16340280 余漫霖

实验名称：Homework2

一、参考资料

stackoverflow

CSDN博客

微软官方技术文档

二、实验步骤

第四周：

> - （Mainpage.xaml.cs）在MainPage中点击checkbox出现横线，输入数据（选择图片），挂起并关闭程序，重新启动时，程序显示在Mainpage界面，并且点击的checkbox与对应横线，数据与图片都存在。
> - （Newpage.xaml.cs）在NewPage中输入数据（或选择图片），挂起并关闭程序，重新启动时，程序显示在Newpage界面，数据与图片都存在。

用比较笨的办法，在OnNavigatedFrom时循环把数据一个个地写入composite，又在OnNavigatedTo时循环读取数据。

```
//OnNavigatedFrom
            //保存窗口大小和状态
               if (Window.Current.Bounds.Width >= 800) state = stateAllView;
               composite["state"] = state;

               //记录要序列化的数据
               int count = ViewModels.AllItems.Count;
               composite["count"] = count;
               composite["selected"] = -1;
               for(int i = 0; i < count; i++)
               {
                   if(ViewModels.SelectedItem != null && ViewModels.SelectedItem.Id ==
ViewModels.AllItems[i].Id)
                       composite["selected"] = i;
                   composite["id" + i] = ViewModels.AllItems[i].Id;
                   composite["title" + i] = ViewModels.AllItems[i].Title;
                   composite["description" + i] = ViewModels.AllItems[i].Description;
                   composite["completed" + i] = ViewModels.AllItems[i].Completed;
                   composite["date" + i] = ViewModels.AllItems[i].Date;
                   composite["imageType" + i] = ViewModels.AllItems[i].ImageType;
               }

               composite["editId"] = EditItem.latestInstance.Id;
               composite["editTitle"] = EditItem.latestInstance.Title;
               composite["editDescription"] = EditItem.latestInstance.Description;
               composite["editDate"] = EditItem.latestInstance.Date;

               ApplicationData.Current.LocalSettings.Values["newPage"] = composite;
```

关于Edit视图内的ImagePicker的图片保存，

```
//挂起时保存图片
var file = await imgPicker.PickSingleFileAsync();
ApplicationData.Current.LocalSettings.Values["imagepicker"]
    = StorageApplicationPermissions.FutureAccessList.Add(file);
IRandomAccessStream ir = await file.OpenAsync(FileAccessMode.Read);

//恢复时读取imagePicker应有的图片
if (ApplicationData.Current.LocalSettings.Values["imagepicker"] != null)
{
    StorageFile file;
    file = await
StorageApplicationPermissions.FutureAccessList.GetFileAsync((String)ApplicationData.Current.Loca
lSettings.Values["imagepicker"]);
    IRandomAccessStream ir = await file.OpenAsync(FileAccessMode.Read);
    BitmapImage bmp = new BitmapImage();
    await bmp.SetSourceAsync(ir);
    EditItem.latestInstance.setImage(bmp);
    ApplicationData.Current.LocalSettings.Values["imagepicker"] = null;
}
```

第五周：

> - 采用Adaptive Tile制作动态磁贴（覆盖至少small、medium、wide）
> - 每添加一条项目，磁贴能进行更新，并且更新的内容循环展示（1-2-3-4-5-1-2-3-4.....）
> - Small tile不要求循环显示，但能够在新建项目时显示更新的信息。

