现代操作系统应用开发实验报告

姓名：　谷雨

学号：　16341005

实验名称：lab3 网络访问&音乐播放器

一、参考资料

第七周：

1. ppt
2. Weather

https://github.com/cnxy/Weather

借用天气接口

第八周：

1. 《使用 MediaPlayer 播放音频和视频》
https://docs.microsoft.com/zh-cn/windows/uwp/audio-video-camera/play-audio-and-video-with-mediaplayer
2. 《C#学习之路之使用 windows media player 实例》
3. https://blog.csdn.net/hk_5788/article/details/44710769
4. 《Canvas 旋转动画(rotate())》
https://blog.csdn.net/houyanhua1/article/details/79949259
5. 《UWP：使用 MediaPlayerElement 实现媒体播放器》
https://blog.csdn.net/linwh8/article/details/70314698

二、实验步骤

第七周：

1. 查找各种网络 API
2. 实现网络访问，网络 API 接口调用

第八周：

1. 实现播放器基本功能。

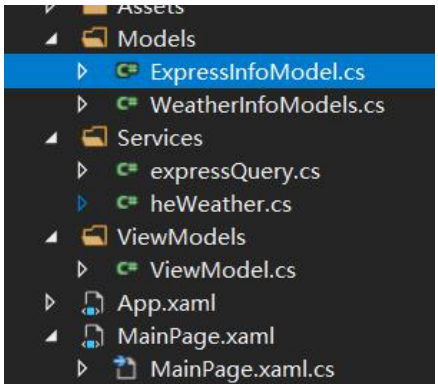
2. 实现 storyboard 动画。

3.修改 BUG 逻辑。

三、关键步骤截图

第七周：

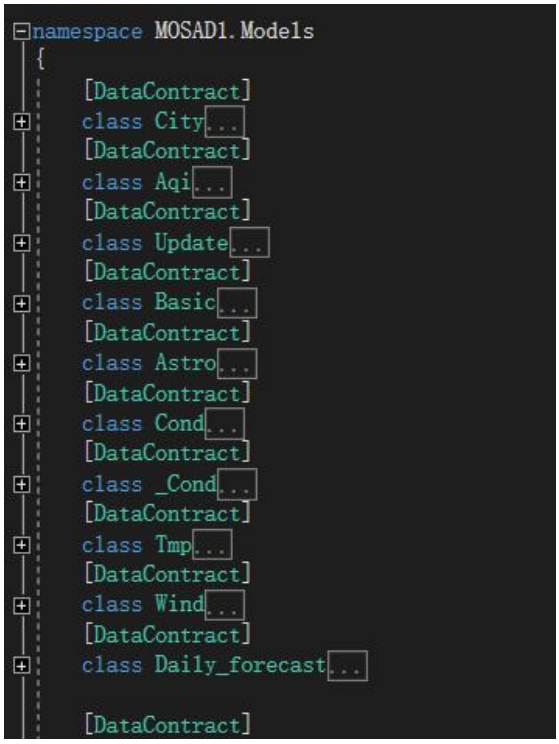
项目文档结构



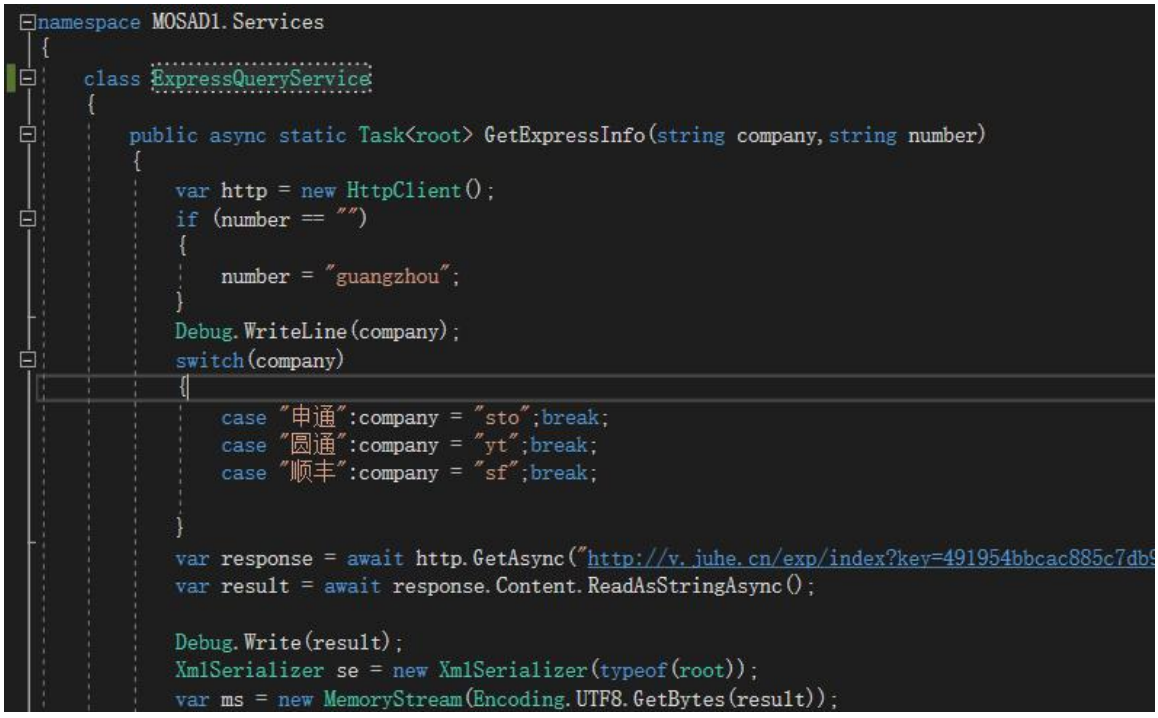
天气查询实现

```
class WeatherService
{
    public async static Task<RootObject> GetWeather(string cityName)
    {
        var http = new HttpClient();
        if (cityName == "")
        {
            cityName = "guangzhou";
        }
        var response = await http.GetAsync("https://free-api.heweather.com/v5/weather?city=" + cityName + "&key=0244c885f15d47da91eec9c7985c073a");
        var result = await response.Content.ReadAsStringAsync();
        if(result.Contains("unknown city"))
        {
            throw  new Exception("No Such City");
        }
        var serializer = new DataContractJsonSerializer(typeof(RootObject));
        Debug.Write(result);
        var ms = new MemoryStream(Encoding.UTF8.GetBytes(result));
        var data = (RootObject)serializer.ReadObject(ms);
        return data;
    }
}
```

天气查询数据模型



快递查询实现



快递查询数据模型

```
namespace MOSAD1.Models
{
    [XmlRoot("root")]
    public class root
    {
        [XmlElement("resultcode")]
        public string resultcode;
        [XmlElement("reason")]
        public string reason;
