#region namespace imports

using System;

using System.Collections;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

using Cognex.VisionPro;

using Cognex.VisionPro.ToolBlock;

using Cognex.VisionPro3D;

using Cognex.VisionPro.ID;

using System.Linq;//\*\*\*

using System.Collections.Generic;

#endregion

public class CogToolBlockSimpleScript : CogToolBlockAdvancedScript

{

/// <summary>

/// Called when the parent tool is run.

/// Add code here to customize or replace the normal run behavior.

/// </summary>

/// <param name="message">Sets the Message in the tool's RunStatus.</param>

/// <param name="result">Sets the Result in the tool's RunStatus</param>

/// <returns>True if the tool should run normally,

/// False if GroupRun customizes run behavior</returns>

public override bool GroupRun(ref string message, ref CogToolResultConstants result)

{

// To let the execution stop in this script when a debugger is attached, uncomment the following lines.

// #if DEBUG

// if (System.Diagnostics.Debugger.IsAttached) System.Diagnostics.Debugger.Break();

// #endif

// Run each tool using the RunTool function

foreach(ICogTool tool in Tools)

RunTool(tool, ref message, ref result);

**//实现排序 先按照 Y坐标 然后按照X坐标**

**CogIDResults res = Tools.CogIDTool1.Results;**

**//定义一个泛型列表**

**List<CogIDResult> myList = new List<CogIDResult>();**

**List<CogIDResult> myList2 = new List<CogIDResult>();**

**foreach(CogIDResult r in res)**

**{**

**myList.Add(r);**

**}**

**myList.Sort(( m,n) => { return m.CenterY.CompareTo(n.CenterY); });//按照x坐标从小到大排序**

**// myList2.Sort(( m,n) => { return m.CenterY.CompareTo(n.CenterY); });//按照x坐标从小到大排序**

**List<int> dis = new List<int>();**

**dis.Add(0);//添加起始位**

**for(int i = 1 ;i < myList.Count ;i++)**

**{**

**double interval = Math.Abs(myList[i].CenterY - myList[i - 1].CenterY);**

**if( interval > 60 )//假定Y坐标误差间距为60**

**{**

**dis.Add(i);//记下分组间隙**

**//MessageBox.Show(i.ToString());**

**}**

**}**

**//MessageBox.Show(dis.Count.ToString());**

**for(int i = 0;i < dis.Count ;i++) //处理每一行**

**{**

**int LastIndex = 0;**

**List<CogIDResult> temp = new List<CogIDResult>();**

**if(i == dis.Count - 1 )//最后一个分隔元素**

**{**

**LastIndex = myList.Count;**

**}**

**else**

**{**

**LastIndex = dis[i + 1];**

**}**

**for(int j = dis[i] ; j < LastIndex ; j++)**

**{**

**temp.Add(myList[j]);**

**}**

**temp.Sort(( m,n) => { return m.CenterX.CompareTo(n.CenterX); });//按照x坐标从小到大排序**

**myList2.AddRange(temp);**

**temp.Clear();**

**}**

**//MessageBox.Show(myList2.Count.ToString());**

**//打印排序好的结果**

**string s = "";**

**foreach( CogIDResult r in myList2 )**

**{**

**//s += r.CenterX.ToString() + " " + r.CenterY.ToString() + " \r\n";**

**s += r.DecodedData.DecodedString;**

**}**

**MessageBox.Show(s);**

return false;

}

#region When the Script is Initialized

/// <summary>

/// Perform any initialization required by your script here

/// </summary>

/// <param name="host">The host tool</param>

public override void Initialize(Cognex.VisionPro.ToolGroup.CogToolGroup host)

{

// DO NOT REMOVE - Call the base class implementation first - DO NOT REMOVE

base.Initialize(host);

}

#endregion

}

#region auto-generated

// ------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

// VisionPro Version: 9.0 CR2

// Generated: 2022-4-3 20:28:33

// Name: CogToolBlockScript, Version: 1.0

// Options:

// IncludeMainScriptClass, OverrideGroupRun,

// OverrideInitialize, IncludeInputsAdapterClass,

// IncludeOutputsAdapterClass, IncludeToolsAdapterClass,

// ExposeToolBlockReference, PlaceInAutoGeneratedRegion,

// AdapterClassIsEnumerable, IncludeRunToolFunction

//

// Changes to this code below this comment may cause incorrect behavior

// and will be lost if the code is regenerated.

// </auto-generated>

// ------------------------------------------------------------------------------

public class CogToolBlockAdvancedScript : CogToolBlockAdvancedScriptBase

{

#region Private Member Variables

private Cognex.VisionPro.ToolBlock.CogToolBlock mToolBlock;

private CollectionAdapterInputs mInputs;

private CollectionAdapterTools mTools;

#endregion

/// <summary>

/// Called when the parent tool is run.