在Tiles.xml里写了磁贴模板，在Services/TileService.cs里写了磁贴服务的定义。

PS：有尝试使用ToastService.cs来实现通知，但未加入Todo应用的正式结果。

> 在MenuFlyoutItem中增加Share选项，点击后相应条目能以邮件方式进行共享（不要求动态共享图片）

```
//ListItem.xaml.cs
private void Share_Click(object sender, RoutedEventArgs e)
{
    MenuFlyoutItem se = sender as MenuFlyoutItem;
    var dc = se.DataContext as TodoItem;
    EditItem.latestInstance.ViewModels.SelectedItem = dc;
    DataTransferManager.ShowShareUI();
}
public void OnShareDataRequested(DataTransferManager sender, DataRequestedEventArgs args)
{
    var dp = args.Request.Data;
    var deferral = args.Request.GetDeferral();
    dp.Properties.Title = EditItem.latestInstance.ViewModels.SelectedItem.Title;
    dp.Properties.Description = EditItem.latestInstance.ViewModels.SelectedItem.Description;
    dp.SetText(dp.Properties.Description);
    dp.SetBitmap(RandomAccessStreamReference.CreateFromUri(new Uri(MainPage.defaultImage)));
    deferral.Complete();
}
```

第六周：

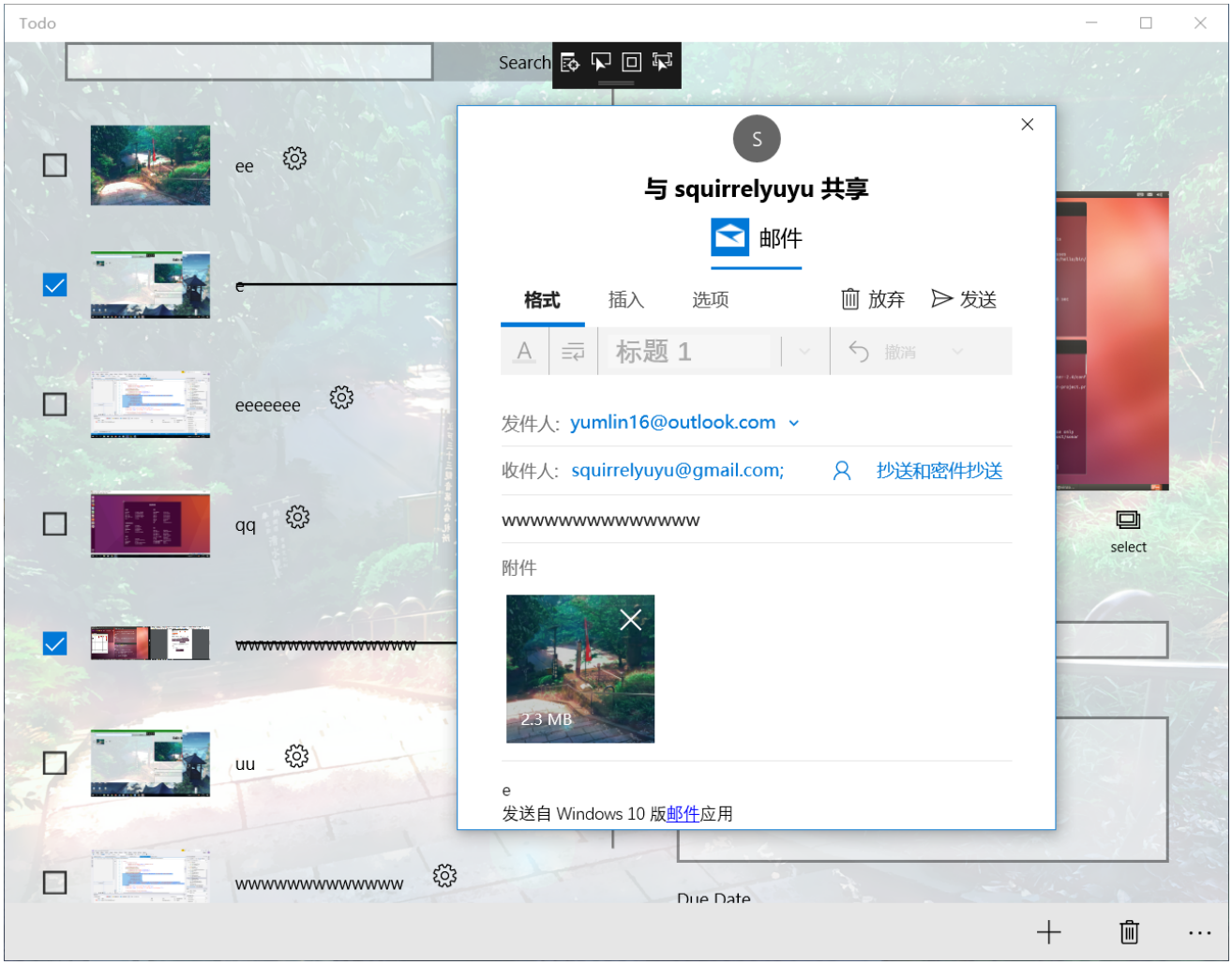
> SQLite数据库本地存储：
>
> - 实现todo表项的增、删、改、查；并且能保存及恢复应用状态。
> - 需要保存：title，description，complete， date（年月日即可），image（**Bonus**项）
> - 查询时为模糊查询，如下图，查询“现”即可显示日期为title或description或date中含有“现”的item（查询到的item用字符串表示title+description+date。若有多条，则每行一个item）。

