第 5 章

界面动画手势与 Web 组件开发

【学习目标】

- 熟悉动画的参数、动画设置，根据需求将动画加入项目中。
- 熟悉基本手势类型，根据应用场景绑定相应的手势。
- 熟悉 Web 组件和 WebController 控制器，掌握 App 与 Web 组件内的网页交互。
- 将动画、手势、Web 组件应用于项目开发中。

【案例目标】

- 下拉列表 loading 动画。
- 旅游攻略详情页。
- 攻略分享页面。
- 自定义弹窗实现。
- 用户注册页面。

5.1 动　　画

5.1.1 动画参数

鸿蒙设备中，可以通过设置组件样式来显示动画，这样就可以让组件动态地旋转、平移、缩放。可以设置动画样式的属性名主要包括以下内容。

（1）transform-origin：用于设置 CSS 变形的旋转、移位、缩放等操作的基点位置。transform-origin 属性值可以是百分比和像素具体的值，也可以是 top、right、bottom、left 和 center 此类的关键词。

（2）transform：用于设置旋转 rotate、扭曲 skew、缩放 scale、移动 translate 以及变形矩阵 matrix 等。具体参数见表 5.1。

表 5.1　transform 参数

参　　数	描　　述
none	不进行任何转换
matrix matrix3d	参数有 6 个值，分别代表 scaleX、scaleY、skewX、skewY、translateX、translateY 参数为 16 个值的 4×4 矩阵
translate translate3d	平移动画属性，支持设置 X 轴和 Y 轴两个维度的平移参数 三个参数，分别代表 X 轴、Y 轴、Z 轴的平移距离
translateX translateY translateZ	X 轴方向平移动画属性 Y 轴方向平移动画属性 Z 轴方向平移动画属性

续表

参　数	描　述
scale scale3d	缩放动画属性，支持设置 X 轴和 Y 轴两个维度的缩放参数 三个参数，分别代表 X 轴、Y 轴、Z 轴的缩放参数
scaleX scaleY scaleZ	X 轴方向缩放动画属性 Y 轴方向缩放动画属性 Z 轴方向缩放动画属性
rotate rotate3d	旋转动画属性，支持设置 X 轴和 Y 轴两个维度的旋转参数 四个参数，前三个分别为 X 轴、Y 轴、Z 轴的旋转向量，第四个为旋转角度
rotateX rotateY rotateZ	X 轴方向旋转动画属性 Y 轴方向旋转动画属性 Z 轴方向旋转动画属性
skew skewX skewY	两个参数，分别为 X 轴和 Y 轴的 2D 倾斜角度 X 轴方向倾斜动画属性 Y 轴方向倾斜动画属性
perspective	3D 透视场景下镜头距离元素表面的距离

5.1.2　动画属性(animation)

animation 属性接口包含尺寸属性、布局属性、位置属性等多种类型，用于控制属性组件的行为。针对当前界面的组件，其部分属性(如位置属性)的变化会引起 UI 的变化。添加动画可以让属性值从起点逐渐变化到终点，产生持续的动画效果。根据变化时是否能够添加动画，可以将属性分为可动画属性和不可动画属性。不可动画属性如 zIndex、focusable 等。可动画属性的参数具有连续性，可通过插值方法来填补数据点之间的空隙，达到视觉上的连续效果。常见的可动画属性如表 5.2 所示。

表 5.2　可动画属性

分　类	说　明
布局属性	位置、大小、内边距、外边距、对齐方式、权重等
仿射变换	平移、旋转、缩放、锚点等
背景	背景颜色、背景模糊等
内容	文字大小、文字颜色、图片对齐方式、模糊等
前景	前景颜色
Overlay	Overlay 属性等
外观	透明度、圆角、边框、阴影等

