现代操作系统应用开发实验报告

学号： 15331046　　**班级　　：** 软工一班

姓名： 陈志扬　　**实验名称：** HW-8

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。

资料：老师的 week8 课件、week2 课件与动画设计相关的 xaml 高级教程.pdf，TA 师兄的 demo 和实验作业要求、相关博客等等

网站：

动画效果的实现参考的网站

https://msdn.microsoft.com/zh-cn/magazine/windows.ui.xaml.uielement.rendertransformorigin.aspx

全屏状态下隐藏底部 CommandBar 和退出全屏状态下显示底部 CommandBar 的关键代码参考：

https://github.com/oneonces/Windows-universal-samples/blob/master/Samples/TitleBar/cs/Scenario2_Extend.xaml.cs

关于音频视频的微软官方文档

https://docs.microsoft.com/zh-cn/windows/uwp/audio-video-camera/media-playback

问答网站：http://stackoverflow.com/　https://www.baidu.com/

等等

二．实验步骤

请在这里简要写下你的实验过程。

关键代码在 第四部分遇到的问题 中体现。

① 先简单了解设计需求，学习相关资料文档。

② 设计界面，实现基本的 UI。主要在 Page 底部添加 AppBarButton 控件，加上 demo 上的参考按钮。

③ 在 MainPage.xaml.cs 文件中对底部 AppBarButton 各个按钮的 Click 事件中的方法实现，可以实现媒体文件的播放、暂停、停止等基本功能。

④ 实现可以选择媒体文件播放的功能，包括音频视频文件。主要调用 media.Play()、media.Pause()、media.Stop()。

⑤ 实现“乞丐版”的全屏功能，因为一开始无法隐藏底部 CommandBar，所以称之为“乞丐版”。点击全屏按钮，进入全屏状态。这里还无法对隐藏 CommandBar 做出可行的操作。

⑥ 实现图片动画效果，这里指唱片旋转而已。主要用到 Ellipse 使图片呈圆显示，RenderTransform、CompositeTransform，动画 Storyboard。

⑦ 实现两个 Slider 的数据绑定。这里说“两个”是因为还没把音量键 Volume 放到 CommandBar 中。整个过程涉及到 MediaOpened、MediaEnded 等，在进度条末尾添加了一个 TextBlock，以时：分：秒的格式显示当前播放的媒体文件的

总时间。

⑧ 实现类网易云音乐的效果：添加了一个音乐播放器臂，并为之添加效果

⑨ 将音量键放到底部 CommandBar，Slider 改成 Flyout 显示。此时对音量的绑定要换到 media 中去，相应的值转换也要改变

⑩ 实现全屏状态下隐藏底部 CommandBar，退出全屏显示底部 CommandBar，顺便绑定进度条和时间长度的 Opacity，同时显示同时隐藏。

⑪ 实现全屏状态下移动鼠标到底部 CommandBar 时，CommandBar、进度条和时间长度显示出来，移动鼠标离开底部 CommandBar 时，CommandBar、进度条和时间长度都隐藏起来。