增加了一个DataAccess类。

在App.xaml.cs里添加了字段 `public DataAccess dataAccess` 。

在提交编辑、删除表项、点击复选框的相关代码里都添加了相应的数据库操作。

三、关键步骤截图



共享。

四、亮点与改进

使用Frame实现两个页面的整合

在第3周之后，我觉得MainPage包括了NewPage的功能，但它们又是分开的，这样有代码冗余，显得不优雅，所以借鉴了大佬的代码，使用Frame来重构Todo。

有3个xaml文件：

- MainPage.xaml——主页面，包含了两个Frame
- ListItem.xaml——显示TodoItem的列表（在实验后期加上了搜索框）
- EditItem.xaml——编辑TodoItem

```
//MainPage.xaml.cs
        private const String stateEdit = "Edit";
        private const String stateList = "List";
        private const String stateAllView = "All";
        private String state = stateAllView;

        public MainPage()
        {
            //......

            left.Navigate(typeof(ListItem));
            right.Navigate(typeof(EditItem));

            if (Window.Current.Bounds.Width < 800)
                VisualStateManager.GoToState(this, stateList, false);

            if (right.Visibility == Visibility.Collapsed) state = stateList;
            else if (left.Visibility == Visibility.Collapsed) state = stateEdit;

            this.SizeChanged += new SizeChangedEventHandler(Resize);
        }

        private void Resize(object sender, SizeChangedEventArgs e)
        {
            double neww = e.NewSize.Width;
            double oldw = e.PreviousSize.Width;
            if (neww >= 800) state = stateAllView;
            else
            {
                if (oldw >= 800) state = stateList;
            }
            VisualStateManager.GoToState(this, state, false);
        }

//MainPage.xaml
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition x:Name="leftColumn"/>
            <ColumnDefinition x:Name="rightColumn"/>
        </Grid.ColumnDefinitions>

        <Frame Name="left" Grid.Column="0" Grid.Row="0"/>
        <Frame Name="right" Grid.Column="1" Grid.Row="0"/>

        <VisualStateManager.VisualStateGroups>
            <VisualStateGroup x:Name="VisualStateGroup">
                <VisualState x:Name="List">
                    <VisualState.Setters>
                        <Setter Target="rightColumn.Width" Value="0" />

                        <Setter Target="left.Visibility" Value="Visible"/>
```

```
                    <Setter Target="right.Visibility" Value="Collapsed"/>
                </VisualState.Setters>
            </VisualState>
            <VisualState x:Name="Edit">
                <VisualState.Setters>
                    <Setter Target="leftColumn.Width" Value="0" />
                    <Setter Target="left.Visibility" Value="Collapsed"/>
                    <Setter Target="right.Visibility" Value="Visible"/>
                </VisualState.Setters>
            </VisualState>
            <VisualState x:Name="All">
                <VisualState.StateTriggers>
                    <AdaptiveTrigger MinWindowWidth="800" />
                </VisualState.StateTriggers>
            </VisualState>
        </VisualStateGroup>
    </VisualStateManager.VisualStateGroups>
</Grid>
```