        [XmlElement("result")]
        public ExpressInfoModel result;
    }
    public class ExpressInfoModel
    {
        [XmlElement("company")]
        public string company;
        [XmlElement("no")]
        public string number;
        [XmlArray("list"), XmlArrayItem("item")]
        public List<ExpressItem> infos;
```

两个功能页面的 ViewModel

```
namespace MOSAD1.ViewModels
{
    class WeatherViewModel...
    class ExpressViewModel : INotifyPropertyChanged
    {
        public void NotifyPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
            }
        }
        public event PropertyChangedEventHandler PropertyChanged;
        private ExpressInfoModel _expressInfoModel;
        public ExpressInfoModel expressInfoModel { get { return this._expressInfoModel; } set { this._expressIn
        public void update(root root)
        {
            expressInfoModel= root.result;
```

按钮切换功能页面的实现，绑定

```
Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}" Width="500">
    <StackPanel>
        <StackPanel HorizontalAlignment="Center" Orientation="Horizontal">
            <RadioButton IsChecked="True" x:Name="WeatherButton">Weather</RadioButton>
            <RadioButton x:Name="ExpressButton">Express</RadioButton>
        </StackPanel>
        <StackPanel Visibility="{x:Bind WeatherButton.IsChecked ,Mode=OneWay}"...>
        <StackPanel Visibility="{x:Bind ExpressButton.IsChecked,Mode=OneWay}"...>
```

第八周:

基本功能暂停恢复停止的实现

```
private void Resume()...
private void Pause()...
private void Begin()...
private void Stop()
{
    StateChange(false);
    ViewModel.MediaTimelineController.Pause();
    ViewModel.MediaPlayer.Source=null;

    timeLine.Value = 0;
    timeLine.Minimum = 0;
    timeLine.Maximum = 0;
    RotateTransformStoryboard.Stop();
    ViewModel.BeStop();
    PauseButton.Label = "Play";
    Debug.WriteLine("???");
}
```

