《嵌入式应用开发》

青蛙影院项目—抢先看实验指导手册

版本：V 1.0

目录

（一）实验目的

- 掌握青蛙影院基于 ArkTS 语言的开发。
- 掌握类似抖音短视频功能的开发。
- 掌握如何在 ArkTS 语言中使用 MVVM 开发模式进行开发。

（二）实验涉及知识点

- 基础组件和布局。
- 装饰器的使用。
- 图片轮播 Swiper 组件的使用。
- Stack 组件的使用。
- Video 组件实现短视频播放。
- MVVM 开发模式。

（三）实验准备

参考开发环境：

操作系统：Window 10

开发工具：DevEco Studio 3.1.1

HarmonyOS SDK 版本：API version 9 及以上版本

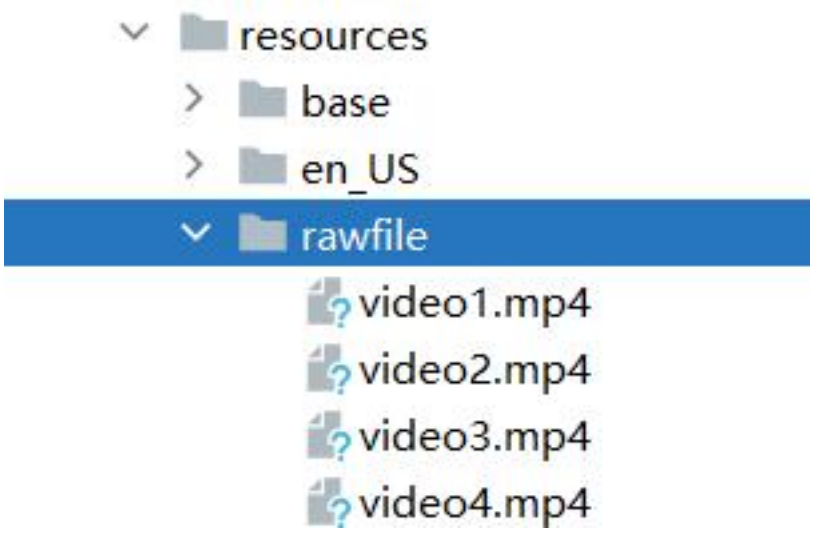
开发语言：ArkTS

内存：8G 及以上

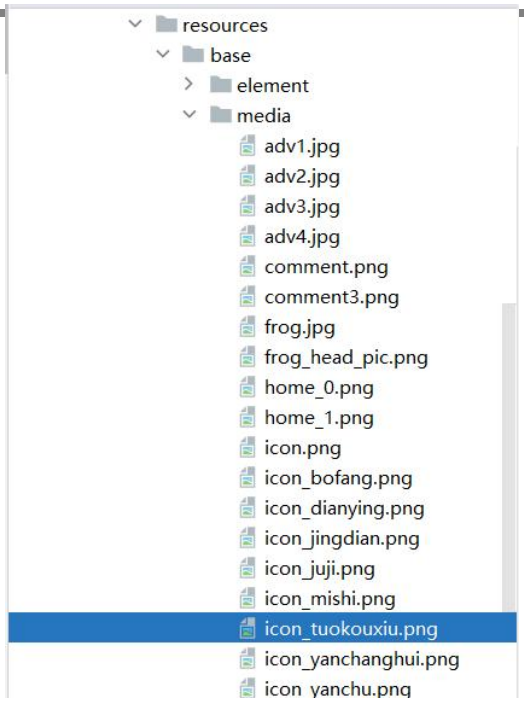
（四）详细实验过程

1 视频组件介绍

准备一些视频素材，将素材导入到 rawfile 目录下。



准备一些图标素材，将素材导入到 media 目录下。



视频的播放需要使用 Video 组件来实现。该组件的声明如下，相关说明见代码注释。

```
interface VideoInterface {
    //设置属性
    (value: VideoOptions): VideoAttribute;
}

declare interface VideoOptions {
    //视频文件
    src?: string | Resource;
    //当前进度控制
    currentProgressRate?: number | string | PlaybackSpeed;
    //前一个Uri 资源
    previewUri?: string | PixelMap | Resource;
    //视频控制器，暂停、播放等操作
    controller?: VideoController;
}
```

专有属性参考 VideoAttribute，常见的专有属性如下。

```
declare class VideoAttribute extends CommonMethod<VideoAttribute> {
   // 自动播放
    autoPlay(value: boolean): VideoAttribute;
    //控制
    controls(value: boolean): VideoAttribute;
    //循环播放
    loop(value: boolean): VideoAttribute;
   //开始播放
    onStart(event: () => void): VideoAttribute;
   //暂停播放
    onPause(event: () => void): VideoAttribute;
   //完成播放
    onFinish(event: () => void): VideoAttribute;
   //...
}
```