本地保存图片

- 在每次imagePicker选择图片时，都将图片存入LocalFolder里的tempImage；EditItem也会记录tempImage的名字

```
//把图片存为LocalFolder/tempImage
tempImage = "tempImage" + imageType;
try
{
    //由该图片创建tempImage
    await file.CopyAsync(ApplicationData.Current.LocalFolder, tempImage);
}
catch(Exception ee)
{
    //已经有tempImage存在，则替换
    await file.CopyAndReplaceAsync(
      await ApplicationData.Current.LocalFolder.GetFileAsync(tempImage));
}
```

- 有几种情况：从页面使用默认图片建立/更新item，从页面使用新的图片建立/更新item，从数据库建立item。在调用ViewModel的Add/Update方法时，会根据情况设置saveImage的布尔值。
- 如果ViewModel接收到的saveImage参数为true，则由EditItem的tempImage保存新的图片，保存在LocalFolder中，名字为【TodoItem的Id + 图片格式】

```
        public async void saveIcon(TodoItem item)
         {
             if (item.ImageType.Equals("default")) return;
             StorageFolder storageFolder = ApplicationData.Current.LocalFolder;
             StorageFile file = await
storageFolder.GetFileAsync(EditItem.latestInstance.tempImage);

             try
             {
                 await file.CopyAsync(storageFolder, item.Id + item.ImageType,
NameCollisionOption.ReplaceExisting);
             }
             catch(Exception ee)
             {

             }
         }
```

五、遇到的问题

本地图片存储的进程问题

在上面的saveIcon代码块里，原本是没有try/catch块的，但是运行后，如果选择新图片并更新，在 `await file.CopyAsync(storageFolder, item.Id + item.ImageType, NameCollisionOption.ReplaceExisting);` 处就会抛出异常，说这个文件正在被某个进程占用。

