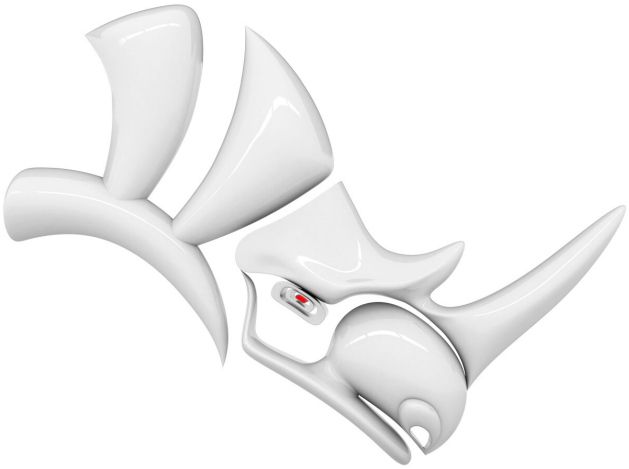
How To . . !

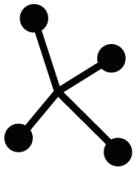


GH Algorithm into C# component

This technique for beginners in grasshopper

+

+



Diagonal Sorting Points Component

Written by : Architecte & Computational Designer

Maan Abdulkareem Akber - 2023 Parastorm lab founder

https://www.youtube.com/@parastormlab4866/videos

Preface

How can you build a C# component to use it inside Grasshopper as a tool and avoid repeating your definition over and over ?

This is an important question to manage your work and compress a huge Grasshopper algorithm and handle the problems by chopping and scaling it .
In this tiny document that I have done for other Grasshopper users, it can help them to understand what is going on inside Grasshopper or the backend processes, and learn more deeply about what a component is !

1- We'll take a bunch of Grasshopper components as a function and then convert all of them into a C# Grasshopper component using lines of code, analysis, evaluation, testing, and final component extraction .
2 -Then we will use .NET Programming Language in Visual Studio to generate a native Grasshopper algorithm component and put it wherever you want in any tab inside Grasshopper .

NEEDS :

- Grasshopper Scripting API .

  https://developer.rhino3d.com/api/grasshopper/html/723c01da-9986-4db2-8f53-6f3a7494df75.htm

- RhinoCommon API .

  https://developer.rhino3d.com/api/RhinoCommon/html/R_Project_RhinoCommon.htm

Grasshopper Algorithm

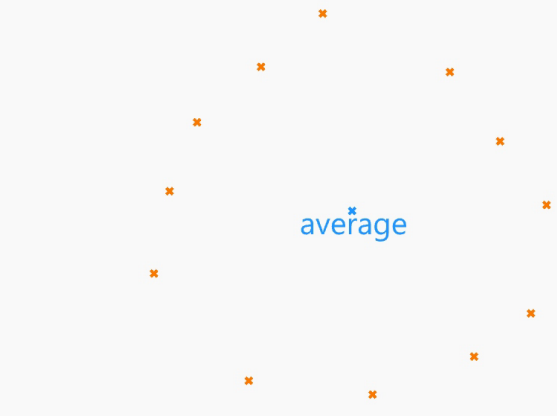
Let me take a problem and create grasshopper algorithm to get what i want to calc. as an example .

Problem :

I have a bunch of points in my definition and i have to sort all point by diagonal way to create a polyline , Nurbs curve or any shape needs sorted points and avoid distorsion that will happen if my points not in sorted way .

