

Skyline 开发入门 (C#)

经过几个月的摸索，对 skyline 的开发有了一定的了解。入门的阶段总是让人郁闷，现将本人的经验进行简单的总结，方便自己，方便他人。本文是个人的经验总结，如果有不同的见解，欢迎各位朋友一起讨论。欢迎转载本文，转载是请注明本文地址 www.3snews.net/?gisbamboo，谢谢！

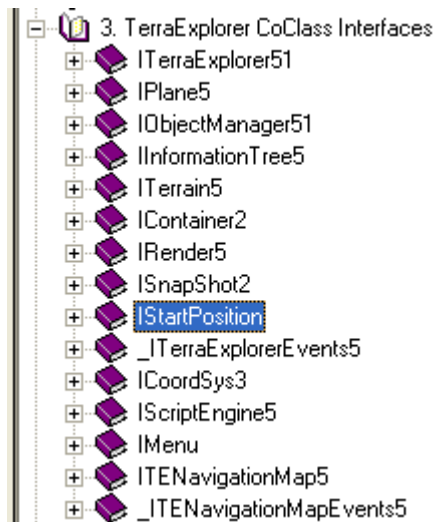
本文包括两方面的内容，一方面是介绍 skyline 开发帮助文档，另一方面是根据帮助文档编写一个 skyline 的例程。

一、帮助文档介绍

安装好 skyline pro 后，在安装目录下可以找到开发帮助文档 ProgrammersGuide.chm 我是把软件安装在 C 盘，C:\Program Files\Skyline\TerraExplorer Pro\Help。

帮助文档共有七个章节，skylin 桌面开发主要是用到第三章和第四章。

1、TerraExplorer CoClass Interfaces



`TerraExplorerClass` 类是创建 TerraExplorer 组件类的入口，**TerraExplorer CoClass Interfaces** 中所有的接口都实现了 `TerraExplorerClass` 类。

以获得 `IInformationTree5` 对象为例，代码如下：

```
TerraExplorer terraExplorer = new TerraExplorerClass();  
IInformationTree5 iInformationTree5 = new TerraExplorerClass();
```

以上两行代码，就可以得到 `iInformationTree5` 实例了，就可以调用 `iInformationTree5` 的方法。其他组件类接口的使用方法都是类似的。

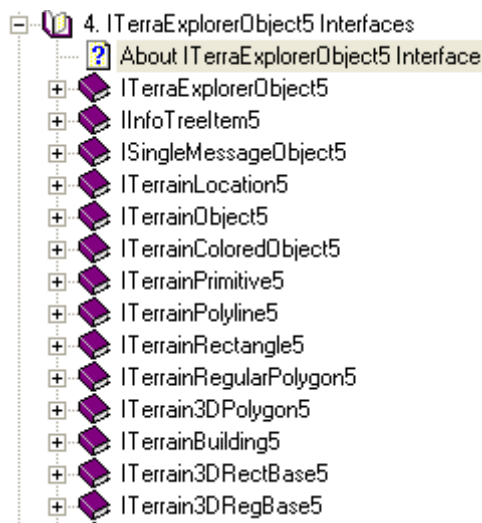
`TerraExplorerClass` 除了能实现组件类接口，也能直接实例化。

```
TerraExplorerClass terraExplorerClass = new TerraExplorerClass();
```

`terraExplorerClass` 对象提供了丰富的方法，最常用的就是 `Invoke` 方法了，这个方法实际上是和 `IMenu` 接口的 `Invoke` 方法是一样的，只要传入菜单命令 ID，就可以调用 skyline 已经封装好的很多功能了，菜单命令可以参考开发帮助 `IMenu` 接口的介绍。

2、ITerraExplorerObject5 Interfaces

主要是介绍创建 TerraExplorer 各种对象的接口。



以上接口不能通过直接实现，必须通过组件类实例的方法创建。

实现 `ITerrainPolyline` 接口的代码如下：

方法一：

```
ITerrainPolyline iTerrainPolyline= null;
iTerrainPolyline= terraExplorerClass.CreatePolyline(null, Color,
HeightStyleCode.HS_ON_TERRAIN, groupID, “线的名称”);
```

方法二：

```
IObjectManager51 iObjectManager51 = (IObjectManager51)terraExplorer;
ITerrainPolyline iTerrainPolyline= null;
iTerrainPolyline =iObjectManager51.CreatePolyline(null, Color, HeightStyleCode.HS_ON_TERRAIN,
groupID, “线的名称”);
```

二、开发实例——路线转移

路线转移功能，可以通过鼠标画一个转移的路线，双击鼠标结束画线。创建好路线之后，再创建一个动态对象，动态对象以之前创建好的路线作为运动的轨迹。

1、创建私有变量

```
private TerraExplorerClass terraExplorerClass;
ITerrainPolyline iTerrainPolyline= null;
```