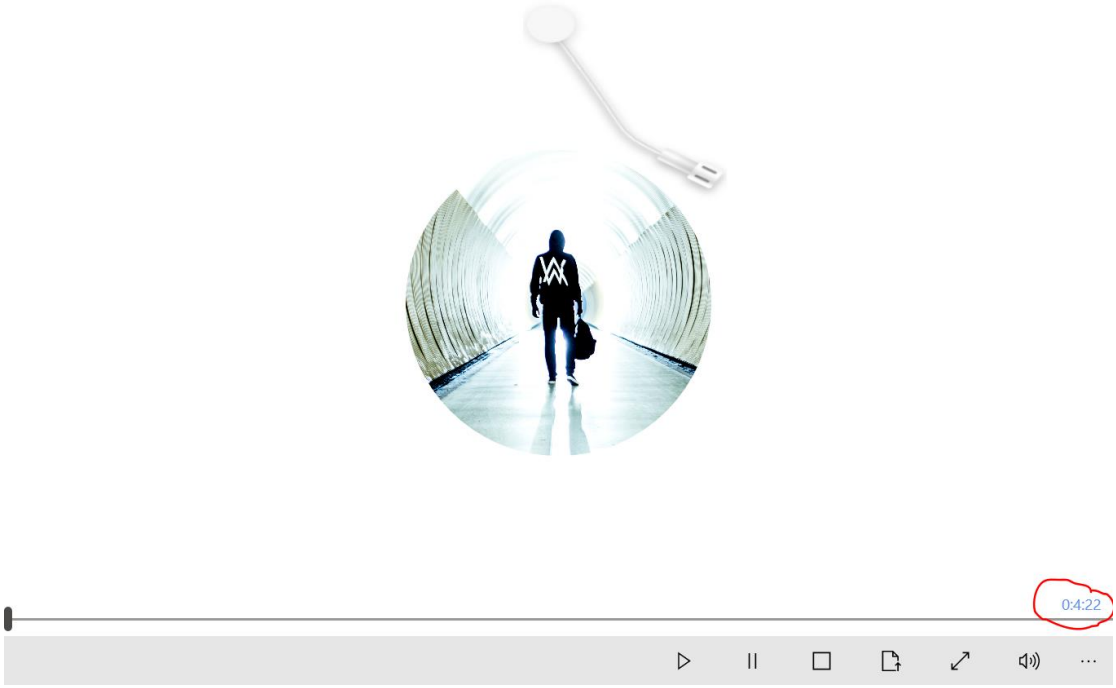
⑫ 调界面 UI 让它好看一点点。不得说这是花时间的一点。

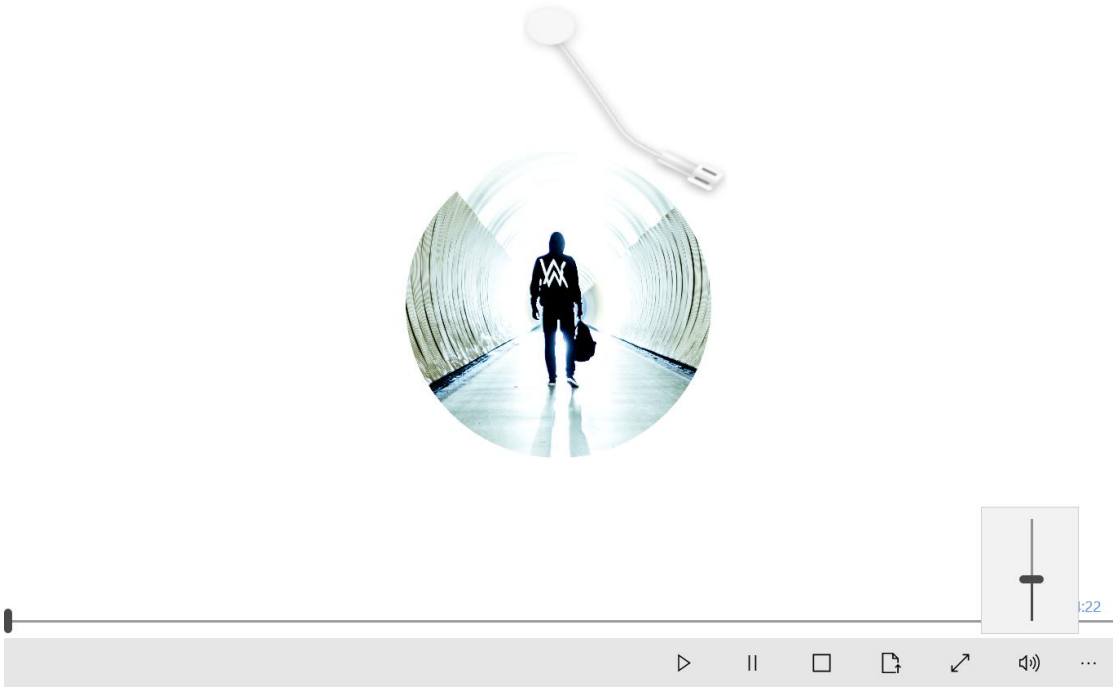
⑬ 最后，发现 bug 修复 bug。主要是动画设计的 bug，在第四部分中详述。

