v：调用JsonUitls.jsonToPoints方法解析成pointsArray

(3) i：调用JsonUtils.jsonToPoints方法解析成inTangents

(4) o：调用JsonUtils.jsonToPoints方法解析成outTangents

若解析的pointsArray为空，就构建空的ShapeData

若pointsArray不为空，则先取出pointsArray中第一个值，将其设置为initialPoint。然后从第二个值始循环pointsArray中的值，在for循环中，根据pointsArray中当前的值与inTangents中当前值生成shapeCp1，pointsArray中前一个值与outTangents中前一个值生成shapeCp2，然后通过shapeCp1与sapeCp2以及pointsArray当前值生成CubicCurveData并加入curves中。

然后判断closed的值，如果closed为true（标志这个Shape是封闭图形），则将pointsArray第一值为当前值，最后一个值作为前一个值，进行上一步for循环中的操作，并将生成的CubicCurveData也入curves中。

最后根据initialPoint、closed、curves生成ShapeData。

10. DocumentDataParser.parse：

(1) t：解析成text

(2) f：解析成fontName

(3) s：解析成size

(4) j：解析成justification

(5) tr：解析成tracking

(6) lh：解析成lineHeight

(7) ls：解析成baselineShift

(8) fc：调用JsonUtils.jsonToColor解析成fillColor

(9) sc：调用JsonUtils.jsonToColor解析成strokeColor

(10) sw：解析成strokeWidth

(11) of：解析成strokeOverFill

最后通过上面解析的值生成DocumentData

11. ColorParser.parse：

若是数组，调用JsonReader.beginArray开始解析，否则直接解析。取JsonReader中连续的4个double值，分别解析成r、g、b、a，若这四个值都小于，则说明它们被配置成色值的比例，将它们分别乘以55，最后调用Color.argb生成颜色值（int类型）

12. GradientColorParser.parse：

若为数组，则调用reader.beginArray开始解析，否则直接解析。将JsonReader中全部的double值析出并存入array中，然后将array中的数据每4个进行循环，这四个值中第一个存入positions，第二三、四分别为r、g、b值，通过r、g、b生成color值后存入colors，之后通过positions、colors值生gradientColor并通过addOpacityStopsToGradientIfNeeded设置透明度（透明度是存放在颜色值后面，不被4整除的部分），最后将gradientColor返回给调用者