了解了这些基本信息，接下来我们分析短视频播放页面的布局。

2 短视频播放页面的布局解析

2.1 整体分析

短视频播放页面的运行效果如下图。

我们对页面进行分割，大概可以得到 3 个部分。

- 第一部分：视频部分。
- 第二部分：头像、点赞、评论、分享部分。
- 第三部分：视频描述部分，有作者、标题、描述等内容。

我们很明显可以看到第二部分在第一部分之上，其实当你的视频是全屏的话，我们也可以理解第三部分在第一部分之上。

我们可以采用 Stack 作为整体的布局样式，让第一部分、第二部分、第三部分依次层叠。这个层叠就要有一定的技巧了，让合适的部分呈现在合适的大小和位置。

2.2 视频部分分析

我们可以让视频显示在最底层，设置它的宽度铺满屏幕，高度自适应。这个时候，我们需要分析，如果视频的高度很短，那么想要有比较好的视觉统一效果，我们将整个页面设置为黑色背景是非常关键的。

2.3 头像和评论图标部分分析

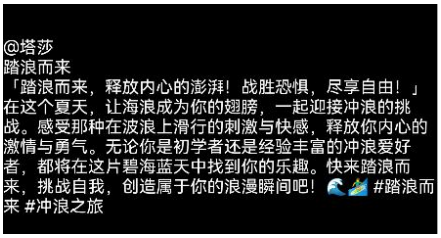
首先，这部分包含了头像、点赞图标和点赞数量文字、评论图标和评论数量文字、分享图标和分享数量文字。我们整体上可以确定采用 Column 布局是最佳选择。

为了有更好的分组效果，我们可以将点赞图标和点赞数量文字作为一组，评论图标和评论数量文字作为一组，分享图标和分享数量文字作为一组进行布局，得到更好的视觉效果，比如每组之间的间距较大，组员之间的间距较小。

头像部分的处理也是非常关键的，我们需要显示的是一个圆形图标。

接下来，最重要的就是如何将这个整体放到屏幕的合适位置，我们需要将其在垂直方向上居中、水平方向靠右，且距离屏幕右边有一定的间距。

2.4 视频描述部分分析



通过上面的图片分析，我们大概得出，总的内容分为三个部分：视频作者、视频标题、视频描述。整体采用 Column 布局，里面放 3 个 Text 即可，注意设置左对齐。

3 视频布局与滑动切换实现

3.1 页面数据分析

我们的视频页面效果如下图所示。

经过对页面数据进行分析，我们知道一个视频大概需要包含如下信息：

- 视频作者
- 视频标题
- 视频描述
- 视频链接或者播放地址
- 视频的分享次数
- 视频的点赞次数

- 是否小红心
- 视频的评论数

3.2 封装数据类

在 bean 包下新建一个 VideoBean.ets 文件，对上述分析的视频信息进行封装，代码如下所示。

```
//视频信息封装
export default class VideoBean {
  id: string;
  author:string //作者
  title: string; //视频标题
  desc: string //描述信息
  src: Resource; //视频链接
  shareCount: number; //分享次数
  likeCount: number; //点赞数
  isLike: boolean; //是否小红心
  commentCount: number; //评论数

  constructor(id: string, author:string,title: string, desc: string, src:
Resource, shareCount: number, likeCount: number, isLike: boolean,
commentCount: number) {
    this.id = id;
    this.author = author
    this.title = title;
    this.desc = desc
    this.src = src;
    this.shareCount = shareCount;
    this.likeCount = likeCount;
    this.isLike = isLike;
    this.commentCount = commentCount;
  }
```

```
}
```

