现代操作系统应用开发实验报告

学号： 14307004　　**班级　　：** 15 软工 1 班

姓名： 蔡冠文　　**实验名称：** Homework8

一．参考资料

请在这里列出对本实验有帮助你所参考的资料或者网站。

https://docs.microsoft.com/zh-cn/windows/uwp/audio-video-camera/play-audio-and-video-with-mediaplayer

https://docs.microsoft.com/zh-cn/windows/uwp/controls-and-patterns/slider

http://blog.csdn.net/lindexi_gd/article/details/51093890

二．实验步骤

请在这里简要写下你的实验过程。

1. 构建播放器 UI。跟着 TA 的 PPT 慢慢敲，发现没有 LoadedBehavior 和 mouseDown 等属性，一气之下换了 MediaPlayerElement

2. 根据官方文档 ,在 CS 文件中创建 mediaPlayer ,对应的时间轴控制器 ,时间片 timeSpan，加载 Assets 里面的资源，实现简单的播放暂停功能。

3. 添加 volume 功能，这里遇到了一点问题，第四部分会描述。

4. 进度条的创建。一开始也是跟着官方文档走，一行行对过了没问题，但是就是跑不动。于是添加断点进入调试模式。具体第四部分描述。

5. 拖动进度条。按照 volume 类似的做法如法炮制。

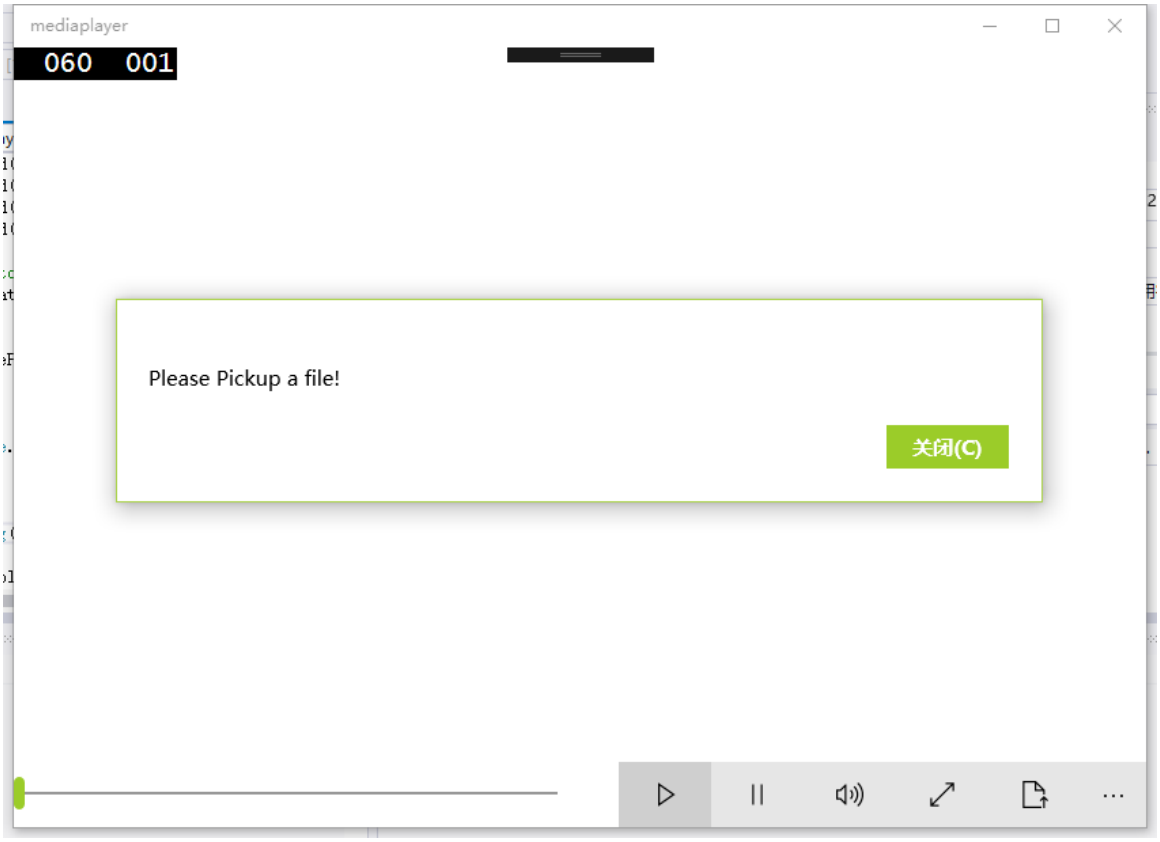
6. 实现全屏。这个问题耗时最长，第四部分慢慢讲。

7. 把全屏的问题弄好已经差不多凌晨两点了，加了一下读取本地文件的功能，基本上也是复制官方文档的代码，把变量名稍作修改

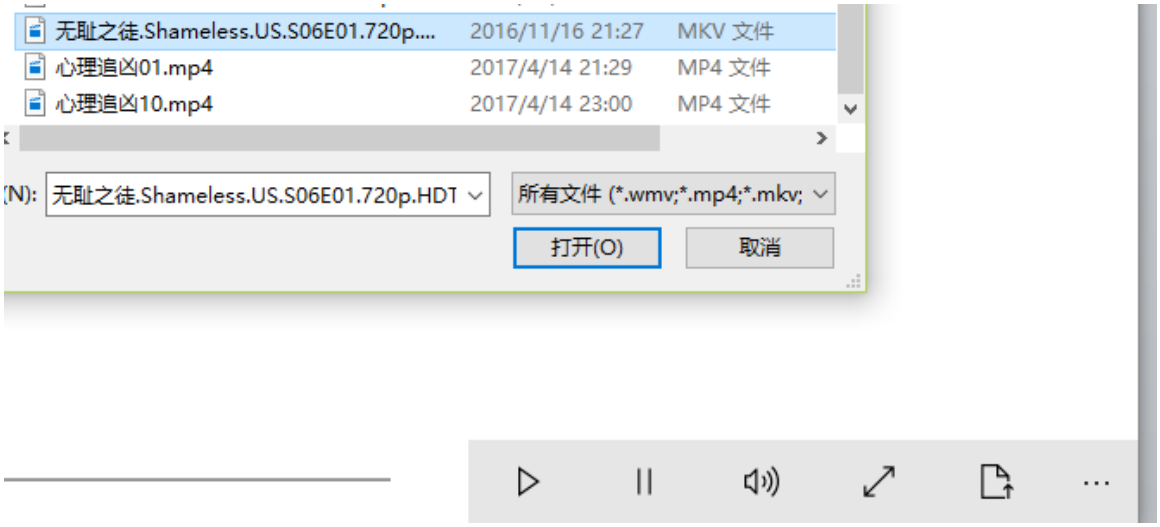
三．实验结果截图

请在这里把实验所得的运行结果截图。

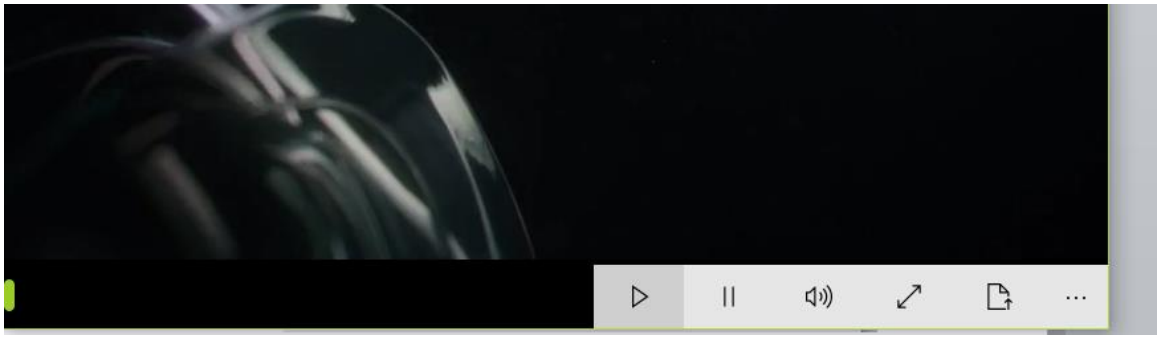
1. 打开播放器，此时还没加载文件，直接点播放或暂停按钮的话会抛出对话框