ArkUI 提供属性动画接口 animation 驱动组件按照动画曲线等参数进行连续的变化，渲染出动画效果。下面给出一个简单的动画示例代码。第 19、20 行代码实现了按钮单击时 Text 组件的字体大小 textSize 属性值缩小为 16vp，textSize 变量是由@State 装饰器装饰的状态变量，这一缩小过程是由代码第 12～18 行的动画 animation 属性控制实现的。

```
1   @Entry
2   @Component
3   struct AnimationPage {
```

```
4     @State message: string = 'Hello World';
5     @State textSize:number = 50
6     build() {
7       Row() {
8         Column({space:20}){
9           Text(this.message)
10            .fontSize(this.textSize)
11            .fontWeight(FontWeight.Bold)
12            .animation({
13              duration:1000,                    //动画时间
14              iterations:3,                     //重复次数,-1 代表不重复
15              curve:Curve.Smooth,               //动画曲线
16              delay:1000,                       //延迟时间
17              playMode:PlayMode.Alternate       //播放模式
18            })
19          Button('变小')
20            .onClick(()=>{this.textSize = 16})
21        }
22      }
23      .width('100%')
24    }
```

5.1.3 显式动画(animateTo)

ArkUI 也可以使用 animateTo 显式实现动画效果,接口为 animateTo(value: AnimateParam, event: ()= >void): void。AnimateTo 接口参数中,value 为 AnimateParameter 对象(包括时长、Curve 等)类型,event()函数内变量的改变将产生属性动画。因此,可以将上面代码第 12～18 行注释掉,代码第 20 行改为 animateTo()函数实现的动画,代码如下,这能实现 Text 组件的字号由 50vp 在 900ms 时间内渐变为 16vp 的大小。

```
.onClick(()=>{
    //动画方法
    animateTo({duration:900},()=>{this.textSize = 16})
})
```

5.1.4 帧动画(ohos.animator)

ohos.animator 帧动画模块通过返回应用 onFrame 逐帧回调的方式,让开发者在应用的每一帧都可以设置属性值,从而设置了该属性值对应组件的动画效果。与 animation 属性动画相比,帧动画可以让开发者感知动画的过程,可实时修改 UI 监测的值,具有事件可实时响应、可暂停的优点。Animation 有以下 6 个动画属性。

(1) animation-name: 设置需要绑定到选择器的 keyname 名称。

(2) animation-duration: 设置完成动画所花费的时间,单位为 ms。

(3) animation-timing-function: 设置动画的速度曲线。

(4) animation-delay: 设置动画开始之前的延迟。

(5) animation-iteration-count: 设置动画迭代次数。

(6) animation-direction：设置动画在循环中是否反向运动。

createAnimation()是通过鸿蒙接口创建动画对象的方法。使用该方法前必须使用下面的语句导入 Animator。

```
import Animator from "@ohos.animator";
```

createAnimation()方法的参数 options 是一个对象，创建 animator 的语句如下：

```
animator = Animator.createAnimator(options);
```