三．实验结果截图

请在这里把实验所得的运行结果截图。

程序起始界面：图中画圆圈的数字表示当前媒体文件的总时间长度，音量键放在底部 CommandBar 中。

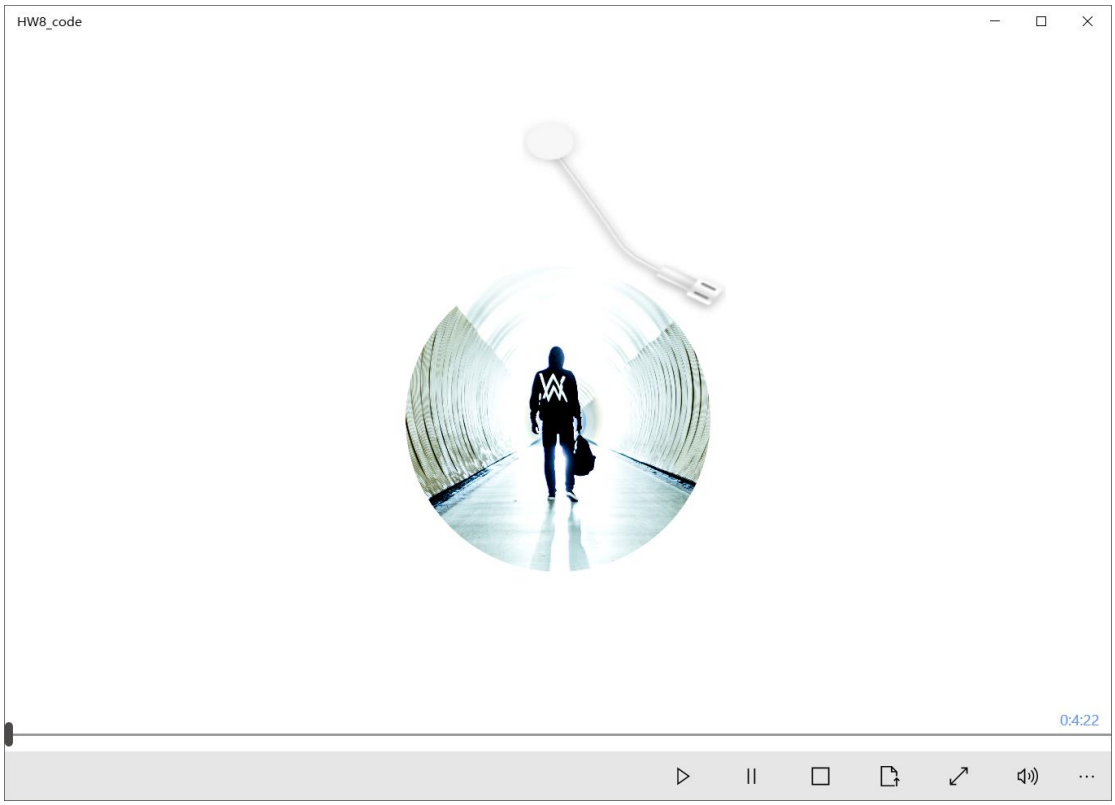




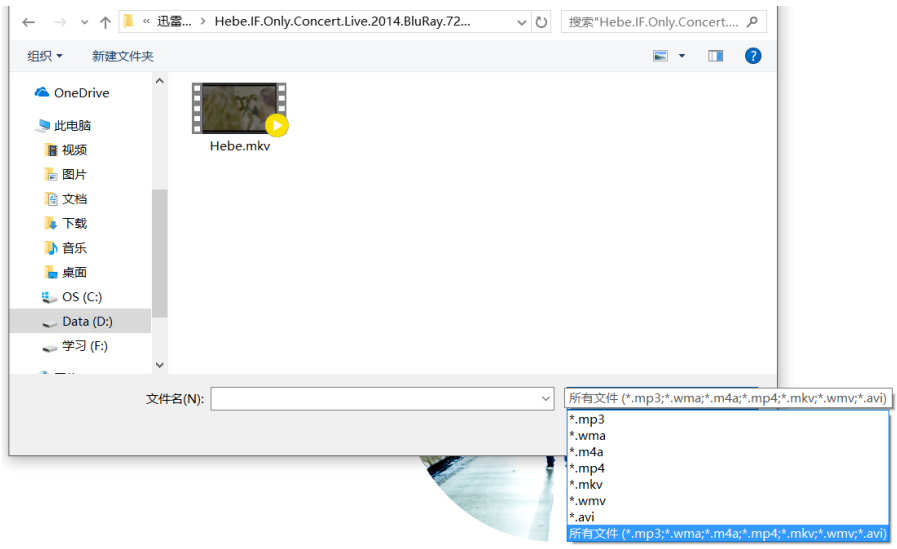
点击播放按钮：可以看到音乐播放器臂放到唱片图片上，唱片图片开始旋转，进度条也跟着走。