2. 选择视频文件



3. 点三角形开始播放



4. 开始播放，进度条在左边，菜单栏在右边

5. 拖动进度条

6. 全屏播放

7. 调节音量

四．实验过程遇到的问题

请在这里写下你在实验过程中遇到的问题以及解决方案。

1. 没有 LoadedBehavior 和 mouseDown

2. 点击 volume 没有弹出 flyout。

解决方案：添加 click 事件，调用 FlyoutBase.ShowAttachedFlyout(element);

3. volume 只有静音和原声的区别

解决方案： volume 是范围 0 到 1 的 double 值，而 slider 默认 0 到 100，赋值的时候除了 100。

4. timeline 进度条不动。

解决方案：添加断点发现 slider 的 value 是有变化的，只是变化比较小，在教程的基础上

换了一句就好。

```
if (duration != TimeSpan.Zero)
{
    await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal, () =>
    {
        //timeLine.Value = sender.Position.TotalSeconds / (float)duration.TotalSeconds;
        timeLine.Value = sender.Position.TotalSeconds;
    });
}
```

5. 不能拖动进度条

解决方案：参考 volume 的设置，获得 slider 的值后转化为 timeSpan。然后修改 position

```
//拖动进度条
private void timeLine_ValueChanged(object sender, RangeBaseValueChangedEventArgs e)
{
   Slider slider = sender as Slider;
    if (slider != null)
    {
        TimeSpan value = TimeSpan.FromSeconds(slider.Value);