其中，options 对象所设置的属性主要如下。

(1) duration：动画播放的时长，单位为 ms，默认为 0。

(2) easing：动画插值曲线，默认为 ease(动画开始和结束时的速度较慢)。

(3) delay：动画延时播放时长，单位为 ms，默认为 0，即不延时。

(4) fill：动画启停模式，默认为 none。

(5) direction：动画播放模式，默认为 normal。

(6) iterations：动画播放次数，默认为 1，设置为 0 时不播放，设置为-1 时无限次播放。

(7) begin：动画插值的起点，默认为 0。

(8) end：动画插值的终点，默认为 1。

下面代码实现了通过按钮控制一幅图片的旋转播放与暂停示例。代码第 1 行引入相关依赖。代码第 28～30 行定义了要做旋转动画的图片组件。代码第 6～15 行设置了动画效果的相关属性 options，最为核心的是通过 begin、end 属性来设置动画 onFrame 插值的首帧和尾帧值。代码第 17 行将 options 作为参数创建了动画遥控器。代码第 19～24 行，在 aboutAppear()函数中以逐帧 onFrame 逐帧回调方式将 value 值设置到 this.angle 旋转角度属性中。在 Button 组件的 onClick()函数中通过遥控器控制 Image 组件旋转动画的播放与暂停，对应代码第 32～36 行。

```
1   import animator,{AnimatorOptions,AnimatorResult}from'@ohos.animator';
2   @Entry
3   @Component
4   struct AnimatorCase {
5     @State angle:number = 0
6     options:AnimatorOptions = {                    //动画配置
7       duration:10 * 1000,
8       easing:"linear",
9       delay:0,
10      fill:"none",
11      direction:"normal",
12      iterations:-1,
13      begin:0,                                     //动画 onFrame 插值首帧值
14      end:360                                      //动画 onFrame 插值尾帧值
15    }
16    //创建动画遥控器
17    myAnimator:AnimatorResult = animator.create(this.options)
18    isPlay:boolean = false
19    aboutToAppear(): void {
```

```
20      //onFrame 是动画执行时的结果
21      this.myAnimator.onFrame = (value)=>{
22        this.angle = value
23      }
24    }
25
26    build() {
27      Column({space:20}) {
28        Image($r('app.media.welcome'))
29          .width(200).aspectRatio(1).borderRadius(100)
30          .rotate({angle:this.angle})
31        Button('播放/暂停')
32          .onClick(()=>{
33            this.isPlay = !this.isPlay
34            //遥控器控制动画的播放
35            this.isPlay ? this.myAnimator.play() : this.myAnimator.pause()
36          })
37      }.height('100%') .width('100%')
38    }
39  }
```

5.1.5 转场动画

对始终出现的组件做的动画可应用 5.1.2 节至 5.1.4 节介绍的动画方式实现，组件的出现或消失过程应用动画效果，称之为转场动画。目前转场动画有出现/消失转场、导航转场和模态转场、共享元素转场四大类。页面路由的入场和退场的动效，华为官网推荐使用导航转场和模态转场实现。

对 4.6 节首页的旅游攻略列表实现下拉列表的 loading 动画，效果如图 5.1 所示。当用

(a) (b)

图 5.1 旅游攻略列表下拉 loading 效果图

户下拉滑动图5.1(a)所示的旅游攻略列表时，会出现一个自定义刷新区域，刷新区域中有一个刷新图标并显示“加载中…”信息，如图5.1(b)所示。

实现上述功能的代码如下所示。首先需要定义一个初始值为false的状态变量refreshing(代码第1行)。下面代码的第4～10行自定义了一个刷新区，可以自定义刷新图标(代码第7行)和显示提示文字信息(代码第8行)。用刷新动效组件Refresh将旅游攻略列表组件WaterFlow包裹起来，如代码第13～16行所示。代码第13行中参数refreshing表示组件当前是否处于刷新中状态。true表示处于刷新中的状态，false表示未处于刷新中的状态，默认值为false，该参数支持$$双向绑定变量。参数builder是CustomBuilder类型的，设定自定义刷新区域显示内容。代码第17～22行为Refresh组件设置onRefreshing函数，当进入刷新状态时触发回调，通过定时函数setTimeout()实现两个操作，第一个操作更新旅游攻略列表数据this.articleList(此处省略)，第二个操作更新状态变量refreshing的值为false。

```
1   @State refreshing:boolean = false
2
3   //自定义刷新区
4   @Builder
5   MyRefreshBuilder(){
6     Row(){
7       Image($r('app.media.refreshing')).height(30).width(30)
8       Text("加载中...").fontSize(16).margin({left:15})
9     }.width('100%').height(80).justifyContent(FlexAlign.Center)
10  }
11
12  //旅游攻略列表
13  Refresh ({ refreshing: $$this. refreshing, builder: this. MyRefreshBuilder
()}){
14    WaterFlow(){ ... }
15      .columnsTemplate('1fr 1fr').columnsGap(10).rowsGap(15)
16    }
17      .onRefreshing(()=>{
18              setTimeout(()=>{
19                //此处省略 this.articleList 列表更新过程
20                this.refreshing = false
21              },2000)
22      })
```