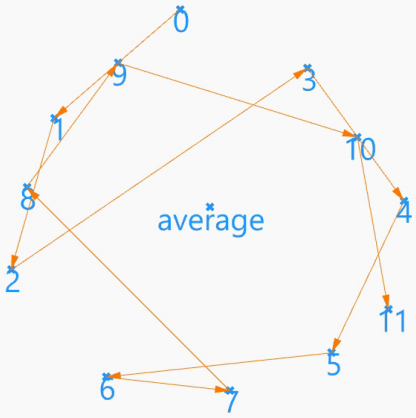
Solving :

- In Grasshopper you need to get an average point between the list of points .
- and get a vectors from average point to the list of points .



bunch of points

(random locations)



distorsion polyline

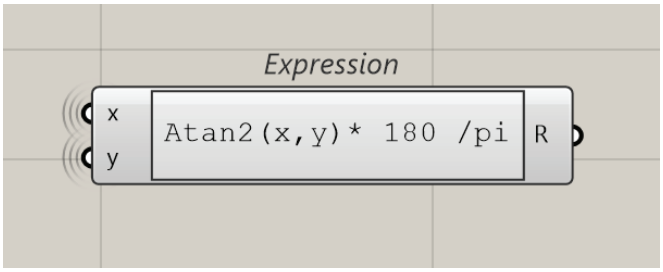
- To solve this small problem you have to follow a bit of steps :

a - first one you have to get a `vector` between an average point and the list of points.

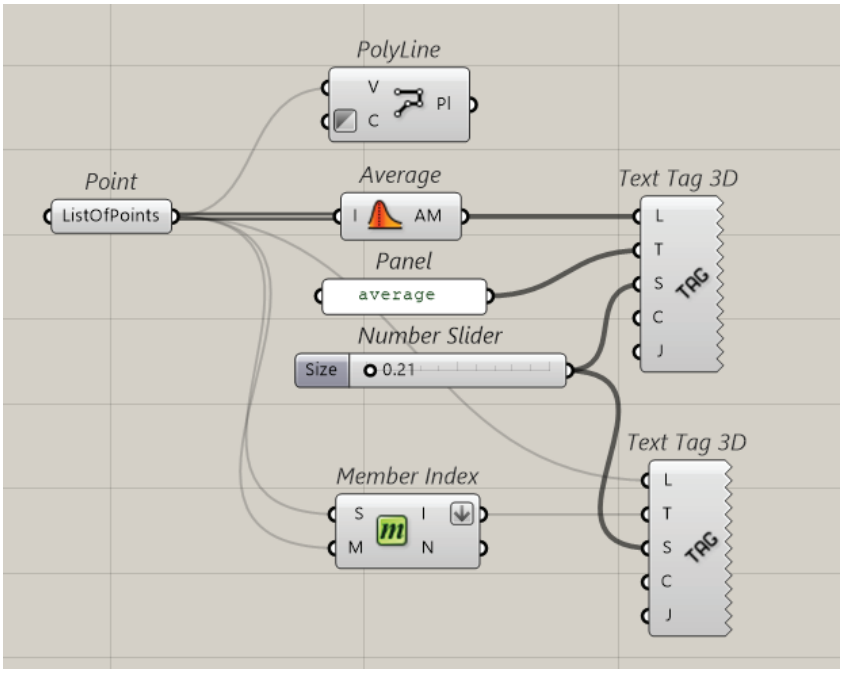
b - then you have to get (`x and y vector`) by deconstruct plane .

c - and third step you have to get a (`dot product`) , first between vector 2pt and (x - axe) from plane , and second one between vector 2pt and (y - axe) from plane .