由于 mediaplayer 的基本状态表示方式无法满足对实现此次功能，所以我又添加了几个基本播放器状态的表示，它们与播放器按钮是否可选以及暂停按钮的状态绑定。

```
public bool Stop { get { return !stop; }set { stop = value;NotifyPropertyChanged("Stop"); } }
private bool pause = false;
public bool Pause { get { return !pause; } set { pause = value; NotifyPropertyChanged("Pause"); } }
private bool running = false;
public bool Running { get { return running; } set { running = value; NotifyPropertyChanged("Running"); } }
```

由于对于音乐和视频，某些功能的实现会有些不同，所以需要表示当前播放内容。
播放内容（音乐，视频的状态表示及切换）

```
private bool video = false;
private void StateChange(bool IfVideo)
{
    if (IfVideo)
    {
        video = true;
        border.CornerRadius = new CornerRadius(0);
        Show();
    }
    else
    {
        video = false;
        if (ElementCompositionPreview.GetElementChildVisual(_compositionCanvas) != null)
            ElementCompositionPreview.GetElementChildVisual(_compositionCanvas).Dispose();
        border.CornerRadius = new CornerRadius(1000);
    }
}
```

对于播放音乐时图片旋转，以及视频的呈现，我为了减少 XAML 文件中无用的控件于是将他们一起实现，通过 canvas，外面包一层 border（控制圆形），然而这样却增加了复杂度：

视频的呈现通过（使用 MediaPlayerSurface 将视频呈现到 Windows.UI.Composition 界面）：

```
private void Show()
{

    MediaPlayer _mediaPlayer = ViewModel.MediaPlayer;
    _mediaPlayer.SetSurfaceSize(new Size(_compositionCanvas.ActualWidth, _compositionCanvas.ActualHeight));

    var compositor = ElementCompositionPreview.GetElementVisual(this).Compositor;
    MediaPlayerSurface surface = _mediaPlayer.GetSurface(compositor);

    SpriteVisual sprit
    spriteVisual.Size

    CompositionBrush brush = compositor.CreateSurfaceBrush(surface.CompositionSurface);
    spriteVisual.Brush = brush;

    ContainerVisual container = compositor.CreateContainerVisual();
    container.Children.InsertAtTop(spriteVisual);
    ElementCompositionPreview.SetElementChildVisual(_compositionCanvas, container);
}
```

```
class Windows.Media.Playback.MediaPlayerSurface
提供对 ICompositionSurface 的访问权限，在 MediaPlayer 和 Compositor 之间共享，以通过使用 Windows.UI.
```

然而这样却不会自动调整视频大小，于是添加对 sizechanged 响应，并重新呈现：

```
private void SizeChange(object sender, SizeChangedEventArgs e)...

private void resetSize()
{
    if(ElementCompositionPreview.GetElementChildVisual(_compositionCanvas) != null&&video)
    {
        ElementCompositionPreview.GetElementChildVisual(_compositionCanvas).Dispose();
        MediaPlayer _mediaPlayer = ViewModel.MediaPlayer;
        _mediaPlayer.SetSurfaceSize(new Size(_compositionCanvas.Width, _compositionCanvas.Height));

        var compositor = ElementCompositionPreview.GetElementVisual(this).Compositor;
        MediaPlayerSurface surface = _mediaPlayer.GetSurface(compositor);
        SpriteVisual spriteVisual = compositor.CreateSpriteVisual();
```

又因为对播放视频和播放音乐逻辑不同，所以需要用到之前的设置的状态。
并且这样还带来一些麻烦，某些函数需要解除这种呈现，否则会出现很多 bug。

导入音频文件函数

```
private async void OpenAndLoadAsync()
{
    var openPicker = new FileOpenPicker();

    openPicker.SuggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.VideosLibrary;
    openPicker.FileTypeFilter.Add(".wmv");
    openPicker.FileTypeFilter.Add(".mp4");
    openPicker.FileTypeFilter.Add(".mp3");
    openPicker.FileTypeFilter.Add(".wma");

    StorageFile file = await openPicker.PickSingleFileAsync();
    if (file != null)
    {
        var mediaSource = MediaSource.CreateFromStorageFile(file);
        mediaSource.OpenOperationCompleted += MediaSource_OpenOperationCompleted;
        ViewModel.MediaPlayer.Source = mediaSource;
        if (file.FileType == ".mp3" || file.FileType == ".wma")
        {
            //
            StateChange(false);


        }
        else
```