但是关闭程序后，再重新打开，新图片就显示了。

所以选择使用try/catch来忽略异常。

奇怪的syntax error

报错如下：

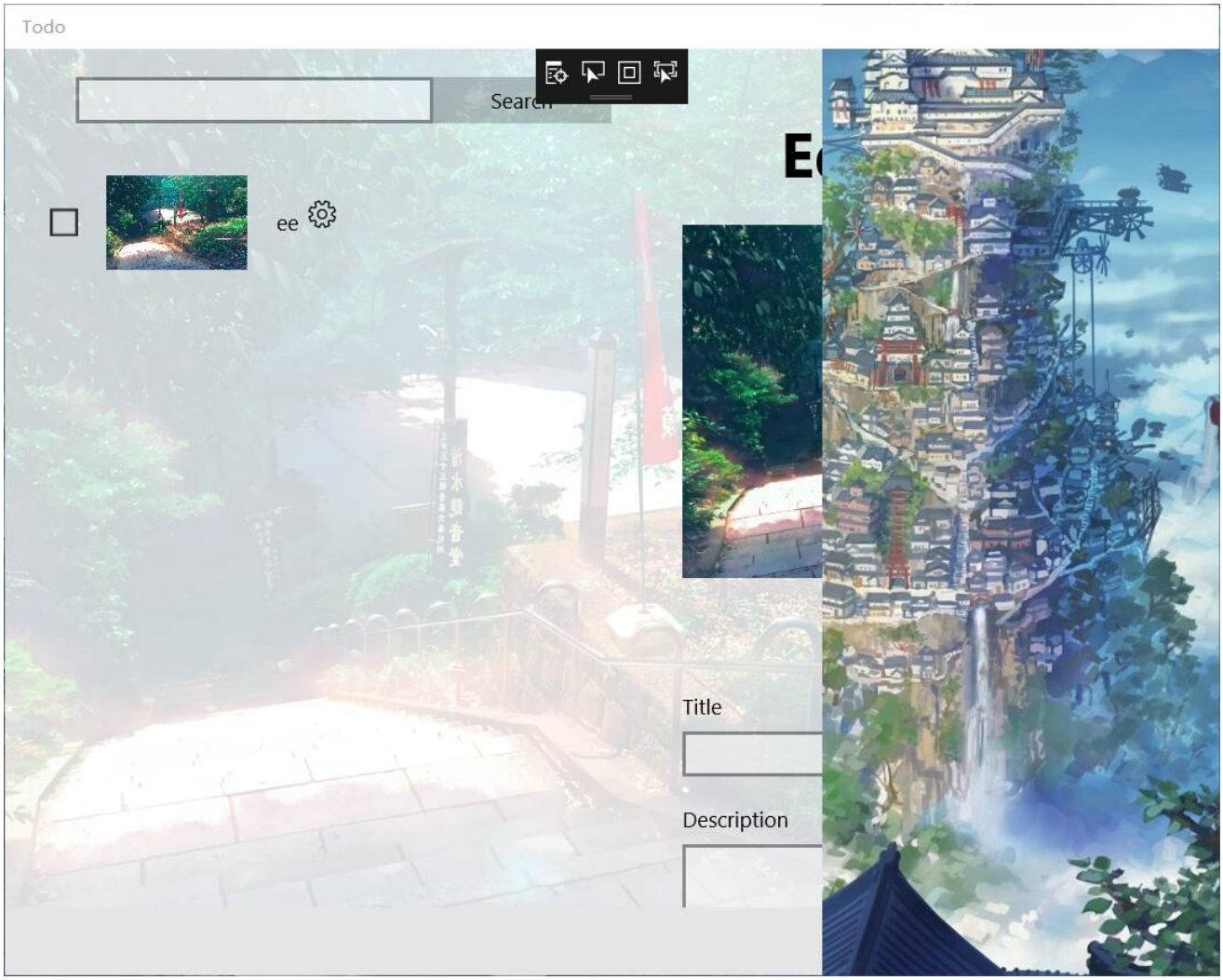
```
SQLitePCL.SQLiteException

HResult=0x80131500

Message=Unable to prepare the sql statement: UPDATE TodoItems SET Completed = ? WHERE Id=?
Details: near "UPDATE TodoItems SET Completed ": syntax error
```