3.3 编写 ViewModel 类

在 MainViewModel.ets 文件中提供数据方法（同样需要在 MainViewModel.ets 中导入 VideoBean.ets）。MainViewModel.ets 中视频模块的代码如下。

```
import ItemBean from '../bean/ItemBean'
import MovieBean from '../bean/MovieBean'
import CinemaBean from '../bean/CinemaBean'
import VideoBean from '../bean/VideoBean'

export class MainViewModel {

  //电影海报轮播图
  ......
  // 常用功能列表数据
  ......
  // 获取热门电影数据
  ......
  //广告轮播图
  ......

  //获取影院数据
  ......
  // 获取预设视频数据
  getAllVideos(): Array<VideoBean> {

    let videos: Array<VideoBean> = [
      new VideoBean('1', '塔莎', '踏浪而来', '「踏浪而来，释放内心的澎湃！战胜恐惧，尽享自由！」在这个夏天，让海浪成为你的翅膀，一起迎接冲浪的挑战。感受那种在波浪上滑行的刺激与快感，释放你内心的激情与勇气。无论你是初学者还是经验丰富的冲浪爱好者，都将在这片碧海蓝天中找到你的乐趣。快来踏浪而来，挑战自我，创造属于你的浪漫瞬间吧！ ♂ #踏浪而来 #冲浪之旅', $rawfile('video1.mp4'), 111, 1231, true, 123),
```

```
        new VideoBean('2', '椰林司机', '椰风起时', '「椰风起时，感受热带海岛的宁静与惬意！」摇曳的椰树、绵延的白沙滩，仿佛置身于天堂般的美景。让微风拂过你的脸庞，听海浪轻轻拍打着沙滩的声音，释放出所有的压力和烦恼。静下心来，享受大自然的馈赠，与心爱的人一起漫步海岛，留下美好的回忆。在这片椰风起时的热带天堂，放下束缚，找回内心的平静和自由。#椰风起时 #热带海岛之旅', $rawfile('video2.mp4'), 122, 1241, false, 234),
        new VideoBean('3', '夜晚凌', '夜航船', '「夜航船，穿越黑夜的旅程，探索未知的奇迹！」当夜幕降临，星光璀璨，乘坐着神秘而豪华的夜航船，启程探索无尽的海洋。船身轻轻摇摆，海风拂面，仿佛置身于一个梦幻般的世界。沿途遇见闪烁的海洋生物，眺望远方的星空，感受神秘与浪漫并存的航海之旅。勇敢的航海家们，跟随这艘夜航船，发现属于你们的宝藏和冒险吧！⚓  #夜航船 #神秘航海之旅', $rawfile('video3.mp4'), 133, 234, true, 345),
        new VideoBean('4', '风清扬', '艳阳高照', '「艳阳高照，阳光洒满大地，让我们一起欢笑享受这美好的夏日！」明媚的阳光洒在肌肤上，温暖而舒适。穿上轻盈的衣裳，踏着轻快的步伐，尽情奔跑在金色的沙滩上。与家人朋友一同嬉戏玩耍，享受大自然的恩赐。感受阳光给予的力量和活力，尽情释放自己的欢愉和快乐。在这个夏日，让艳阳高照成为我们生活的常态，绽放出青春的光芒！    #艳阳高照 #夏日欢乐', $rawfile('video4.mp4'), 144, 3456, false, 456),
    ]

    return videos;

  }

}

export default new MainViewModel()
```

3.4 视频布局与滑动切换

需求分析：我们想要实现类似抖音的向上向下滑动来切换视频。视频的滑动切换，本质上就是一个 Swiper，只不过方向设置为垂直方向而已。然后，我们注意到抖音的短视频播放是自动播放的，且没有播放控制器。

3.4.1 定义页面框架

首先定义出页面框架。我们通过“viewmodel.getAllVideos()”拿到所有的视频数据，通过 ForEach 进行循环遍历，这样就得到了每一个视频的数据。然后我们定义一个 PlayView 自定义组件用来显示我们的视频的所有元素。

打开 LookVideo.ets 文件编写如下所示代码。

```
import viewmodel from '../../viewmodel/MainViewModel'
import VideoBean from '../../bean/VideoBean'

@Preview
@Entry
@Component
export struct LookVideo {
  build() {
    Column() {
      Swiper() {
        ForEach(viewmodel.getAllVideos(), (item: VideoBean, index?: number) =>
{
          PlayView({ videoBean: item })
        }, item => item.toString())
      }
      .vertical(true)
      .width("100%")
      .height("100%")
      .indicator(false)
    }
  }
}
```

PlayView 组件的实现大体框架如下，我们将一个视频页面元素分为三个部分，采用层叠样式布局进行实现。

- 最底层是我们的视频

- 然后是视频右边垂直显示的头像、点赞、评分、分享信息。
- 然后是视频下方的描述信息。

注意：这里设置背景颜色为黑色是非常重要的，只有设置成黑色，才能得到我们想要的结果。

分析示意图如下。

继续在 LookVideo.ets 文件编写如下所示代码。

```
import viewmodel from '../../viewmodel/MainViewModel'
import VideoBean from '../../bean/VideoBean'
```

```
@Preview
@Entry
@Component
export struct LookVideo {
  build() {
    Column() {
      Swiper() {
        ForEach(viewmodel.getAllVideos(), (item: VideoBean, index?: number) =>
{
          PlayView({ videoBean: item })
        }, item => item.toString())
      }
      .vertical(true)
      .width("100%")
      .height("100%")
      .indicator(false)
    }
  }
}



@Component
struct PlayView {
  @State videoBean: VideoBean = null

  build() {
    Stack() {
      //视频
      // 点赞、评论图标列表
      //视频描述信息
    }.backgroundColor(Color.Black)
  }
}
```