点击暂停按钮，可以看到音乐播放器臂离开唱片图片，音乐也随着停止

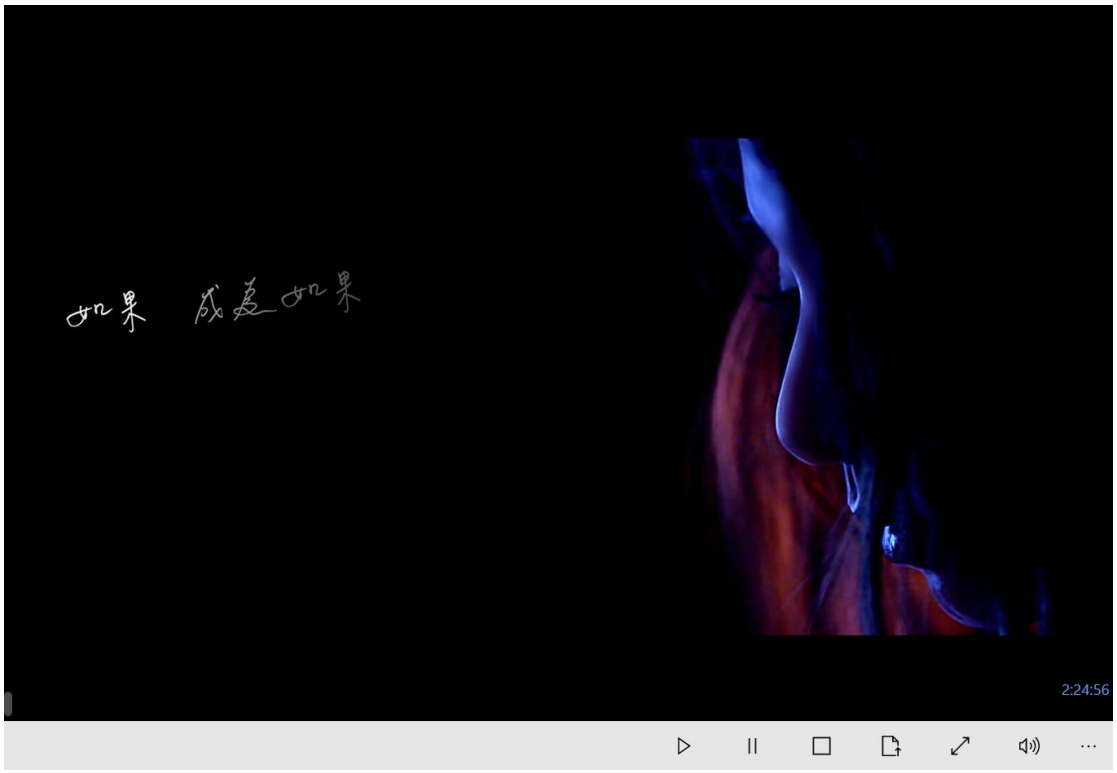
点击停止按钮，音乐播放停止，动画恢复到初始状态



点击选择文件按钮 ,可以选择音频、视频文件 ,具体支持 mp3、m4a、wma、mp4、mkv、wmv、avi 格式的文件



选择该视频文件进行播放：自动播放，注意到左下角有该视频的总时间长度。



点击暂停按钮可实现暂停：

点击停止按钮，可实现停止播放。这里截图略。

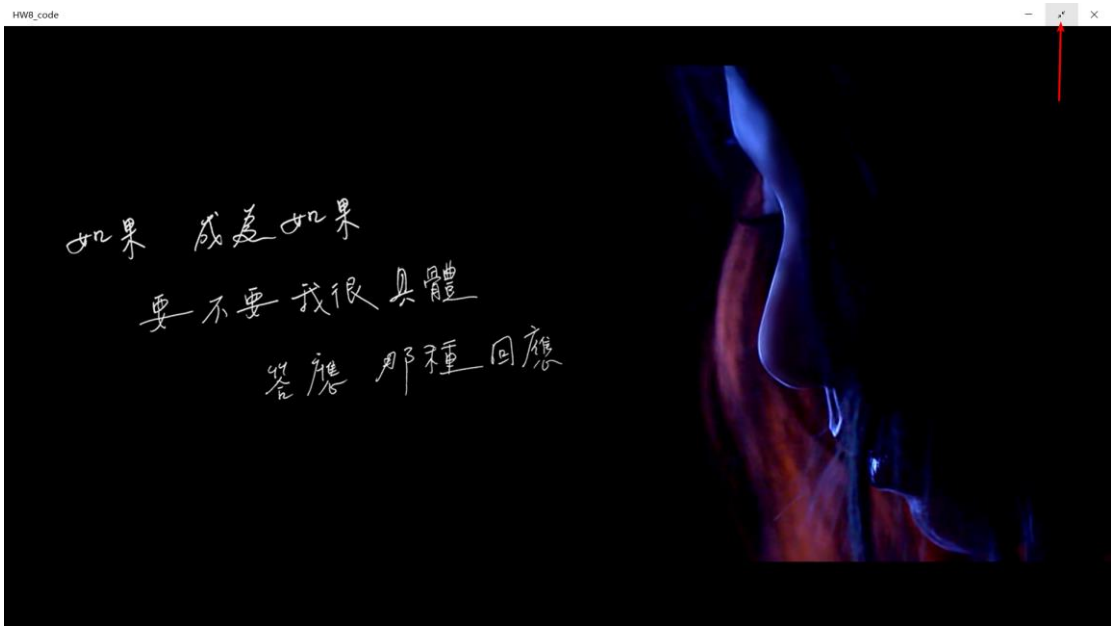
接下来比较重要的功能，全屏

点击全屏按钮，全屏显示，底部隐藏了 CommandBar、进度条和时间长度：

将鼠标移到底部可看到 CommandBar、进度条和时间长度显示出来：

将鼠标移出底部，CommandBar、进度条和时间长度均又隐藏起来：

点击退出全屏按钮或者右上角第二个按钮可退出全屏：



退出全屏后底部 CommandBar 再一次显示出来了：

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 进度条和音量条的位置怎么放的问题。这里经群里大佬指点，把音量键放到底部 CommandBar 里，但是由于前面我是放在

grid 里的，已经实现了 Slider 的 Value 值绑定到 media 的 volume，放到 AppBarButton 里 Slider 的绑定无效，要在 MediaElement 中的属性 Volume 绑定 Slider 的 Value ,注意值转换此时也要更改（原本乘以 100 要改成除以 100）;

```
<!--音量键的位置-->
<AppBarButton x:Name="volumn" Icon="Volume" Label="音量" Click="showVolumnClicked">
    <FlyoutBase.AttachedFlyout>
        <Flyout>
            <!--这里的value值转化为音量volume=0.4，是初始音量-->
            <Slider x:Name="volume" Orientation="Vertical" Height="100" Width="35" Value="40" Maximum="100" />
        </Flyout>
    </FlyoutBase.AttachedFlyout>
</AppBarButton>

<!--媒体文件 Volume绑定slider的value值-->
<MediaElement x:Name="media" Volume="{Binding ElementName=volume, Path=Value, Converter={StaticResource volumeConverter}, Mode=TwoWay}"