13. JsonUtils：

1) jsonToColor：取出传入的JsonReader中连续的3个double值，分别设置为r、g、b，然后调用olor.argb方法生成颜色值

2) jsonToPoints：若是数组，循环数组中的值，调用jsonToPoint解析出PointF并存入points，后将points返回

3) jsonToPoint：

(1) 若传入的JsonReader为数字，调用jsonNumbersToPoint并返回

(2) 若传入的JsonReader为数组，调用jsonArrayToPoint并返回

(3) 若传入的JsonReader为对象，调用jsonObjectToPoint并返回

4) jsonNumbersToPoint：

取出JsonReader连续的两个double值，分别置成x、y，通过x、y、scale生成PointF

5) jsonArrayToPoint：

取出数组中连续的两个double值，分别置成x、y，通过x、y、scale生成PointF

6) jsonObjectToPoint：

取出对象中的x值置为x，对象中的y值置为y，通过x、y、scale生成PointF

7) valueFromObject：

若传入的JsonReader为数字，返回第一个double值；若是数组，返回数组中第一个值。

五、Shape解析（layers中的shapes）：

LayerParser会调用ContentModelParser.parse将shapes数组依次解析成ContentModel并存入shaps中。

ContentModleParser.parse首先会解析ty，之后根据ty解析成不同的ContentModel（附录2）。

1. gr：调用ShapeGroupParser.parse解析成ShapeGroup，这个是Shape组，里面会包含各种子hape。

ShapeGroupParser中会调用ContentModelParser.parse将it数组中的内容依次解析成ContentMode，并存入items中。

2. st：调用ShapeStrokeParser.parse解析成ShapeStroke，Shape线条的信息。

ShapeStrokeParser.parse中会解析如下字段：

1) nm：name

2) c：color，调用AnimatableValueParser.parseColor解析

3) w：width，调用AnimatableValueParser.parseFloat解析

4) o：opacity，调用AnimatableValueParser.parseInteger解析

5) lc：capType，ShapeStroke.LineCapType类型（附录3）

6) lj：joinType，ShapeStroke.LineJoinType类型（附录4）

7) d：数组，首先会解析n、v（v会解析成val）：

(1) n为o，将val赋值给offset

(2) n为d或g，将val添加到lineDashPattern中

解析完，如果lineDashPattern的个数如果是1，就将该数据再加入lineDashPattern一次

最后会根据上面解析出的值生成ShapeStroke

3. gs：调用GradientStrokeParser.parse解析成GradientStroke

1) nm：name

2) o：opacity

3) t：gradientType，渐变类型（GradientType.Linear或GradientTypeRadial）

4) s：startPoint

5) e：endPoint

6) w：width

7) lc：capType

8) lj：joinType

9) d：lineDashPattern或offset

10) g：会进一步解析g中的字段

(1) p：points

(2) k：color，调用AnimatableValueParser.parseGradientColor并传入points解析

最后根据解析的值生成GradientStroke

4. fl：调用ShapeFillParser.parse解析成ShapeFill

1) nm：name

2) c：color

3) o：opacity

4) fillEnabled：fillEnabled，boolean类型

5) r：fillTypeInt，int类型，之后会根据fillTypeInt的值设置fillType

最后根据解析的值生成ShapeFill

5. gf：调用GradientFillParser.parse解析成GradientFill

1) nm：name

2) g：color，AnimatableGradientColorValue类型

3) o：opacity

4) t：gradientType

5) s：startPoint

6) e：endPoint

7) r：fillType

6. tr：调用AnimatableTransFormParser.parse解析成AnimatableTransform

7. sh：调用ShapePathParser.parse解析成ShapePath，Shape的绘制路径

1) nm：name

2) ind：ind

3) ks：shape，调用AnimatableValueParser.parseShapeData解析

8. el：调用CircleShapeParser.parse解析成CirCleShape

1) nm：name

2) p：position

3) s：size

4) d：reversed，boolean类型，通过d的值是否为3确定

9. rc：调用RectangleShapeParser.parse解析成RectangleShape

1) nm：name

2) p：position

3) s：size

4) r：roundedness，调用AnimatableValueParser.parseFloat解析

10. tm：调用ShapeTrimPathParser.parse解析成ShapeTrimPath

1) s：start，AnimatableValueParser.parseFloat解析

2) e：end，AnimatableValueParser.parseFloat解析

3) o：offset，AnimatableValueParser.parseFloat解析

4) nm：name

5) m：type，ShapeTrimPath.Type类型，附录5

11. sr：调用PolystarShapeParser.parse解析成PolystarShape

1) nm：name

2) sy：type，PolystarShape.Type，附录6

3) pt：points

4) p：position

5) r：totation

6) or：outerRadius，AnimatableValueParser.parseFloat解析

7) os：outerRoundedness，AnimatableValueParser.parseFloat解析

8) ir：innerRadius，AnimatableValueParser.parseFloat解析

9) is：innerRoundedness，AnimatableValueParser.parseFloat解析

12. mm：调用MergePathParser.parse解析成MergePath，只支持KitKat及之后的版本

1) nm：name

2) mode：mode，MergePaths.MergePathsMode类型，附录7

13. rp：调用RepeaterParser.parse解析成Repeater

1) nm：name

2) c：copies，AnimatableValueParser.parseFloat解析

3) o：offset，AnimatableValueParser.parseFloat解析

4) tr：transform

六、Assets解析（assets）：

LottieCompositionParser中调用parseAssets方法解析assets中的内容。assets中内容分两类，一类图层信息，一类是图片信息：

1. 图层信息：

图层信息解析出的内容存放在precomps中，主要解析id与layers的内容。

assets中的layers与外层的layers一样是调用LayerParser.parse解析

2. 图片信息（解析出的值会生成LottieImageAsset并存入images中）：

1) id：解析成id

2) w：解析成width

3) h：解析成height

4) u：解析成relativeFolder

5) p：解析成imageFileName

七、LottieDrawable构建：

动画json解析完成后，会生成LottieComposition，onCompositionLoaded方法中将LottieComposition设置给LottieDrawable。

LottieDrawable的setComposition方法会通过buildCompositionLayer、animator.setCompositio、setScale、updateBounds、等方法重新构建LottieDrawable

buildCompositionLayer会重新构建LottieDrawable中的CompositionLayer。

animator.setComposition会重置animator（LottieValueAnimator）minFrame、maxFrame、frae与lastFrameTimeNs的信息，minFrame会取原minFrame与composition的startFrame（json中的i值）中的最大值，maxFrame会取原maxFrame与composition的endFrame（json中的op值）中的小值，frame会通过一系列比较获取。

八、CompositionLayer构建：

CompositionLayer中存储了动画的所有层级的信息。

它在LottieDrawable中构建

compositionLayer = new CompositionLayer(this,LayerParser.parse(composition), composition getLayers(), composition);