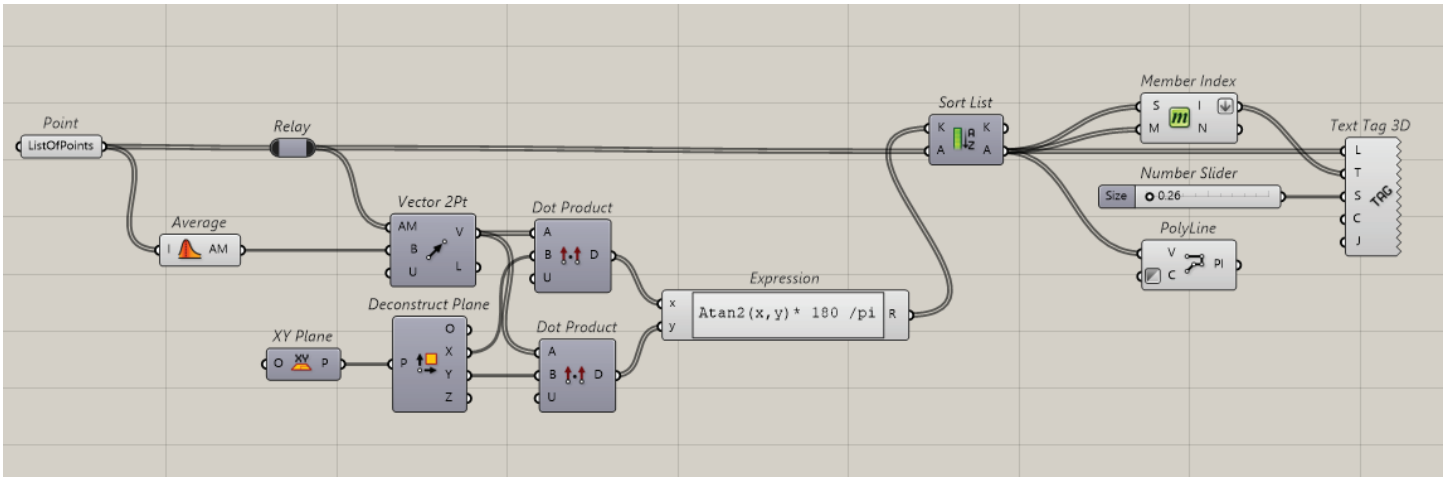
d - and last but not lest you have to use (`Atan2`) Math function and use it as a sorting keys . this step to get angles between dot product values . you can use (`Expression`)grasshopper math component



https://en.wikipedia.org/wiki/Atan2

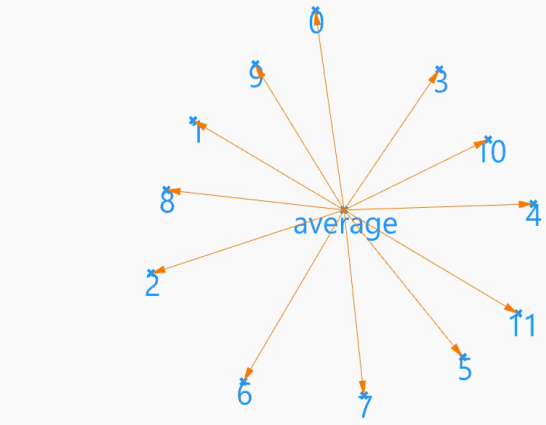


grasshopper definition

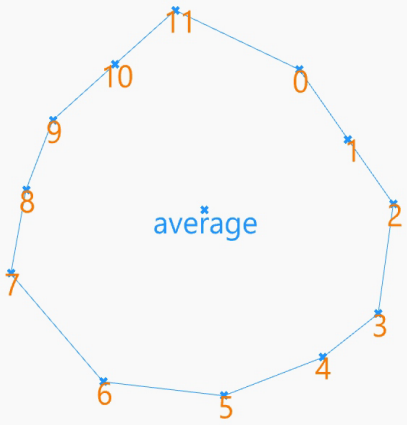


Final Grasshopper Algorithm

Final Result >>



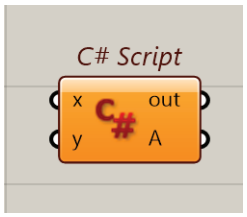
Get Vector 2Pt



Diagonal Sorted Points

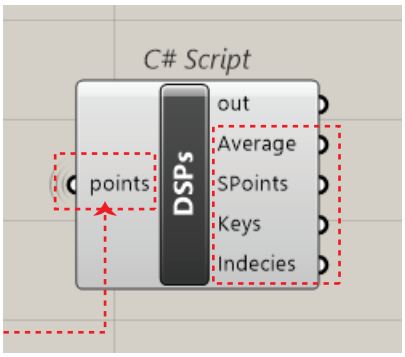
GH# Algorithm

let me do some C# script inside grasshopper to create an algorithm that contain a same previous steps to `Sort Points Diagonaly` to get the same function and use it when you want ..