              Source="Assets\Fade.mp3" AutoPlay="False" MediaOpened="mediaOpened" MediaEnded="mediaEnded" MediaFailed="mediaFailed"/>
```

```
// 点击Volumn按钮，显示Slider
private void showVolumnClicked(object sender, RoutedEventArgs e)
{
    FrameworkElement element = sender as FrameworkElement;
    if (element != null) FlyoutBase.ShowAttachedFlyout(element);
}
class volumeConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, string language)
    {
        // 注意这里是value转volume，是除以100
        return (double)value / 100;
    }

    public object ConvertBack(object value, Type targetType, object parameter, string language)
    {
        return (double)value * 100;
    }
}
```

进度条的转换：

```
<Slider x:Name="speed" Value="{Binding ElementName=media, Path=Position, Converter={StaticResource speedConverter}, Mode=TwoWay}"
      VerticalAlignment="Bottom"/>
<TextBlock x:Name="mediaTimeSpan" Foreground="CornflowerBlue" HorizontalAlignment="Right" VerticalAlignment="Bottom"
         Width="50" Margin="0,20,0,0"/>
```

```
class speedConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, string language)
    {
        return ((TimeSpan)value).TotalSeconds;
    }

    public object ConvertBack(object value, Type targetType, object parameter, string language)
    {
        return TimeSpan.FromSeconds((double)value);
    }
}
```