全屏功能

```
private void FullScreenButton_Clicked(object sender, RoutedEventArgs e)
{
    ApplicationView view = ApplicationView.GetForCurrentView();
    bool isInFullScreenMode = view.IsFullScreenMode;
    if (isInFullScreenMode)
    {
        view.ExitFullScreenMode();
    }
    else
    {
        view.TryEnterFullScreenMode();
    }
}
```

关于 slider 与进度的绑定也是一大坑

我采用了 mediaplayer 或 mediaplayercontroller 绑定，似乎是因为没有实现 iNotifypropertychanged 接口无法实现进度条的移动。更改了多种方法，最终采用两个事件函数实现通信。

封面读取功能

```
using (StorageItemThumbnail thumbnail = await file.GetThumbnailAsync(ThumbnailMode.MusicView, 300))
{
    if (thumbnail != null && thumbnail.Type == ThumbnailType.Image)
    {
        var bitmapImage = new BitmapImage();
        bitmapImage.SetSource(thumbnail);
        var imageBrush = new ImageBrush();
        imageBrush.ImageSource = bitmapImage;
        _compositionCanvas.Background = imageBrush;
    }
}
```

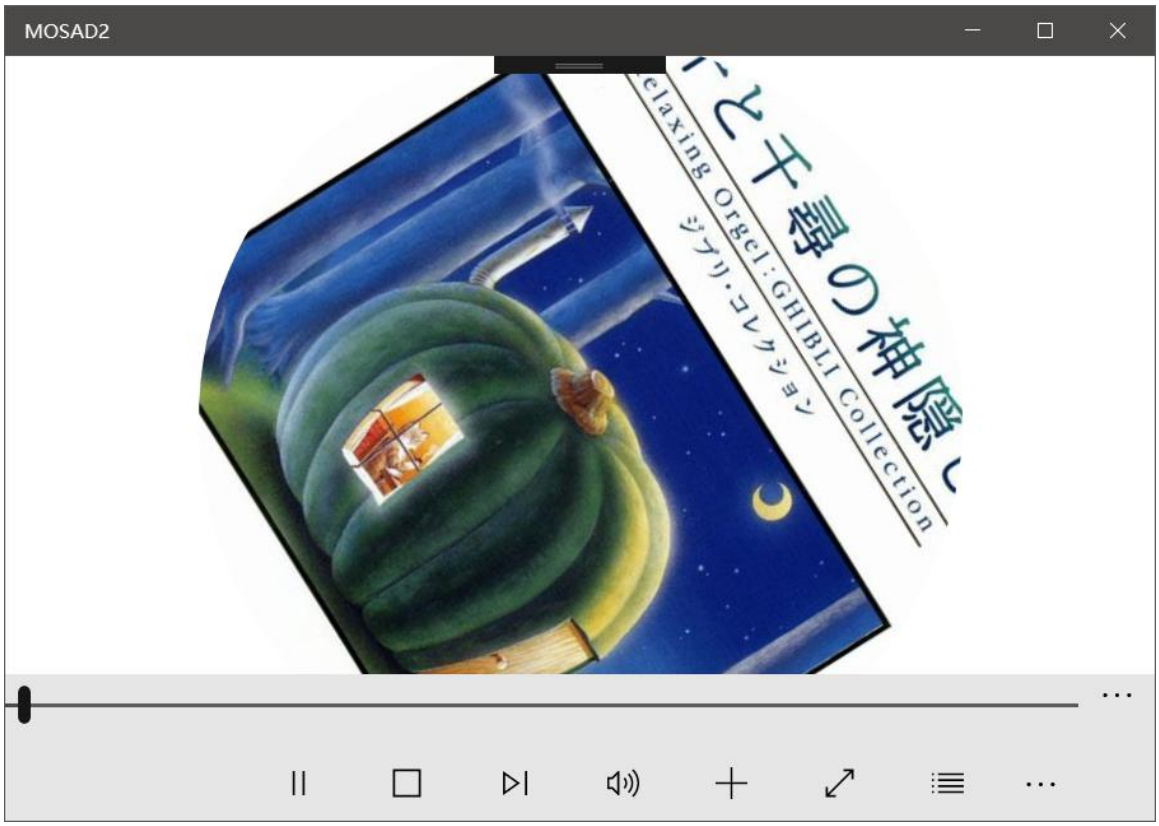
四、程序运行截图

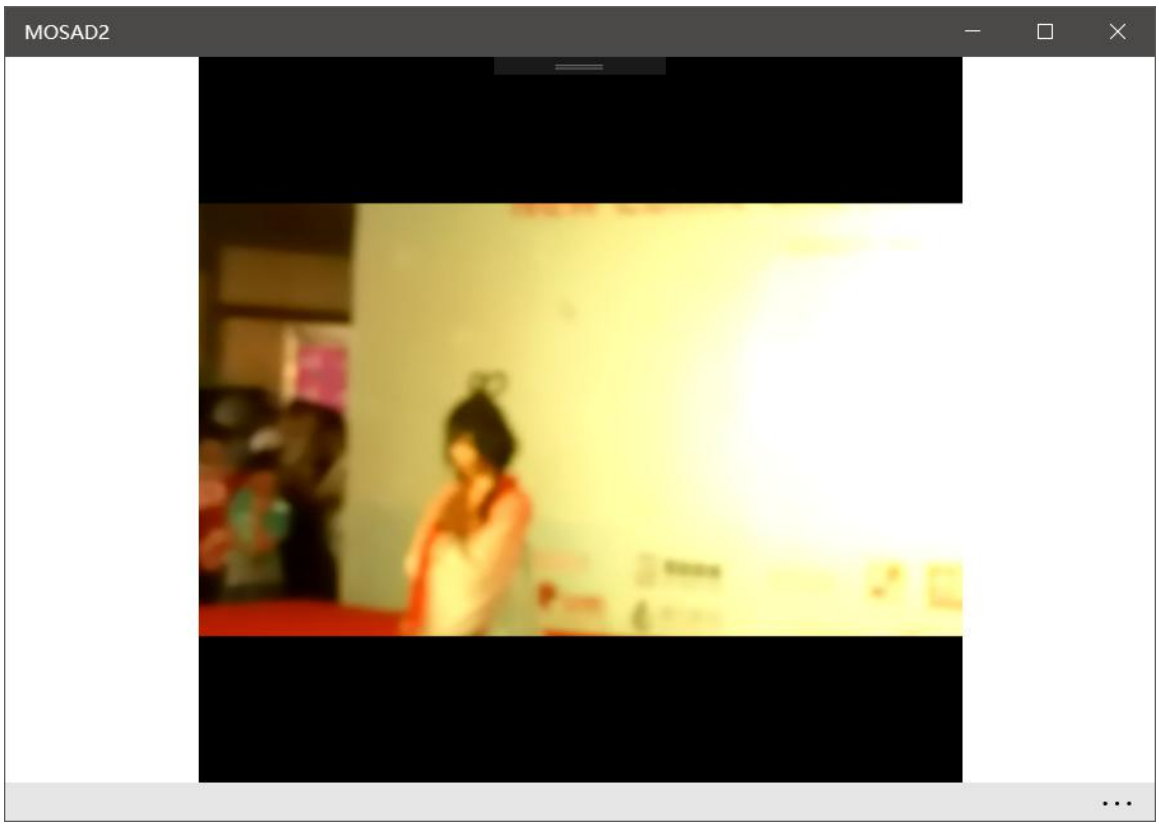
第七周


MOSAD1
Weather
Express
Search
广州
阴
28 -28 ℃
东北风2级
空气质量 :优
气象条件非常有利于空气污染物稀释、扩散和清除，可在室外正常活动。
舒适程度 :较舒适
白天有降雨，但会使人们感觉有些热，不过大部分人仍会有比较舒适的感觉。
穿衣指数 :热
天气热，建议着短裙、短裤、短薄外套、T恤等夏季服装。
2018-05-03
中雨 转中雨
22 -29 ℃
无持续风向1-2级
2018-05-04
雷阵雨 转多云
21 -27 ℃
无持续风向1-2级
2018-05-05
雷阵雨 转雷阵雨
23 -28 ℃
南风3-4级