step - 1

set an input name and data type by right - click onto points input name and chose `Point3d` as data type input and `List Access` , then set an outputs parts as you see here :



step - 2

open code editor to type some c# script inside and the first value we get from list of points is an `Average Point` . see here a code :

```
//------------------------------------------------------------------------------------------------------------------------
if(points.Count == 0)
{
  Component.AddRuntimeMessage(GH_RuntimeMessageLevel.Warning,
    "There are no data input yet!" + " Plase input a list of points");
  return;
}
else{
  // get an average point-----start
  Point3d avg = new Point3d();

  for(int i = 0; i < points.Count;i++)
  {
    avg += new Point3d((points[i].X / points.Count), (points[i].Y / points.Count), (points[i].Z / points.Count));
  }
  //------------------------------------------------------------------------------------------------------------------------
```

step - 3

in this step you have to compute `List Vector 2Pt` between an average and list of points from input , and then get a `Dot - Product` values to sort angles . angles values we will using its as a sorting keys in the next step .

```
//-------------------------------------------------------------------------------------------------------------

// compute victors btween 2points & dot-product values and get sorted angles!--------start

List<Vector3d> vectorsList = new List<Vector3d>();

List<double> angles = new List<double>();

for(int i = 0; i < points.Count;i++)
{
  Vector3d vector = points[i] - avg;
  vectorsList.Add(vector);
  double dx = Vector3d.Multiply(vectorsList[i], Plane.WorldXY.XAxis);
  double dy = Vector3d.Multiply(vectorsList[i], Plane.WorldXY.YAxis);
  double angle = Math.Atan2(dx, dy) * 180 / Math.PI;
  angles.Add(angle);
}
double[] key = angles.ToArray();
angles.Sort();
//-------------------------------------------------------------------------------------------------------------
```