② 全屏时底部 CommandBar 隐藏后退出全屏无法再次显示。一开始是点击全屏按钮判断是否处于全屏状态，若非，直接进入全屏并隐藏 CommandBar(Visibility 设为 Collapsed)；但是这样点击右上角第二个按钮退出全屏后 CommandBar 彻底不见了。原因就在于判断全屏的过程写在 fullscreenClicked 函数里，点击进入全屏后再退出全屏（此时不是点击全屏按钮退出的，无法更改 CommandBar 的 Visibility）。解决这个问题，我们用到了 OnNavigatedTo，当前窗口大小发生改变时注册一个事件，让它来执行全屏的判断，若全屏，隐藏；否则显示出来。这是第一个版本。

接下来是第二个版本，利用鼠标来处理 CommandBar 的可见性问题，主要用到 PointerEntered 和 PointerExited，鼠标移到底部 CommandBar 显示出来，移出底部 CommandBar 隐藏起来。这里遇到的问题是不能设置 Visibility，要设置 Opacity 来实现可见性，而且不要把 CommandBar 放到 Page.ButtonAppBar 中，那样的话一旦隐藏掉之后就不会再显示出来了，直接把 CommandBar 放到 Grid 的 Buttom 中

即可。

最后我又把进度条和时间长度的 Opacity 绑定到 CommandBar 的 Opacity，实现三者的同时显示和隐藏，比较符合常规播放器。

把进度条和 CommandBar 用一个 Grid 拼合在一起，避免鼠标移到两者之间的空隙中造成可见与否的“鬼畜”。

具体的 xaml 如下：

```
<!--时间长度-->
<TextBlock x:Name="mediaTimeSpan" Opacity="{Binding ElementName=commandBar, Path=Opacity, Mode=OneWay}"
          Foreground="CornflowerBlue" HorizontalAlignment="Right" VerticalAlignment="Bottom" Width="50" Margin="0,0,0,70"/>
```

```
<Grid PointerEntered="commandBar_PointerEntered" PointerExited="commandBar_PointerExited">
    <Slider x:Name="speed" Opacity="{Binding ElementName=commandBar, Path=Opacity, Mode=OneWay}"
            Value="{Binding ElementName=media, Path=Position, Converter={StaticResource speedConverter}, Mode=TwoWay}"
            VerticalAlignment="Bottom" Margin="0,0,0,40"/>

    <CommandBar x:Name="commandBar" VerticalAlignment="Bottom">
        <AppBarButton x:Name="play" Icon="Play" Label="播放" Click="playClicked"/>
        <AppBarButton x:Name="pause" Icon="Pause" Label="暂停" Click="pauseClicked"/>
        <AppBarButton x:Name="stop" Icon="Stop" Label="停止" Click="stopClicked"/>
        <AppBarButton x:Name="selectfile" Icon="OpenFile" Label="选择文件" Click="selectFileClicked"/>
        <AppBarButton x:Name="fullscreen" Ico                    l="全屏" Click="fullScreenClicked"/>
        <!--音量键的位置-->
        <AppBarButton x:Name="volumn" Icon="Volume" Label="音量" Click="showVolumnClicked">
            <FlyoutBase.AttachedFlyout>
                <Flyout>
                    <!--这里的value值转化为音量volume=0.4，是初始音量-->
                    <Slider x:Name="volume" Orientation="Vertical" Height="100" Width="35" Value="40" Maximum="100" />
                </Flyout>
            </FlyoutBase.AttachedFlyout>
        </AppBarButton>
    </CommandBar>
</Grid>
```