3.4.2 视频布局实现

首先，我们看一下视频部分的实现。注意这里，我们设置了视频的尺寸、自动播放、还有不显示控制器。

```
// 视频
 Video({
  src: this.videoBean.src
 }).width("100%")
  .height("40%")
  .autoPlay(true) //自动播放
  .controls(false) //不显示控制器
```

3.4.3 头像、评论等图标布局

接下来，我们看一下头像和评论等图标的布局。这里采用 Column 布局，并设置“alignItems(HorizontalAlign.End)”非常关键，这样它才会显示在屏幕的右边，然后通过 padding 设置一定的边距，这样它距离屏幕右边缘就有一定的距离。

```
  // 点赞、评论图标列表
Column() {
  //头像
  Image($r('app.media.frog_head_pic'))
    .width(50).height(50)
    .borderRadius(25)
    .borderWidth(1)
  // 点赞
  Column({ space: 5 }) {
    Image($r('app.media.like1'))
      .width(50).height(50)
    Text(this.videoBean.likeCount + "").fontColor(Color.White)
  }.alignItems(HorizontalAlign.Center).margin({top:10})
```

```
  //评论
  Column({ space: 5 }) {
    Image($r('app.media.comment3'))
      .width(40).height(40)
    Text(this.videoBean.commentCount + "").fontColor(Color.White)
  }.alignItems(HorizontalAlign.Center).margin({top:10})

  //分享
  Column({ space: 5 }) {
    Image($r('app.media.share3'))
      .width(40).height(40)
    Text(this.videoBean.shareCount + "").fontColor(Color.White)
  }.alignItems(HorizontalAlign.Center).margin({top:10})

}
.width("100%")
.alignItems(HorizontalAlign.End)
.padding({ right: 15 })
```

3.4.4 视频描述

最后，是我们的视频描述信息。

```
//视频描述信息
Column({ space: 5 }) {
  Text("@" + this.videoBean.author).fontColor(Color.White)
  Text(this.videoBean.title).fontColor(Color.White)
  Text(this.videoBean.desc).fontColor(Color.White)
}
.width("100%")
.height("30%")
.offset({ y: "40%" })
.alignItems(HorizontalAlign.Start)
.padding({ left: 5 })
```

运行效果图如下所示，且可以上下滑动，视频可以自动播放。

4 短视频播放页面完整代码

```
import viewmodel from '../../viewmodel/MainViewModel'
import VideoBean from '../../bean/VideoBean'

@Preview
@Entry
@Component
export struct LookVideo {
  build() {
    Column() {
      Swiper() {
        ForEach(viewmodel.getAllVideos(), (item: VideoBean, index?: number) =>
{
          PlayView({ videoBean: item })
        }, item => item.toString())
      }
      .vertical(true)
      .width("100%")
      .height("100%")
      .indicator(false)
    }
  }
}

@Component
struct PlayView {
  @State videoBean: VideoBean = null

  build() {
    Stack() {
      // 视频
      Video({
        src: this.videoBean.src
      }).width("100%")
        .height("40%")
```

```
      .autoPlay(true) //自动播放
      .controls(false) //不显示控制器

    // 点赞、评论图标列表
    Column() {
      //头像
      Image($r('app.media.frog_head_pic'))
        .width(50).height(50)
        .borderRadius(25)
        .borderWidth(1)
      // 点赞
      Column({ space: 5 }) {
        Image($r('app.media.like1'))
          .width(50).height(50)
        Text(this.videoBean.likeCount + "").fontColor(Color.White)
      }.alignItems(HorizontalAlign.Center).margin({top:10})

      //评论
      Column({ space: 5 }) {
        Image($r('app.media.comment3'))
          .width(40).height(40)
        Text(this.videoBean.commentCount + "").fontColor(Color.White)
      }.alignItems(HorizontalAlign.Center).margin({top:15})

      //分享
      Column({ space: 5 }) {
        Image($r('app.media.share3'))
          .width(40).height(40)
        Text(this.videoBean.shareCount + "").fontColor(Color.White)
      }.alignItems(HorizontalAlign.Center).margin({top:10})

    }
    .width("100%")
    .alignItems(HorizontalAlign.End)
    .padding({ right: 15 })
```

```
      //视频描述信息
      Column({ space: 5 }) {
        Text("@" + this.videoBean.author).fontColor(Color.White)
        Text(this.videoBean.title).fontColor(Color.White)
        Text(this.videoBean.desc).fontColor(Color.White)
      }
      .width("100%")
      .height("30%")
      .offset({ y: "40%" })
      .alignItems(HorizontalAlign.Start)
      .padding({ left: 5 })

    }.backgroundColor(Color.Black)
  }
}
```