step - 4

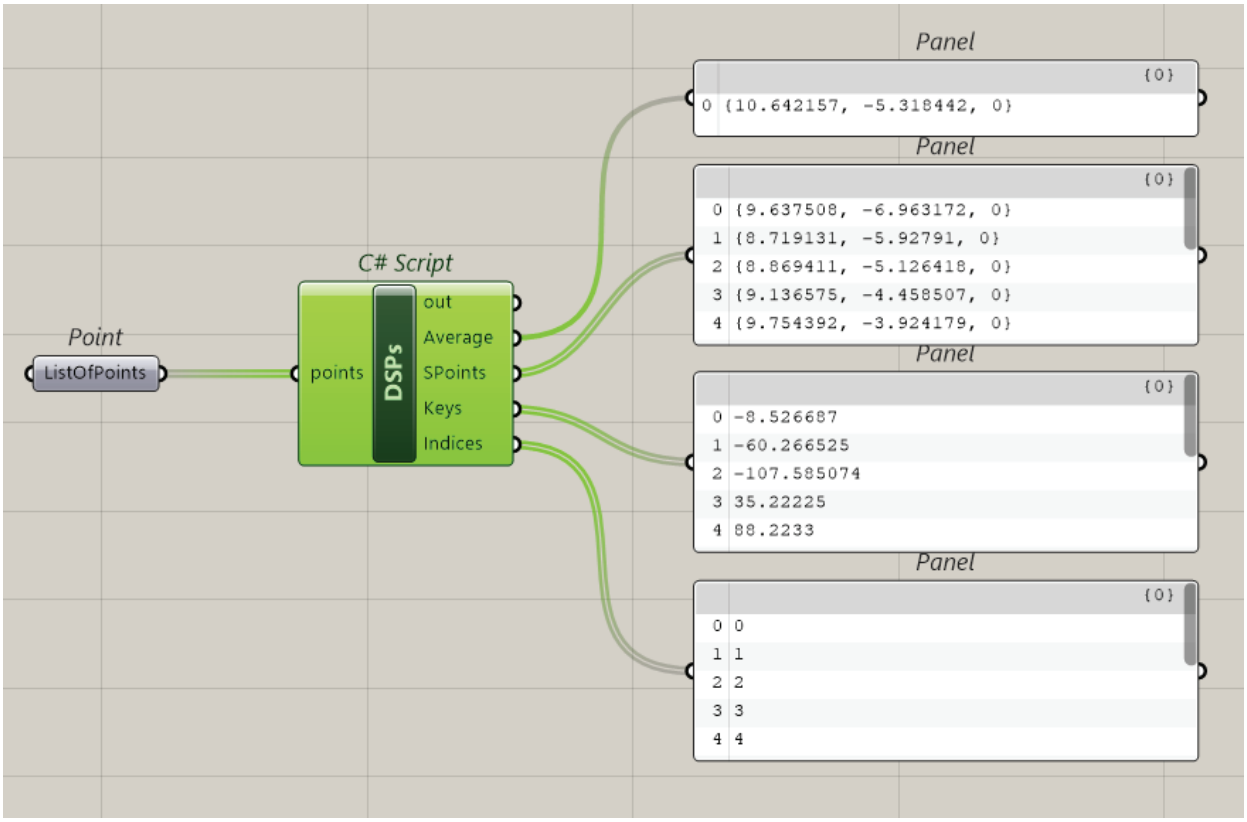
now you have to get (`x,y,z`) coordinates in separated list each and sorting these values to be ready to generate a `new List of points` and that what you want to do in the end of an algorithm .

```
//-------------------------------------------------------------------------------------------------------------

// get sorted data!--------start
RhinoList<double> sx = new RhinoList<double>();
RhinoList<double> sy = new RhinoList<double>();
RhinoList<double> sz = new RhinoList<double>();


double[] x0 = new double[points.Count];
double[] y0 = new double[points.Count];
double[] z0 = new double[points.Count];

for(int i = 0; i < points.Count;i++)
{
  Point3d xyzPoints = points[i];
  x0[i] = xyzPoints.X;
  y0[i] = xyzPoints.Y;
  z0[i] = xyzPoints.Z;

  sx.Add(xyzPoints.X);
  sy.Add(xyzPoints.Y);
  sz.Add(xyzPoints.Z);
}
sx.Sort(key);
sy.Sort(key);
sz.Sort(key);
//-------------------------------------------------------------------------------------------------------------
```

step - 5

Finaly generate a List of `Point3d` from new sorted keys and get `Item Index` for each point from `for loop` function then define output values .

```
  //------------------------------------------------------------------------------------------------------------------------

  // Create a sorted list of points from last data!-----start
  List<Point3d> pts = new List<Point3d>();
  List<int> indexNum = new List<int>();

  for(int i = 0; i < points.Count;i++)
  {
    Point3d pt = new Point3d(sx[i], sy[i], sz[i]);
    pts.Add(pt);
    indexNum.Add(i);
  }
  //------------------------------------------------------------------------------------------------------------------------

  // Outputs
  Average = avg;
  SPoints = pts;
  Keys = key;
  Indices = indexNum;
}
```