AppBarButton.Icon

具体的 C#代码如下：

```
// 处理鼠标移到底部时的可见性
private void commandBar_PointerEntered(object sender, PointerRoutedEventArgs e)
{
    // 全屏状态下移动鼠标到底部CommandBar可显示出来，
    // 同时进度条和时间长度也显示出来（绑定）
    commandBar.Opacity = 100;

}
// 处理鼠标移出底部时的可见性
private void commandBar_PointerExited(object sender, PointerRoutedEventArgs e)
{
    // 获取应用程序是否在全屏下运行
    ApplicationView view = ApplicationView.GetForCurrentView();
    bool isfull = view.IsFullScreenMode;
    // 全屏状态下移动鼠标离开CommandBar可隐藏出来
    // 同时进度条和时间长度也隐藏起来（绑定）
    if (isfull)
    {
        commandBar.Opacity = 0;
        fullscreen.Label = "退出全屏";
    }
}
```

```
// 当前窗口大小发生变化注册一个事件，主要来处理退出全屏底部CommandBar的可见性问题
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    Window.Current.SizeChanged += windowSizeChanged;
}
private void windowSizeChanged(object sender, WindowSizeChangedEventArgs e)
{
    // 获取应用程序是否在全屏下运行
    ApplicationView view = ApplicationView.GetForCurrentView();
    bool isfull = view.IsFullScreenMode;
    if (!isfull)
    {
        commandBar.Opacity = 100;// 否，显示底部CommandBar,同时显示进度条和时间长度（绑定）
        fullscreen.Label = "全屏";
    }

}
// 点击进入全屏模式
private void fullScreenClicked(object sender, RoutedEventArgs e)
{

    // 获取应用程序是否在全屏下运行
    ApplicationView view = ApplicationView.GetForCurrentView();
    bool isfull = view.IsFullScreenMode;
    if (isfull)
    {
        ApplicationView.GetForCurrentView().ExitFullScreenMode();// 进入全屏
    }
    else
    {
        ApplicationView.GetForCurrentView().TryEnterFullScreenMode();// 进入全屏
    }
}
```

③ 动画设置的问题，最主要的就是当陆续点击暂停播放时，动画会越来越慢，经过仔细分析，发现当每一次点击播放，我都采用了 storyboard.Begin()，这就造成了转到 360°的所需角度变小了，而 Begin()又是从头开始，转一圈时间不变，所以只能减慢速度来实现到达 360°时刚刚好用完了设置的时间。解决的方法是加入一个判断音频文件是否暂停的 bool 值 ispause，当为真时，动画恢复 storyboard.Resume()，可解决该问题。

但是很快又出现了新问题，如果音乐播放着而用户先点击停止再点击暂停，再点击播放，这时候由于 Resume 之前的状态是

stop，所以动画不会继续旋转，所以加上一个 isstop 来判断；但是又来了个问题，如果程序刚刚启动还没播放音乐用户先点击暂停或者停止，此时的判断又出现了 bug，所以还要加上 isplay 表示音频文件是否在播放中。（心好累）

具体代码见图：

```
public bool ismusic = true;// 表示是音频文件
public bool isplay = false;// 表示音频文件是否播放
public bool ispause = false;// 表示音频文件是否暂停
public bool isstop = false;// 表示音频文件是否停止
```

```
private void playClicked(object sender, RoutedEventArgs e)
{
    media.Play();
    isplay = true;
    // 音频文件启动动画效果
    if (ismusic)
    {
        // 音频文件的状态是从暂停转到播放的，恢复图片动画
        if (ispause)
        {
            // 处理：点击停止再点击暂停再点击播放动画不选转问题
            // 因为停止时Resume还是停在那里不动的
            if (isstop)
            {
                storyboard.Begin();
                isstop = false;
            }
            else
                storyboard.Resume();
        }
        // 是刚播放的，启动动画效果
        else
        {
            storyboard.Begin();
        }
        storyboard1.Begin();
    }
    else
    {
        storyboard.Stop();
        storyboard1.Stop();
    }
}
```

```
// 点击暂停，暂停播放，暂停动画
private void pauseClicked(object sender, RoutedEventArgs e)
{
    media.Pause();
    // 处理：刚运行时不点播放然而先点暂停按钮的bug
    if (isplay)
    {
        if (ismusic)
        {
            ispause = true;
            storyboard.Pause();
            storyboard1.Stop();
        }
    }

}
// 点击停止，停止播放，停止动画
private void stopClicked(object sender, RoutedEventArgs e)
{
    media.Stop();
    // 处理：刚运行时不点播放然而先点停止按钮的bug
    if (isplay)
    {
        if (ismusic)
        {
            isstop = true;
            storyboard.Stop();
            storyboard1.Stop();
        }
    }
}
```