/// Add code here to customize or replace the normal run behavior.

/// </summary>

/// <param name="message">Sets the Message in the tool's RunStatus.</param>

/// <param name="result">Sets the Result in the tool's RunStatus</param>

/// <returns>True if the tool should run normally,

/// False if GroupRun customizes run behavior</returns>

public override bool GroupRun(ref string message, ref CogToolResultConstants result)

{

// To let the execution stop in this script when a debugger is attached, uncomment the following lines.

// #if DEBUG

// if (System.Diagnostics.Debugger.IsAttached) System.Diagnostics.Debugger.Break();

// #endif

return false;

}

#region When the Script is Initialized

/// <summary>

/// Perform any initialization required by your script here

/// </summary>

/// <param name="host">The host tool</param>

public override void Initialize(Cognex.VisionPro.ToolGroup.CogToolGroup host)

{

// DO NOT REMOVE - Call the base class implementation first - DO NOT REMOVE

base.Initialize(host);

// Initialize convenience types for ToolBlock Inputs, Outputs, and Tools

this.mInputs = new CollectionAdapterInputs(((Cognex.VisionPro.ToolBlock.CogToolBlock)(host)));

this.mTools = new CollectionAdapterTools(((Cognex.VisionPro.ToolBlock.CogToolBlock)(host)));

// Store a local copy of the script host

this.mToolBlock = ((Cognex.VisionPro.ToolBlock.CogToolBlock)(host));

}

#endregion

/// <summary>

/// Access the Tool's Inputs through an adapter class

/// </summary>

public CollectionAdapterInputs Inputs

{

get

{

return this.mInputs;

}

}

/// <summary>

/// Access the Tool's Tools through an adapter class

/// </summary>

public CollectionAdapterTools Tools

{

get

{

return this.mTools;

}

}

#region Public Methods

/// <summary>

/// Called from a script to run a tool.

/// </summary>

/// <param name="tool">The tool to run.</param>

/// <param name="message">The RunStatus Message.</param>

/// <param name="result">The RunStatus Result.</param>

/// <exception cref="System.Exception">

/// Thrown when <paramref name="tool"/> returns and

/// the calling tool's AbortRunOnToolFailure property is True.</exception>

protected virtual void RunTool(Cognex.VisionPro.ICogTool tool, ref string message, ref CogToolResultConstants result)

{

this.mToolBlock.RunTool(tool, ref message, ref result);

}

#endregion

}

#region Adapter Classes

public class CollectionAdapterInputs

{

private Cognex.VisionPro.ToolBlock.CogToolBlock mToolBlock;

public CollectionAdapterInputs(Cognex.VisionPro.ToolBlock.CogToolBlock toolBlock)

{

this.mToolBlock = toolBlock;

}

public Cognex.VisionPro.ICogImage OutputImage

{

get

{

try

{

return ((Cognex.VisionPro.ICogImage)(this.mToolBlock.Inputs["OutputImage"].Value));

}

catch (System.Exception ex)

{

throw new System.Exception(String.Format(Cognex.VisionPro.CogLocalizer.GetString(typeof(Cognex.VisionPro.ToolGroup.Resources.CogResourceKeys), Cognex.VisionPro.ToolGroup.Resources.CogResourceKeys.RkScritptAdapterMemberAccessError), this.mToolBlock.Name, "Inputs.OutputImage"), ex);

}

}

}

}

public class CollectionAdapterTools : System.Collections.IEnumerable

{

private Cognex.VisionPro.ToolBlock.CogToolBlock mToolBlock;

public CollectionAdapterTools(Cognex.VisionPro.ToolBlock.CogToolBlock toolBlock)

{

this.mToolBlock = toolBlock;

}

public Cognex.VisionPro.ID.CogIDTool CogIDTool1

{

get

{

try

{

return ((Cognex.VisionPro.ID.CogIDTool)(this.mToolBlock.Tools["CogIDTool1"]));

}

catch (System.Exception ex)

{

throw new System.Exception(String.Format(Cognex.VisionPro.CogLocalizer.GetString(typeof(Cognex.VisionPro.ToolGroup.Resources.CogResourceKeys), Cognex.VisionPro.ToolGroup.Resources.CogResourceKeys.RkScritptAdapterMemberAccessError), this.mToolBlock.Name, "Tools.CogIDTool1"), ex);

}

}

}

public System.Collections.IEnumerator GetEnumerator()

{

return this.mToolBlock.Tools.GetEnumerator();

}

}

#endregion

#endregion