5.2　手势(gesture)

5.2.1　手势类型

用户使用手势与设备交互，本节将描述系统支持的核心基础手势，即各平台都支持的基础手势。常见的基础手势有如表5.3所示的7种。

表 5.3 基础手势类型

名　　称	描　　述
TapGesture	单击手势，支持单次单击、多次单击识别
LongPressGesture	长按手势
PanGesture	拖动手势，滑动最小距离为 5vp 时识别成功
PinchGesture	捏合手势
RotationGesture	旋转手势
SwipeGesture	滑动手势，滑动最小速度为 100vp/s 时识别成功
GestureGroup	手势识别组，多种手势组合为复合手势，支持连续识别、并行识别和互斥识别

手势表示由单个或多个事件识别的语义动作（例如触摸、单击和长按）。一个完整的手势可能由多个事件组成，对应手势的生命周期。手势支持的事件如表 5.4 所示。

表 5.4 手势支持的事件

事　　件	具 体 事 件	
触摸	touchstart	手指触摸动作开始
	touchmove	手指触摸后移动
	touchcancel	手指触摸动作被打断，如来电提醒、弹窗
	touchend	手指触摸动作结束
单击	click	用户快速轻敲屏幕
长按	longpress	用户在相同位置长时间保持与屏幕接触

5.2.2 绑定手势(gesture)

ArkUI 框架为开发者提供了灵活且强大的手势事件处理机制，允许开发者针对不同场景为组件绑定特定的手势，并设计个性化的相应逻辑。getsture 方法是最基本的手势绑定方式，可以直接将指定的手势类型绑定到组件上，从而定义了该组件对手势的响应行为。下面代码使用 getsture 方法绑定 TapGesture，定义手势触发后的动作，对应代码的第 3～7 行。当用户单击文本组件时，控制台将输出“TapGesture 被触发”。

```
1  Text('手势')
2    .fontSize(30)
3    .gesture(
4      TapGesture().onAction(()=>{
5          console.info('TapGesture 被触发');
6      })
7  )
```

在复杂的嵌套场景中，可能需要控制手势事件的优先级来决定哪个组件优先响应。priorityGesture 方法允许父组件的手势优先于子组件的相同类型手势被识别。

对于需要同时响应多个相同手势的场景，parallelGesture 提供了解决方案。它使得父子组件的相同手势可以并行触发，实现了多点响应。

受篇幅所限，关于 priorityGesture 和 parallelGesture 的具体用法可参见华为官网文档（https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V13/arkts-gesture-

events-binding-V13)。

5.2.3　各手势接口

1. 单击手势接口 TapGesture

单击手势支持单次单击和多次单击,接口定义如下所示:

```
TapGesture(value?:{count?:number, fingers?:number})
```

参数 count:可选参数,声明该单击手势识别的连续单击次数。默认值为 1,若设置小于 1 的非法值会被转化为默认值。如果配置多次单击,推荐上一次抬起和下一次按下的超时时间为 300ms。

参数 fingers:可选参数,用于声明触发单击的手指数量,最小值为 1,最大值为 10,默认值为 1。当配置多指时,若第一根手指按下 300ms 内没有足够的手指数按下则手势识别失败。

2. 长按手势接口 LongPressGesture

长按手势用于触发长按手势事件,接口定义如下所示:

```
LongPressGesture (value?: {fingers?: number, repeat?: boolean, duration?:
number})
```

参数 fingers:可选参数,用于声明触发长按的手指数量,最小值为 1,最大值为 10,默认值为 1。

参数 repeat:可选参数,用于声明是否连续触发事件回调,默认值为 false。

参数 duration:可选参数,用于声明触发长按所需的最短时间,单位为毫秒,默认值为 500。

3. 拖动手势接口 PanGesture

拖动手势用于触发拖动手势事件,滑动达到最小滑动距离(默认值为 5vp)时拖动手势识别成功,接口定义如下所示:

```
PanGesture (value?: {fingers?: number, direction?: PanDirection, distance?:
number})
```