Diagonal Sorted Points

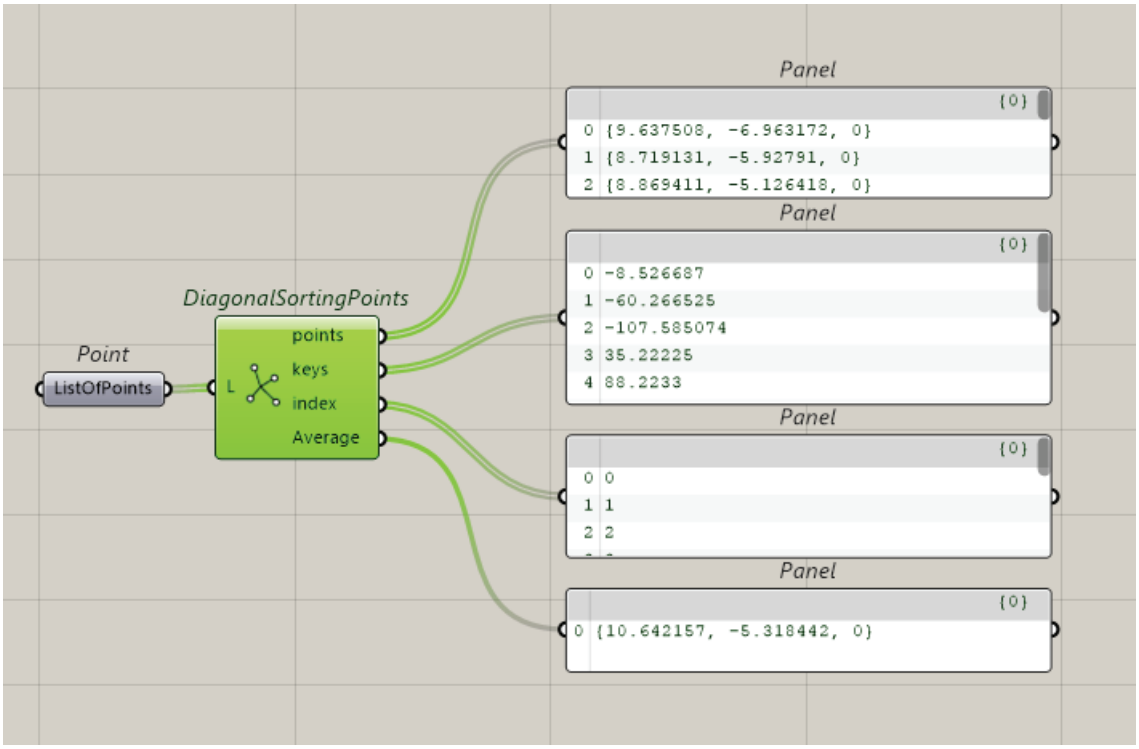
Final Results

```
//------------------------------------------------------------------------------------------------------------------
if(points.Count == 0)
{
  Component.AddRuntimeMessage(GH_RuntimeMessageLevel.Warning,
    "There are no data input yet!" + " Plase input a list of points");
  return;
}
else{
  // get an average point-----start
  Point3d avg = new Point3d();

  for(int i = 0; i < points.Count;i++)
  {
    avg += new Point3d((points[i].X / points.Count), (points[i].Y / points.Count), (points[i].Z / points.Count));
  }
  //-------------------------------------------------------------------------------------------------------------

  // compute victors btween 2points & dot-product values and get sorted angles!--------start

  List<Vector3d> vectorsList = new List<Vector3d>();

  List<double> angles = new List<double>();

  for(int i = 0; i < points.Count;i++)
  {
    Vector3d vector = points[i] - avg;
    vectorsList.Add(vector);
    double dx = Vector3d.Multiply(vectorsList[i], Plane.WorldXY.XAxis);
    double dy = Vector3d.Multiply(vectorsList[i], Plane.WorldXY.YAxis);
    double angle = Math.Atan2(dx, dy) * 180 / Math.PI;
    angles.Add(angle);
  }
  double[] key = angles.ToArray();
  angles.Sort();
  //-------------------------------------------------------------------------------------------------------------

  // get sorted data!--------start
  RhinoList<double> sx = new RhinoList<double>();
  RhinoList<double> sy = new RhinoList<double>();
  RhinoList<double> sz = new RhinoList<double>();


  double[] x0 = new double[points.Count];
  double[] y0 = new double[points.Count];
  double[] z0 = new double[points.Count];

  for(int i = 0; i < points.Count;i++)
  {
    Point3d xyzPoints = points[i];
    x0[i] = xyzPoints.X;
    y0[i] = xyzPoints.Y;
    z0[i] = xyzPoints.Z;

    sx.Add(xyzPoints.X);
    sy.Add(xyzPoints.Y);
    sz.Add(xyzPoints.Z);
  }
  sx.Sort(key);
  sy.Sort(key);
  sz.Sort(key);
  //-------------------------------------------------------------------------------------------------------------

  // Create a sorted list of points from last data!-----start
  List<Point3d> pts = new List<Point3d>();
  List<int> indexNum = new List<int>();

  for(int i = 0; i < points.Count;i++)
  {
    Point3d pt = new Point3d(sx[i], sy[i], sz[i]);
    pts.Add(pt);
    indexNum.Add(i);
  }
  //-------------------------------------------------------------------------------------------------------------

  // Outputs
  Average = avg;
  SPoints = pts;
  Keys = key;
  Indices = indexNum;
}
```