④接下来贴上动画设计的 xaml 代码。主要的问题是 RepeatBehavior 要设成 Forever，循环播放时动画才会继续旋转；

```
<!--唱片动画旋转-->
<Ellipse x:Name="ellipse" RenderTransformOrigin="0.5,0.5" Width="300" Height="300">
    <Ellipse.Fill>
        <ImageBrush ImageSource="Assets\Alan-Walker.jpg"/>
    </Ellipse.Fill>
    <Ellipse.RenderTransform>
        <CompositeTransform />
    </Ellipse.RenderTransform>
    <Ellipse.Resources>
        <Storyboard x:Name="storyboard" RepeatBehavior="Forever">
            <DoubleAnimation From="0" To="360" Duration="0:0:20" Storyboard.TargetName="ellipse"
                             Storyboard.TargetProperty="(UIElement.RenderTransform).(CompositeTransform.Rotation)" />
        </Storyboard>
    </Ellipse.Resources>
</Ellipse>

<!--类网易云音乐动画旋转-->
<Rectangle x:Name="rectangle" RenderTransformOrigin="0,0" Width="200" Height="200" Margin="130,-400,0,0">
    <Rectangle.Fill>
        <ImageBrush ImageSource="Assets\hhh.png"/>
    </Rectangle.Fill>
    <Rectangle.RenderTransform>
        <CompositeTransform />
    </Rectangle.RenderTransform>
    <Rectangle.Resources>
        <Storyboard x:Name="storyboard1">
            <DoubleAnimation To="5" Duration="0:0:0.2" Storyboard.TargetName="rectangle"
                             Storyboard.TargetProperty="(UIElement.RenderTransform).(CompositeTransform.Rotation)" />
        </Storyboard>
    </Rectangle.Resources>
</Rectangle>
```

⑤ MediaOpened 获取媒体文件的时间并转换成时：分：秒的格式显示出来的代码如下：

```
<Slider x:Name="speed" Value="{Binding ElementName=media, Path=Position, Converter={StaticResource speedConverter}, Mode=TwoWay}"
        VerticalAlignment="Bottom"/>
<TextBlock x:Name="mediaTimeSpan" Foreground="CornflowerBlue" HorizontalAlignment="Right" VerticalAlignment="Bottom"
           Width="50" Margin="0,20,0,0"/>
```

```
private void mediaOpened(object sender, RoutedEventArgs e)
{
    // 获取当前打开的媒体文件的持续时间（以总秒数来算）
    speed.Maximum = media.NaturalDuration.TimeSpan.TotalSeconds;
    // 以 时:分:秒 的格式显示媒体文件的总时间
    mediaTimeSpan.Text = ((int)speed.Maximum / 3600).ToString() + ":"
        + ((int)speed.Maximum % 3600 / 60).ToString() + ":"
        + (speed.Maximum % 60).ToString().Substring(0,2);

    speed.Value = 0;
}
```

⑥ 选取本地文件进行播放的代码：

```
// 选取本地文件播放
private async void selectFileClicked(object sender, RoutedEventArgs e)
{
    var picker = new FileOpenPicker();
    picker.ViewMode = PickerViewMode.Thumbnail;
    picker.SuggestedStartLocation = PickerLocationId.VideosLibrary;
    // 可选的文件格式
    picker.FileTypeFilter.Add(".mp3");
    picker.FileTypeFilter.Add(".wma");
    picker.FileTypeFilter.Add(".m4a");
    picker.FileTypeFilter.Add(".mp4");
    picker.FileTypeFilter.Add(".mkv");
    picker.FileTypeFilter.Add(".wmv");
    picker.FileTypeFilter.Add(".avi");

    StorageFile file = await picker.PickSingleFileAsync();
    if (file != null)
    {
        IRandomAccessStream fileStream = await file.OpenAsync(FileAccessMode.Read);
        // 自动播放
        media.SetSource(fileStream, file.FileType);
        media.AutoPlay = true;
        // 判断选择的文件是否是音频文件
        // 是，启动动画效果
        if (isMusic(file.FileType))
        {
            ismusic = true;
            isplay = true;
            storyboard.Begin();
            storyboard1.Begin();
        }
        else
        {
            // 从音频文件切换到视频文件也要停止动画
            ismusic = false;
            storyboard.Stop();
            storyboard1.Stop();
        }
    }
}
```

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。

这次实验做的很开心，但同时心好累。主要是动画设计 Begin()、Stop()、Resume()这一块要搞清楚各个按钮之间的逻辑，每一个点击需要为它们加上 bool 值判断当前处于哪种状态，以免出现可避免的 bug。

开心的是做出了一个貌似功能还行的播放器，实现了全屏、音量、进度条、选择媒体文件、播放、暂停、停止等功能，而且还做出了类网易云音乐的播放效果，还实现了全屏下移动鼠标显示相关控件的功能，这几点令我非常开心。

这次是 uwp 的最后一个作业了，选了这门课，确确实实学到了一些东西，开心至极。

接下来还有期中项目，希望能有个好结果！

1. 实验报告提交格式为 pdf。
2. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。