1 功能概述

视频播放是多场景项目中提供的基础功能，云课堂场景会使用到直播播放器和回放播放器。在云课堂场景模块中，对直播播放器整个布局区域的元素封装为`PLVLCLiveMediaLayout`，对回放播放器整个布局区的元素域封装为`PLVLCPlaybackMediaLayout`，它们共同实现`IPLVLCMediaLayout`接口。

2 使用演示

以下实例了如何创建直播播放器布局、回放播放器布局，以及它们的初始化及视频播放，代码如下：

```
//播放器布局
IPLVLCMediaLayout mediaLayout;
//ViewStub控件
ViewStub videoLyViewStub = findViewById(R.id.plvlc_video_ly);
if (liveRoomDataManager.getConfig().isLive()) {//直播类型
    //设置直播播放器布局
    videoLyViewStub.setLayoutResource(R.layout.plvlc_live_player_layout_view_stub);
    //填充直播播放器布局
    mediaLayout = (IPLVLCMediaLayout) videoLyViewStub.inflate();
    //初始化
    mediaLayout.init(liveRoomDataManager);
    //设置横屏控制栏
    mediaLayout.setLandscapeControllerView(liveLandscapeChannelController);
    //开始播放
    mediaLayout.startPlay();
} else {//回放类型
    //设置回放播放器布局
    videoLyViewStub.setLayoutResource(R.layout.plvlc_playback_player_layout_view_stub);
    //填充回放播放器布局
    mediaLayout = (IPLVLCMediaLayout) videoLyViewStub.inflate();
    //初始化
    mediaLayout.init(liveRoomDataManager);
    //设置PPTView
    mediaLayout.setPPTView(floatingPPTLayout.getPPTView().getPlaybackPPTViewToBindInPlayer());
    //开始播放
    mediaLayout.startPlay();
}
```