参数 fingers:可选参数,用于声明拖动手势的手指数量,最小值为 1,最大值为 10,默认值为 1。

参数 direction:可选参数,用于声明触发拖动的手势方向,此枚举值支持逻辑与(&)和逻辑或(|)运算。默认值为 Pandirection.All。

参数 distance:可选参数,用于声明触发拖动的最小拖动识别距离,单位为 vp,默认值为 5。

4. 捏合手势接口 PinchGesture

捏合手势用于触发捏合手势事件,接口定义如下:

```
PinchGesture(value?:{fingers?:number, distance?:number})
```

参数 fingers：可选参数，用于声明触发捏合手势所需要的最少手指数量，最小值为 2，最大值为 5，默认值为 2。

参数 distance：可选参数，最小识别距离，单位为 vp，默认值为 5。

5. 旋转手势接口 RotationGesture

旋转手势用于触发旋转手势事件，接口定义如下：

```
RotationGesture(value?:{fingers?:number, angle?:number})
```

参数 fingers：可选参数，用于声明旋转手势所需要的最少手指数量，最小值为 2，最大值为 5，默认值为 2。

参数 angle：可选参数，用于声明触发旋转手势的最小改变度数，单位为 deg，默认值为 1。

6. 滑动手势接口 SwipeGesture

滑动手势用于触发滑动事件，当滑动速度大于 100vp/s 时可以识别成功，接口定义如下：

```
SwipeGesture (value?: {fingers?: number, direction?: SwipeDirection, speed?:
number})
```

参数 fingers：可选参数，用于声明触发滑动的手指数量，最小值为 1，最大值为 10，默认值为 1。

参数 direction：可选参数，用于声明滑动手势的方向，此枚举值支持逻辑与(&)和逻辑或(|)运算。默认值为 SwipeDirection.All。

参数 speed：可选参数，用于声明触发滑动的最小滑动识别速度，单位为 vp/s，默认值为 100。

5.3 Web 组件

在 App 开发过程中，有时需要在 App 内打开 HTML5 页面，加载和显示网页的任务由浏览器完成。ArkUI 提供了 Web 组件来加载网页，借助它如同在开发的 App 中嵌入了一个浏览器，从而可以轻松展示各式各样的网页。

5.3.1 Web 组件和 WebController 控制器

访问互联网信息需要先申请权限。HarmonyOS 提供了访问的权限 ohos.permission.INTERNET，用于保证这些数据或功能的安全与合理使用。可在 module.json5 文件中编写以下配置信息，其中第 3 行代码为新增的权限设置。

```
1  {
2    "module": {
```

```
    "requestPermissions": [{"name": "ohos.permission.INTERNET"}]
  }
}
```

5.3.2　加载远程网页和本地网页

在 ArkTS 页面中创建一个 Web 组件，只需传入两个参数。其中，参数 src 指定引用网页的路径；参数 controller 为组件的控制器，通过 controller 绑定 Web 组件，用于实现对 Web 组件的控制，如下面代码第 9 行所示，运行效果如图 5.2 所示。

```
1  import webview from '@ohos.web.webview'
2  @Entry
3  @Component
4  struct WebTestPage {
5  controller:webview.WebviewController =new webview.WebviewController();
6    build() {
7      Column() {
8        Text('下面加载网络 HTML 链接').width('100%').height(50)
9        Web({src:`https://www.baidu.com/`,controller:this.controller})
10     }
11     .height('100%') .width('100%')
12   }
13 }
```

图 5.2　加载网络 HTML 链接运行效果图

有时候在真机上测试，会出现代码没有问题但远程网页无法加载的情况，这时可以检查网络连接，或者尝试重启手机。

加载本地网页的方法如下。

(1) Web 组件同样也可以加载本地网页，在 main/resources/rawfile 目录下创建一个