解决：

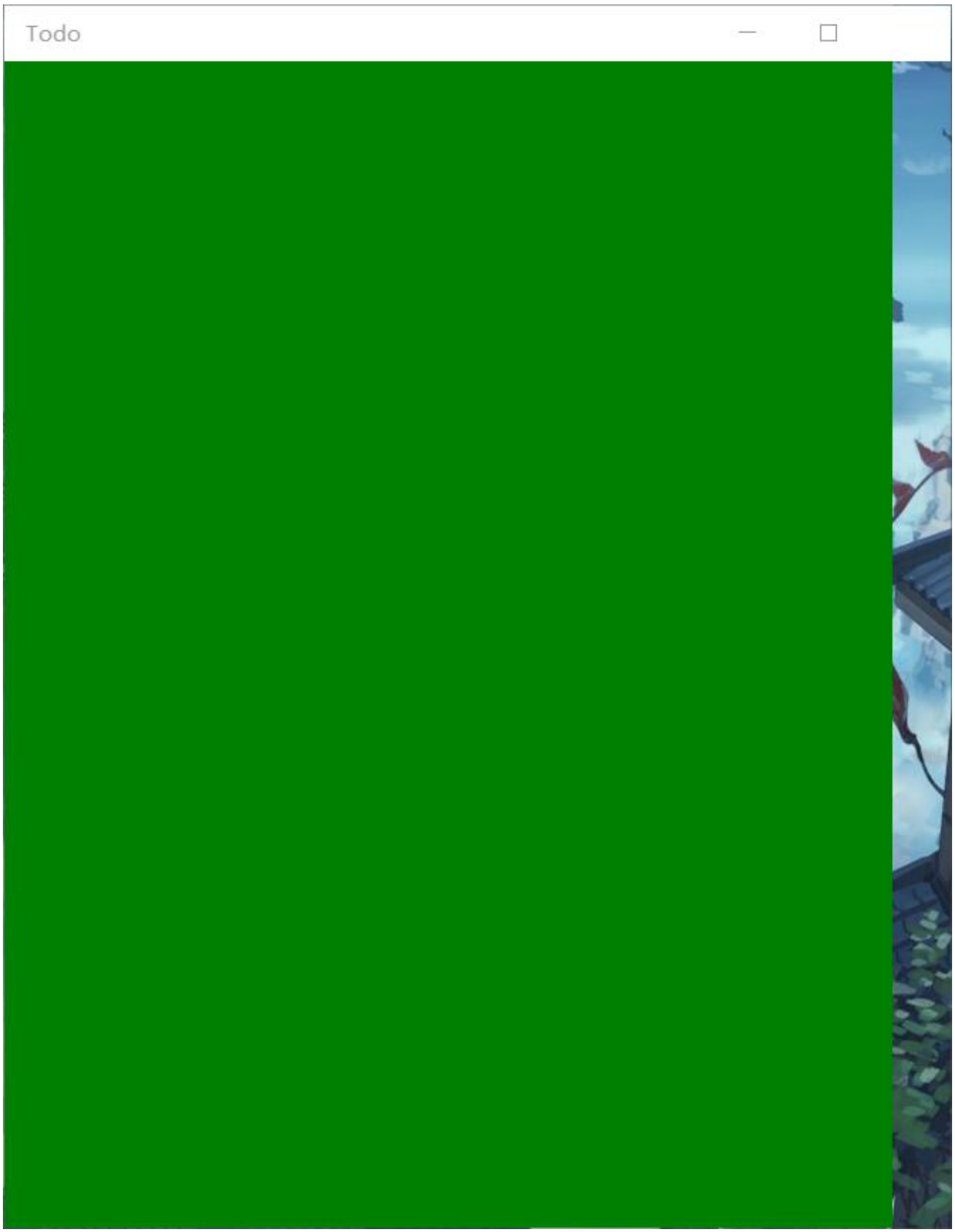
在SQLite Expert中手打语句后执行，直到语句运行成功时才拷到代码里。之后代码运行通过了。

不知道是不是因为先前复制的语句有编码问题？还是有空格要求？还是我的vs抽风了？

窗口显示错乱



（↑蓝色部分是我的桌面。）



（↑如果最小化窗口，过了一段时间后再打开，就会这样，无法恢复到正常界面。）

解决：因为我应该没有动关于visual state的代码，所以对这个显示感到很纳闷。发给同学跑了后，发现是我电脑抽风了，需要重启电脑。

齿轮不总位于最右边

未解决。

六、思考与总结

- 经过本次实验后，我越发体会到项目开发的不易。每添加一个功能，都要修改很多代码块，要照料到很多角落，甚至要重构代码和数据结构。
- C#的线程有点微妙，如果时间够的话，还是希望我可以好好研究一下同步和异步的问题的，然而并没有那么多时间。[捂脸]
- 我觉得及时写注释和保持记录实验过程中遇到的问题很有必要，这样有助于消化知识和加深印象，也便于后期撰写实验报告。这次在这方面做得不太好，希望下次能进一步加强。
- 即便不使用可视化工具进行开发，也可以用它来辅助调试。比如，我用SQLite Expert看数据库的数据，可以在调试过程中直接看到一些异常的数据值，从而更快地定位问题。