        _mediaPlayer.TimelineController.Position = value;

    }

}
```

6. 全屏不能退出

解决方案：一开始用的 isFullWindow 属性，发现设置 visibility 等的 方法无济于事，感觉这个有点类似于窗口播放模式，其它的东西加不进去。后来百度搜 UMP 全屏，看到了 ApplicationView 这个东西，使用 Screen 模式。然而全屏了之后出现了新的问题

来自博客：

```
ApplicationView view = ApplicationView.GetForCurrentView();

bool isInFullScreenMode = view.IsFullScreenMode;

if (isInFullScreenMode)
{
    view.ExitFullScreenMode();
}
else
{
    view.TryEnterFullScreenMode();
}
```

在我们应用变为全屏，textblock就会 `In full screen mode`

我们可以设置 `PreferredLaunchWindowingMode` ，在我们应用打开

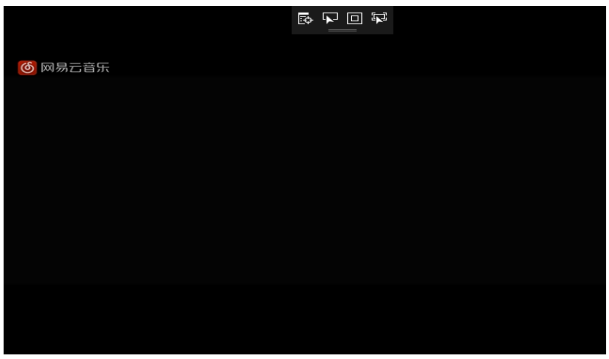
```
ApplicationView.PreferredLaunchWindowingMode = ApplicationViewWindow
```

收藏到代码笔记

ApplicationViewWindowingMode可以 `Auto` ，`PreferredLaunchViewSize` 设置窗口

和 `ApplicationView.PreferredLaunchViewSize` ，如果没有设置 `ApplicationView.PreferredLaunchViewSize` 会使用上次关闭窗口， `FullScreen`

7. 如图



可以说是一脸懵逼了，怎么视频文件的大小没有跟着全屏呢？花了好长的时间，才发现是布局的问题，一开始贪图方便，抄了 TA 的 UI 设计，其中用到了 StackPanel，后来改成 Grid