2、创建路线

```
iTerrainPolyline= terraExplorerClass.CreatePolyline(null, Color,
HeightStyleCode.HS_ON_TERRAIN, groupID, “路线”);
//创建鼠标左键委托
this. terraExplorerClass.OnLButtonDown += new
_ITerraExplorerEvents5_OnLButtonDownEventHandler(TE_OnLButtonDown);
//创建鼠标双击委托
this. terraExplorerClass.OnLButtonDb1C1k +=
new_ITerraExplorerEvents5_OnLButtonDb1C1kEventHandler(TE_OnLButtonDb1C1k);
```

```
this. TE. SetMouseInputMode(MouseInputMode.MI_COM_CLIENT);
```

3、私有方法

```
/// <summary>
/// 添加线坐标
/// </summary>
/// <param name="Flags"></param>
/// <param name="X"></param>
/// <param name="Y"></param>
/// <param name="pbHandled"></param>
void TE_OnLButtonDown(int Flags, int X, int Y, ref object pbHandled)
{
    object x, y, flag;
    double terrainX, terrainY, terrainH;
    //获得鼠标坐标
    this. terraExplorerClass. GetMouseInfo(out flag, out x, out y);
    //将鼠标坐标转为地理坐标
    this. terraExplorerClass. IRender3_ScreenToTerrain(Convert.ToInt32(x),
Convert.ToInt32(y), out terrainX, out terrainY, out terrainH);
    //将地理坐标添加到鼠标点击创建的线中
    iTerrainPolyline.AddVertex(terrainX, newPolylineHeigh, terrainY, (int)flag);

}

/// <summary>
/// 双击结束
/// </summary>
/// <param name="Flags"></param>
/// <param name="X"></param>
/// <param name="Y"></param>
/// <param name="pbHandled"></param>
void TE_OnLButtonDblClk(int Flags, int X, int Y, ref object pbHandled)
{
    this. terraExplorerClass. OnLButtonDown -= new
_ITerraExplorerEvents5_OnLButtonDownEventHandler(TE_OnLButtonDown);
    this. terraExplorerClass. OnLButtonDblClk -= new
_ITerraExplorerEvents5_OnLButtonDblClkEventHandler(TE_OnLButtonDblClk);
    this. terraExplorerClass. SetMouseInputMode(MouseInputMode.MI_FREE_FLIGHT);
}

/// <summary>
/// 根据对象名称获取ItemID
/// </summary>
/// <param name="grouName">组名</param>
/// <param name="Name">对象名称</param>
```

```

/// <returns></returns>
public static int GetItemIDByName(string Name)
{
    int ItemID = terraExplorerClass.GetNextItem(0, ItemCode.CHILD);
    while (ItemID != 0)
    {
        string itemName = terraExplorerClass.GetItemName(ItemID);
        if (itemName.Equals(Name))
        {
            return ItemID;
        }
        ItemID = terraExplorerClass.GetNextItem(ItemID, ItemCode.NEXT);
    }
    return ItemID;
}

```

4、创建动态对象，并与创建好的路线作为运动的轨迹

//创建动态对象

```

IObjectManager51 objectCar = new TerraExplorerClass();
    string filename = @".\data\小车.xpc";
    ITerrainDynamicObject5 iTDynamicObject =
objectCar.CreateDynamicObject(DynamicMotionStyle.MOTION_GROUND_VEHICLE,
    DynamicObjectType.DYNAMIC_3D_MODEL, filename, 2.0,
HeightStyleCode.HS_ON_TERRAIN, gorupID, obejectName);
    iTDynamicObject.TurnSpeed = 200;
    iTDynamicObject.Distance = 300;
    iTDynamicObject.CircularRoute = 0;
    iTDynamicObject.ScaleFactor = 3;

//添加路线里面的坐标
int itemID = this.GetItemIDByName("路线");
ITerrainPolyline5 iTerrainPolyline5=terraExplorerClass.GetObjectEx(itemID,
"ITerrainPolyline5");
//添加路线里的坐标点
int NumofPoint = iTerrainPolyline5.NumOfVertices;
    object X, Height, Y;
    for (int i = 0; i <= NumofPoint - 1; i++)
    {
        iTerrainPolyline5.GetVertex(i, out X, out Height, out Y);
        iTerrainDynamicObject5.AddWaypoint((double)X, (double)Height,
(double)Y, speed, i);
    }

//视角定位至路线的第一个点
terraExplorerClass.FlyTo((double)X, (double)Y, (double)Height, 123, 0, -90, "FlyToLocation");
//飞至该路线
terraExplorerClass.FlyToObject(iTerrainDynamicObject5.ID, ActionCode.AC_PLAY);

```

三、小结

`TerraExplorerClass` 是 skyline 开发的起点，本身能实例化，也实现了 `TerraExplorer` 接口和所有的组件类接口。应该说 `TerraExplorerClass` 是 skyline 开发最重要的一个类了，如果要做成插件式的三维系统，只要实例化 `TerraExplorerClass`，并将其对象声明为公共变量，就可以与地图窗口和树控件进行交互了。