上述方法的具体用例可以在polyv demo项目中的`PLVLCCloudClassActivity`找到。

3 接口介绍

3.1 IPLVLCMediaLayout

云课堂场景下，针对播放器布局进行封装的接口。定义了：

1、外部直接调用的方法 2、需要外部响应的事件监听器

```
public interface IPLVLCMediaLayout {

    // <editor-fold defaultstate="collapsed" desc="1、外部直接调用的方法 - common部分，定义 直播播放器布局 和 回放播放器布局 通用的方法">

    /**
     * 初始化
     *
     * @param liveRoomDataManager 直播间数据管理器
     */
    void init(IPLVLiveRoomDataManager liveRoomDataManager);

    /**
     * 开始播放
     */
    void startPlay();

    /**
     * 暂停播放
     */
    void pause();

    /**
     * 恢复播放
     */
    void resume();

    /**
     * 停止播放
     */
    void stop();

    /**
     * 是否在播放中
     *
     * @return true：在播放，false：不在播放
     */
    boolean isPlaying();

    /**
     * 设置系统音量
     *
     * @param volume，音量值，范围[0,100]
     */
    void setVolume(int volume);

    /**
     * 获取系统音量
     *
     * @return 音量值，范围[0,100]
     */
    int getVolume();

    /**
     * 发送弹幕
```

```
     *
     * @param message 弹幕信息
     */
    void sendDanmaku(CharSequence message);

    /**
     * 点击关闭悬浮窗
     */
    void updateOnClickCloseFloatingView();

    /**
     * 获取播放器切换View
     *
     * @return 播放器切换View
     */
    PLVSwitchViewAnchorLayout getPlayerSwitchView();

    /**
     * 获取横屏的聊天布局
     *
     * @return 横屏聊天布局
     */
    PLVLCChatLandscapeLayout getChatLandscapeLayout();

    /**
     * 设置view交互事件监听器
     *
     * @param listener 监听器
     */
    void setOnViewActionListener(OnViewActionListener listener);

    /**
     * 添加播放器状态的监听器
     *
     * @param listener 监听器
     */
    void addOnPlayerStateListener(IPLVOnDataChangedListener<PLVPlayerState> listener);

    /**
     * 添加PPT是否显示状态的监听器
     *
     * @param listener 监听器
     */
    void addOnPPTShowStateListener(IPLVOnDataChangedListener<Boolean> listener);

    /**
     * 是否拦截返回事件
     *
     * @return true：拦截，false：不拦截。如果当前处于横屏状态，会拦截返回事件，并切换到竖屏。
     */
    boolean onBackPressed();

    /**
     * 销毁，销毁播放器及相关资源
     */
    void destroy();
```

```
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="1、外部直接调用的方法 - live部分，定义 直播播放器布局 独有的方法">

    /**
     * 设置横屏控制器
     *
     * @param landscapeControllerView 横屏控制器
     */
    void setLandscapeControllerView(@NonNull IPLVLiveLandscapePlayerController landscapeControllerView);

    /**
     * 更新观看热度
     *
     * @param viewerCount 热度数
     */
    void updateViewerCount(long viewerCount);

    /**
     * 当加入连麦时，更新布局
     *
     * @param linkMicLayoutLandscapeWidth 连麦布局在横屏的宽度
     */
    void updateWhenJoinLinkMic(int linkMicLayoutLandscapeWidth);

    /**
     * 当离开连麦时，更新布局
     *
     * @param shouldStartPlay 是否应该在离开连麦后播放视频
     */
    void updateWhenLeaveLinkMic(boolean shouldStartPlay);

    /**
     * 添加连麦是否开启状态的监听器
     *
     * @param listener 监听器
     */
    void addOnLinkMicStateListener(IPLVOnDataChangedListener<Pair<Boolean, Boolean>> listener);

    /**
     * 添加sei数据监听器
     *
     * @param listener 监听器
     */
    void addOnSeiDataListener(IPLVOnDataChangedListener<Long> listener);
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="1、外部直接调用的方法 - playback部分，定义 回放播放器布局 独有的方法">

    /**
     * 获取视频总时长
     *
     * @return 视频总时长，单位：毫秒
     */
    int getDuration();
```

```
    /**
     * 根据progress占max的百分比，跳转到视频总时间的该百分比进度
     *
     * @param progress 进度
     * @param max      总进度
     */
    void seekTo(int progress, int max);

    /**
     * 设置播放速度
     *
     * @param speed 速度值，建议范围为[0.5,2]
     */
    void setSpeed(float speed);

    /**
     * 获取播放速度
     *
     * @return 速度值
     */
    float getSpeed();

    /**
     * 设置PPTView
     *
     * @param pptView pptView
     */
    void setPPTView(IPolyvPPTView pptView);

    /**
     * 添加播放信息的监听器
     *
     * @param listener 监听器
     */
    void addOnPlayInfoVOListener(IPLVOnDataChangedListener<PLVPlayInfoVO> listener);
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="2、需要外部响应的事件监听器 - 定义 播放器布局中UI控件 触发的交互事件的回调方法">

    /**
     * view交互事件监听器
     */
    interface OnViewActionListener {
        /**
         * 点击显示或隐藏浮窗/连麦布局（直播和回放共有）
         *
         * @param toShow true：显示，false：隐藏
         */
        void onClickShowOrHideSubTab(boolean toShow);

        /**
         * 显示皮肤（直播独有）
         *
         * @param show true表示播放器皮肤显示，false表示播放器皮肤隐藏
```

```
         */
        void onShowMediaController(boolean show);

        /**
         * 横屏发送的消息应同步到聊天室
         *
         * @param message 发送的信息
         * @return <是否发送成功，结果码>
         */
        Pair<Boolean, Integer> onSendChatMessageAction(String message);

        /**
         * 显示公告动作（直播独有）
         */
        void onShowBulletinAction();

        /**
         * 发送点赞动作
         */
        void onSendLikesAction();
    }
    // </editor-fold>

}
```