LayerParser.parse(composition)构建了最外层的Layer，高和宽是json最外层的h、w的值。

composition.getLayers()获取的是json最外层layers中的内容。

CompositionLayer的构造方法会遍历传入的composition.getLayers中的Layer，并将Layer通过BaseLayer.forModel方法，根据layerType生成不同的BaseLayer，对应关系如下表：

layerType

BaseLayer子类

备注

Shape（ty=4）

ShapeLayer

PreComp（ty =0）

CompositionLayer

也就是Assets中的内容

Solid（ty=1）

SolidLayer

Image（ty=2）

ImageLayer

Null（ty=3）

NullLayer

Text（ty=5）

TextLayer

Unknown/default

null

生成的BaseLayer会存入layerMap。

之后会根据前一个BaseLayer的matteType（配置文件中的tt）值判断是否为mattedLayer，若是MatedLayer就将当前的BaseLayer设置为前一个BaseLayer的MatteLayer，否则就添加到layers最前的置。下表标识该layer是否为MattedLayer

matteType

是否为MattedLayer

Add/Invert（tt=1或2）

是

其他情况

否

最后会遍历layerMap，找到每一个BaseLayer的parentLayer（根据BaseLayer的parentId查找，parntId对应于json中的parent），并设置到该BaseLayer中。

九、绘制：

Json解析完成后，会调用setComposition方法，该方法中会重新设置LottieComposition，并调用LotieDrawable.setComposition方法刷新LottieDrawable。

上面步骤完成后会调用setImageDrawable、requestLayout方法重绘LottieAnimationView。

LottieAnimationView的onDraw方法会调用LottieDrawable的draw方法（详细步骤需要查看ImageVew的绘制流程）。下面具体分析下LottieDrawable的draw流程。

首先会确定动画的scale，scale是通过动画外层的width与height与canvas的比例确定，取高、宽比中较小的值。若scale后的动画大于canvas，会调用canvas的translate、scale方法重置canvas。然会将获取的scale填入矩阵matrix。

之后会调用BaseLayer.draw方法绘制动画中的全部Layer，具体步骤如下：

BaseLayer.draw()方法中有三个参数，分别是canvas（画布）、parentMatrix（最外层动画Matrix、parentAlpha（最外层透明度）。

1. 调用buildParentLayerListIfNeeded方法构建出CompositionLayer的全部parentLayer并添加parentLayers中。

2. 会将parentMatrix设置为matrix，并将parentLayers中的Layer动画Matrix与传入的matrix相。

3. 然后，根据传入的parentAlpha与该BaseLayer的transform的opacity属性（即json中{“layer”:[{“ks”:{“o”:{...}}}]}中从o中解析出来的值，一般就是o中k的值）计算出透明度

4. 如果该BaseLayer没有matteLayer与mask，就将canvas、matrix、alpha传入drawLayer方法一步绘制，LottieDrawalbe中调用的是CompositionLayer的drawLayer方法

十、CompositionLayer绘制：

1. CompositionLayer的drawLayer中首先保存canvas状态， 然后通过构建时生成的Layer的宽、（具体分析见CompositionLayer构建）设置到newClipRect（RectF类型），然后将newClipRect按步传入的parentMatrix进行变换。

2. 之后，会循环CompositionLayer中的layers（见CompositionLayer的构建），并调用对应BasLayer的draw方法将所有layer依次绘制出来，同时，会按照newClipRect重新裁剪canva（这段代码能会造成动画显示不全）。最后，将canvas设置回之前的状态

十一、 ShapeLayer构建：

CompostionLayer构造方法中遍历解析出的layers（见三），通过BaseLayer.forModel根据生成对的BaseLayer。

ShapeLayer对应的type为4，BaseLayer.forModel直接调用ShapeLayer构造函数生成ShapeLayer。

ShapeLayer构造函数首先通过Layer中的shapes（List<ContentModel>）构建shapeGroup，然后过shapeGroup构建出contentGroup。（lottie文件中也可能会配置ShapeGroup即”gr”。之后会用ContentGroup的setContents，setContents会循环ContentGroup中的contents，并调用Content setContents方法将其前后的contents传入。

ContentGroup构造方法中：

1. 通过contentsFromModels方法将Layer中的shapes（即List<ContentModel>）通过toConte t方法全部转换成对应的Content（附录2），并存入contents中

2. 通过findTransform方法取出该ShapeLayer对应的AnimatableTransform（”tr”），然后通 AnimatableTransform生成TransformKeyframeAnimation赋值给transformAnimation