之后，跑的比西方记者还快~

五．思考与总结

请在这里写下你本次试验的心得体会以及所思所想。

这次实验学习了使用 windows 自带的插件构建一个简单的多媒体播放器，从时间的花费上来看，耗费时间最长的是全屏这个功能的实现。由于之前都是在 TODO 的基础上写，所以对 XML 的布局不太熟悉 ,以至于用了 StackPanel ,然后找了半天问题 ,最后改了 grid。

到目前为止的 UWP 编程的过程都是大同小异，查官方文档看补全。下面总结一下这次作业功能的实现方法：

1. **MediaPlayer** 的每个实例均进行实例化，并且 **Source** 设置为媒体文件。 接下来，创建新 **MediaTimelineController**。 对于每个 **MediaPlayer**，将 IsEnabled 属性设为 False 禁用与每台播放器关联的 MediaPlaybackCommandManager。 然后，TimelineController 属性设置为时间线控制器对象。

实例化：

```
mediaSource.OpenOperationCompleted += MediaSource_OpenOperationCompleted;
_mediaPlayer.Source = mediaSource;

_mediaPlayerElement.SetMediaPlayer(_mediaPlayer);


_mediaTimelineController = new MediaTimelineController();
```

将 **MediaTimelineController** 附加到一个或多个媒体播放器后，可以通过使用控制器公开的方法控制播放状态。

播放和暂停：

```
//播放play
private void play_Click(object sender, RoutedEventArgs e)
{
    if (_mediaTimelineController == null)
    {
        var K = new MessageDialog("Please Pickup a file!").ShowAsync();
    }
    else
    {
        _mediaTimelineController.Start();
    }
}
暂停恢复
private void resume_Click(object sender, RoutedEventArgs e)
{
    if (_mediaTimelineController == null)
    {
        var K = new MessageDialog("Please Pickup a file!").ShowAsync();
    }
    else
    {
        if (_mediaTimelineController.State == MediaTimelineControllerState.Running)
        {
            _mediaTimelineController.Pause();
            resume.Content = "Resume";
        }
        else
        {
```

为时间线控制器的 **PositionChanged** 事件注册处理程序。注意到时间轴的 Value 直接是总秒数，而不用除以时间片的长度。

```
//TimelineController 属性设置为时间线控制器对象。
MediaTimelineController _mediaTimelineController;
```

```
//在 PositionChanged 的处理程序中，更新滑块值以反映时间线控制器的当前位置。
private async void _mediaTimelineController_PositionChanged(MediaTimelineController sender, object args)
{
    if (duration != TimeSpan.Zero)
    {
        await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal, () =>
        {
            //timeLine.Value = sender.Position.TotalSeconds / (float)duration.TotalSeconds;
            timeLine.Value = sender.Position.TotalSeconds;
        });
    }
}
```

拖动进度条：

```
//拖动进度条
private void timeLine_ValueChanged(object sender, RangeBaseValueChangedEventArgs e)
{
   Slider slider = sender as Slider;
    if (slider != null)
    {
        TimeSpan value = TimeSpan.FromSeconds(slider.Value);
        _mediaPlayer.TimelineController.Position = value;

    }

}
```

全屏效果：

```
//全屏效果
private void fullScreen_Click(object sender, RoutedEventArgs e)
{

    var view = ApplicationView.GetForCurrentView();
    //_mediaPlayerElement.IsFullWindow = !_mediaPlayerElement.IsFullWindow;
    if (view.IsFullScreenMode)
    {

        view.ExitFullScreenMode();
        command.Visibility = Visibility.Visible;
        ApplicationView.PreferredLaunchWindowingMode = ApplicationViewWindowingMode.Auto;

    }
    else
    {
        if (view.TryEnterFullScreenMode())
        {
            command.Visibility = Visibility.Visible;
            ApplicationView.PreferredLaunchWindowingMode =
                ApplicationViewWindowingMode.FullScreen;

        }
    }

}
```