4 实现介绍

4.1 PLVLCLiveMediaLayout

云课堂场景下的直播播放器布局，实现 `IPLVLCMediaLayout` 接口。

该布局包含的元素有：直播播放器、子播放器、横屏聊天室区、弹幕布局、亮度手势布局、音量手势布局、直播控制栏，等在直播播放区域出现的UI。

下面会列举介绍该布局中涉及到的主要方法。

4.1.1 初始化view方法

`PLVLCLiveMediaLayout` 继承于 `FrameLayout`，在构造器中对使用 `initView` 方法对 `view` 进行了初始化处理。

```
public PLVLCLiveMediaLayout(@NonNull Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
    super(context, attrs, defStyleAttr);
    initView();
}

private void initView() {
    //填充播放器布局到该view中
    LayoutInflater.from(getContext()).inflate(R.layout.plvlc_live_player_layout, this);
    //一系列findViewById，这里不详细列出
    //初始化播放器布局中包含的view
    initVideoView();
    initDanmuView();
    initMediaController();
    initMoreLayout();
    initAudioModeView();
    initLoadingView();
    initSwitchView();
    initLayoutWH();
}
```

4.1.2 初始化数据方法

`PLVLCLiveMediaLayout` 的 `init` 方法是对外API，需要外部传入 `IPLVLiveRoomDataManager` 后进行初始化。

```
//播放器presenter
private IPLVLivePlayerContract.ILivePlayerPresenter livePlayerPresenter;

@Override
public void init(@NonNull IPLVLiveRoomDataManager liveRoomDataManager) {
    this.liveRoomDataManager = liveRoomDataManager;
    //观察直播间数据管理器中的直播详情数据
    observeClassDetailVO();
    //创建播放器mvp-presenter
    livePlayerPresenter = new PLVLivePlayerPresenter(liveRoomDataManager);
    //注册播放器mvp-view
    livePlayerPresenter.registerView(livePlayerView);//这里的livePlayerView是播放器mvp-view
    //初始化播放器mvp-prensenter
    livePlayerPresenter.init();
    //传递presenter给mediaController调用
    mediaController.setLivePlayerPresenter(livePlayerPresenter);
}

//创建播放器mvp-view
private IPLVLivePlayerContract.ILivePlayerView livePlayerView = new PLVAbsLivePlayerView() {
     //...
}
```

这里的 `PLVLivePlayerPresenter` 是使用 `mvp` 模式封装的播放器，用 `presenter` 将 `view` 和 `mode` 隔离开来，一切业务逻辑都是通过 `presenter` 来进行操作，也就是说 `presenter` 是视图的数据的桥梁，视图和数据相隔两端。

4.1.3 设置横屏控制栏方法

`PLVLCLiveMediaLayout` 的 `setLandscapeControllerView` 方法是对外API，需要外部传入 `IPLVLiveLandscapePlayerController` 进行配置。 `PLVLCLiveMediaLayout` 内部已经包含了竖屏控制栏，由于横屏控制栏的层级在云课堂场景是需要覆盖在悬浮窗，连麦控制器之上。因此需要在 `xml` 布局中，调整横屏控制栏的位置后，再传入到 `PLVLCLiveMediaLayout` 中。

```
@Override
public void setLandscapeControllerView(@NonNull IPLVLiveLandscapePlayerController landscapeControllerView) {
    mediaController.setLandscapeController(landscapeControllerView);
}
```

4.1.4 开始播放方法

`PLVLCLiveMediaLayout` 的 `startPlay` 方法是对外API，该方法内部会调用 `PLVLivePlayerPresenter` 的 `start` 方法来进行视频的加载及播放。

```
@Override
public void startPlay() {
    livePlayerPresenter.startPlay();
}
```

`PLVLCLiveMediaLayout` 内部持有 `PLVLivePlayerPresenter` ，因此除了内部可以自由调用播放器相关的方法外，也能将其封装成接口提供给外层调用， `startPlay` 方法就是其中一个例子。

4.2 PLVLCPlaybackMediaLayout

云课堂场景下的回放播放器布局，实现 `IPLVLCMediaLayout` 接口

该布局包含的元素有：回放播放器、亮度手势布局、音量手势布局、快进/退手势布局，回放控制栏，等在回放播放区域出现的UI。

下面会列举介绍该布局中涉及到的主要方法。

4.2.1 初始化view方法

`PLVLCPlaybackMediaLayout` 继承于 `FrameLayout` ，在构造器中对使用 `initView` 方法对 `view` 进行了初始化处理。

```
public PLVLCPlaybackMediaLayout(@NonNull Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
    super(context, attrs, defStyleAttr);
    initView();
}

private void initView() {
    //填充播放器布局到该view中
    LayoutInflater.from(getContext()).inflate(R.layout.plvlc_playback_player_layout, this, true);
    //一系列findViewById，这里不详细列出
    //初始化播放器布局中包含的view
    initVideoView();
    initMediaController();
    initLoadingView();
    initSwitchView();
    initLayoutWH();
}
```