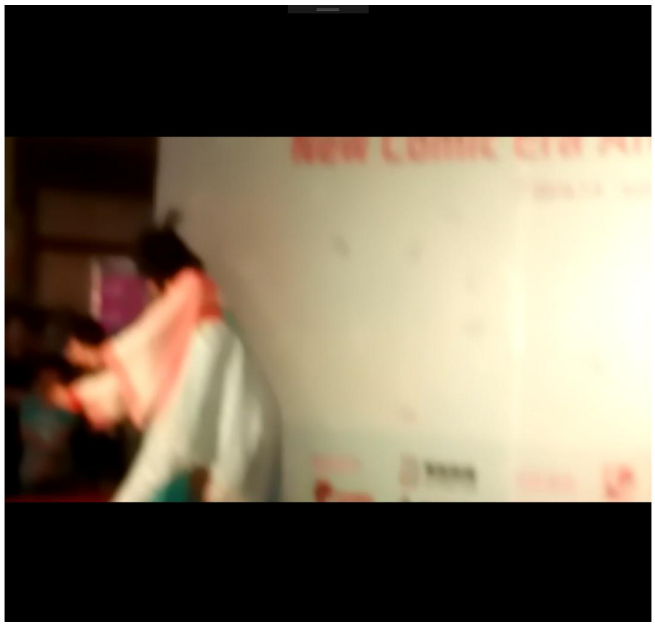

MOSAD1
Weather
Express
圆通
28729141397275
圆通
888528729141397275
2018-03-05 00:46:02
【广东省广州市白云区金沙洲公司】 取件人: 陈银巧 已收件
2018-03-05 02:10:53
【广东省广州市白云区金沙洲公司】 已收件
2018-03-05 02:25:26
【广东省广州市白云区金沙洲公司】 已打包
2018-03-05 03:24:02
【广东省广州市白云区金沙洲公司】 已发出 下一站 【广州转运中心】
2018-03-05 05:57:46
【广州转运中心】 已收入
2018-03-05 06:17:21
【广州转运中心】 已发出 下一站 【佛山转运中心】

第八周


MOSAD2
と千尋の神隠し
elaxing Orgel: GHIBLI Collection
ジブリ・コレクション


MOSAD2

全屏：

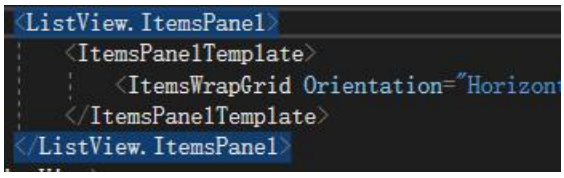


五、亮点与改进（可选）

第七周：

实现了 bonus，使用两个数据类型分别为 Xml 和 json 的 Api 接口。分别实现了快递查询和天气查询。

天气查询：功能不只实现当天的天气查询，还是实现了未来三天的天气预报。未来三天采用 ListView 绑定数据，通过



水平排布功能。

第八周：

1.实现了 bonus，选择本地文件，封面的旋转暂停复位等功能。

2.一个 canvas 控件完成封面和视频两种功能。

3.封面读取功能

六、遇到的问题

第七周：

网络访问实现方式很多，代码量也很少。主要的困难还是在生活功能 API 使用。

首先，各种 API 返回的数据结构差异很大，有些很复杂。因为开始时不懂还有辅助工具可以帮助生成对应的数据，浪费了很多时间。

其次，我选择的天气 API 有多个版本，最新版需要使用加密算法，由于对 C# 语言不太熟悉，尝试了一下最终还是避开了需要加密算法的版本，选择了一个旧版本接口使用。

另外，快递查询是最后实现的功能。天气接口返回的数据类型是 JSON，为了实现 bonus，快递查询接口只能选择 XML 类型，然而最终只能找到一个免费查询共 100 次的接口，使用几次后可能很快该接口很快无法使用。

第八周：

使用 canvas 实现封面旋转和视频呈现两种功能遇到了一些困难。

1. canvas 的圆形实现

网上没有直接的解决方案，在查阅相关资料中发现使用 border 可以控制边的圆度。

2. 视频的大小改变

当呈现到 Windows.UI.Composition 时由于并非真正在 canvas 上呈现，也不会随着 canvas 改变而改变大小，所以需要重新呈现。

七、思考与总结

由于对 uwp 各种类不熟悉，官方文档的功能描述也有点模糊，所以实现一些东西时有些困难。解决问题也遇到了很多困难。

关于函数的命名和功能的分离，即使两个功能基本同时出现，也不要写在一起，除非可以抽象成一个可以命名的更大的功能。否则逻辑混乱后，自己也看不懂自己在做什么。

另外我发现，在查看某些类的功能时直接通过 Ctrl 键进入定义查看注释可能比查看官方文档的描述更方便，也更明确。