3. 找出contents中全部的GreedyContent子类存入greedyContents中，并对greedyContents进进一步操作。

十二、 ShapeLayer绘制：

ShapeLayer的drawLayer会委托给contentGroup.draw方法进行。contentGroup.draw中首先会处动画与透明度，之后遍历构造时生成的contents，调用其中DrawingContent的子类的draw进行实际绘制。

DrawingContent子类见附录2

DrawingConteng实际上有3种：

1. 画线图（StrokeContent、GradientStrokeContent）：

StrokeContent与GradientStrokeContent的区别是线条的颜色是否为渐变色。

2. 画填充图（FillContent、GradientFillContent）：

FillContent与GradientFillContent的区别是填充色是否为渐变色。

3. 画需要重复的图（RepeaterContent）：

十三、 StrokeContent构建与绘制：

StrokeContent是BaseStrokeContent的子类，大部分逻辑都在BaseStrokeContent中，以下是BaseStrokeContent的分析：

1. 构造函数：

构造函数中主要是完成的Paint以及Animation的初始化。

2. setContents：

setContents会遍历与其同处一个ContentGroup下的其他Content，找出其中的TrimPathContent其对应的全部PathContent，然后生成PathGroup并存入pathGroups中

3. draw：

1) 通过配置中的width、color、alpha对paint进行设置。

2) 遍历pathGroups，对每一个PathGroup分情况处理：

(1) 若pathGroup中trimPath不为空，则调用applyTrimPath方法，applyTrimPath方法中会通过PathMeasure根据TrimPathContent对PathContent构成的Path进行处理。

PathMeasure用法可以参考：

https://blog.csdn.net/u013831257/article/details/51565591

(2) 若pathGroup中trimPath为空，直接通过canvas.drawPath绘制PathContent构成的Path

十四、 动画过程：

Lottie动画过程是通过LottieValueAnimator进行控制，animator中会维护两个容器，一个存放向Lotte中注册的全部ValueAnimator.AnimatorUpdateListener，一个存放向Lottie中注册的全部AnimatoListener。

ValueAnimator.AnimatorUpdateListener用于动画更新的监听，回调接口是onAnimationUpdate。

AnimatorListener用于动画开始、结束、取消、重复等监听。

LottieDrawable构建的时候会向animator注册一个AnimatorUpdateListener的监听，这个监听会收animator发送的动画更新的回调，在此回调中调用compositionLayer设置progress实现动画的更新。

十五、 附录：

1. LayerType含义：

LayerType对应于json文件layers中的ty字段，映射关系如下：

ty

LayerType

含义

0

PreComp

1

Solid

2

Image

图片

3

Null

4

Shape

绘制形状

5

Text

文字

6

Unknown

2. ContentModel(shape)中type含义，标黄的是DrawingConent的子类：

ty

ContentModel

toContent类型

备注

gr

ShapeGroup

ContentGroup

Shape组，里面包含子Shape

st

ShapeStroke

StrokeContent

Shape描边信息，如线的颜色、宽度、透明度等

gs

GradientStroke

GradientStrokeContent

fl

ShapeFill

FillContent

gf

GradientFill

GradientFillContent

tr

AnimatableTransform

null

Shape的动画

sh

ShapePath

ShapeContent

Shape的绘制路径

el

CircleShape

EllipseContent

rc

RectangleShape

RectangleContent

tm

ShapeTrimPath

TrimPathContent

修剪路径

sr

PolystarShape

PolystarContent

mm

MergePaths

MergePathsContent

rp

Repeater

RepeaterContent

3. ShapeStroke.LineCapType类型：

index

ShapeStroke.LineCapType类型

Paint.Cap（线帽子）

0

LineCapType.Butt

Paint.Cap.BUTT 无线帽

1

LineCapType.Round

Paint.Cap.ROUND 圆线帽

2

LineCapType.Unknown

Paint.Cap.SQUARE 方线帽

4. ShapeStroke.LineJoinType类型：

index

ShapeStroke.LineJoinType类型

Paint.Join（线段连接样式）

0

Miter

Paint.Join.MITER 锐角连接

1

Round

Paint.Join.ROUND 圆弧连接

2

Bevel

Paint.Join.BEVEL 斜接

5. ShapeTrimPath.Type类型：

id

ShapeTrimPath.Type类型

备注

1

Simultaneously

2

Individually

6. PolystarShape.Type类型：

value

PolystarShape.Type类型

备注

1

Star

2

Polygon

7. MergePaths.MergePathsMode类型：

id

MergePaths.MergePathsMode

备注
1
Merge
2
Add
3
Subtract
4
Intersect
5
ExcludeIntersections