4.2.2 初始化数据方法

`PLVLCPlaybackMediaLayout` 的 `init` 方法是对外API，需要外部传入 `IPLVLiveRoomDataManager` 后进行初始化。

```
//播放器presenter
private IPLVPlaybackPlayerContract.IPlaybackPlayerPresenter playbackPlayerPresenter;

@Override
public void init(IPLVLiveRoomDataManager liveRoomDataManager) {
    this.liveRoomDataManager = liveRoomDataManager;
    //创建播放器mvp-presenter
    playbackPlayerPresenter = new PLVPlaybackPlayerPresenter(liveRoomDataManager);
    //注册播放器mvp-view
    playbackPlayerPresenter.registerView(playbackPlayerView);
    //初始化播放器mvp-presneter
    playbackPlayerPresenter.init();
    //传递presenter给mediaController调用
    mediaController.setPlaybackPlayerPresenter(playbackPlayerPresenter);
}

//创建播放器mvp-view
private IPLVPlaybackPlayerContract.IPlaybackPlayerView playbackPlayerView = new PLVAbsPlaybackPlayerView() {
    //...
}
```

4.2.3 设置PPTView方法

`PLVLCPlaybackMediaLayout` 的 `setPPTView` 方法是对外API，云课堂场景下，回放播放器需要关联PPTView，因此需要外部传入 `IPolyvPPTView` 进行设置。

```
@Override
public void setPPTView(IPolyvPPTView pptView) {
    playbackPlayerPresenter.bindPPTView(pptView);
}
```

4.2.4 开始播放方法

`PLVLCPlaybackMediaLayout` 的 `startPlay` 方法是对外API，该方法内部会调用 `PLVPlaybackPlayerPresenter` 的 `start` 方法来进行视频的加载及播放。

```
@Override
public void startPlay() {
    playbackPlayerPresenter.startPlay();
}
```

`PLVLCPlaybackMediaLayout` 内部持有 `PLVPlaybackPlayerPresenter`，因此除了内部可以自由调用播放器相关的方法外，也能将其封装成接口提供给外层调用， `startPlay` 方法就是其中一个例子。

5 子目录介绍

这里的子目录指的云课堂场景播放器模块下的子目录，对应项目中 `polyvLiveCloudClassScene` 模块里包名为 `com.easfun.polyv.livecloudcalss.modules.nedia` 包下的子包。

5.1 controller目录

```
//直播播放器控制栏接口
IPLVLCLiveMediaController
//回放播放器控制栏接口
IPLVLCPlaybackMediaController
//直播播放器控制栏接口实现类
PLVLCLiveMediaController
//回放播放器控制栏接口实现类
PLVLCPlaybackMediaController
```

该目录主要是放置直播及回放控制栏相关的类。

5.2 danmu目录

```
//弹幕控制器接口
IPLVLCDanmuController
//横屏信息发送输入框布局接口
IPLVLCLandscapeMessageSender
//弹幕控制器接口实现类
PLVLCDanmuFragment
//横屏信息发送输入框布局接口实现类
PLVLCLandscapeMessageSendPanel
```

该目录主要是放置弹幕相关的类。

5.3 widget目录

```
//亮度手势提示view
PLVLCLightTipsView
//音频模式显示的view
PLVLCLiveAudioModeView
//直播"更多"布局的view
PLVLCLiveMoreLayout
//占位图view
PLVLCPlaceHolderView
//回放"更多"布局的view
PLVLCPlaybackMoreLayout
//快进/退手势提示view
PLVLCProgressTipsView
//缓冲中显示的view
PLVLCVideoLoadingLayout
//音量手势提示view
PLVLCVolumeTipsView
```

该目录主要是放置播放器布局中使用到的view元素。