Visual Studio + C# Algorithm

In this approach we will know how can build a native grasshopper component use C# programming language as a developer to create own grasshopper plug-in and share it with other grasshopper users and more >>>

And we have token a previous example (diagonal sorting points)



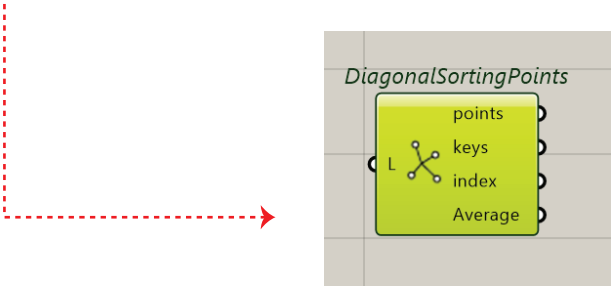
Diagonal Sorted Points Component

to start this task you have to install visual studio and install grasshopper assembly for rhino then create a new project to use GH_Component class that drive you to main templet to start your component development .
it is the same thing or the same lines of code these we done inside grasshopper component just an input and output are different a little bit just check my tiny code >>

```
50          /// constructor without any arguments.
51          /// Category represents the Tab in which the component will appear,
52          /// Subcategory the panel. If you use non-existing tab or panel names,
53          /// new tabs/panels will automatically be created.
54          /// </summary>
55          ///
56
57          // Component Written By @maan.arch - PARASTORM lab.
58          // Sometimes I Need A Component That Do Something Like Sort a List Of
              Points.
59          // And So, I Done This Code To Help Me & You (: . Try And Let Me
              Know !!
60          public DiagonalSortingPointsComponent()
61            : base("DiagonalSortingPoints", "DSP",
62              "Diagonal Sorting List of Points",
63              "Sets", "List")
64          {
65          }
66
67          /// <summary>
68          /// Registers all the input parameters for this component.
69          /// </summary>
70          protected override void RegisterInputParams
              (GH_Component.GH_InputParamManager pManager)
71          {
72              pManager.AddPointParameter("ListOfPoints", "L", "List of Point3d To
                 Sort Them Diagonally Like O'clock Wise",
73                  GH_ParamAccess.list);
74          }
75
76          /// <summary>
77          /// Registers all the output parameters for this component.
78          /// </summary>
79          protected override void RegisterOutputParams
              (GH_Component.GH_OutputParamManager pManager)
80          {
81              pManager.AddPointParameter("Points", "points", "Sorted Points",
                  GH_ParamAccess.list);
82              pManager.AddNumberParameter("Keys", "keys", "Sorted Keys",
                  GH_ParamAccess.list);
83              pManager.AddIntegerParameter("Index", "index", "Indices of Points",
                   GH_ParamAccess.list);
84              pManager.AddPointParameter("Average", "Average", "An Average
                  Point", GH_ParamAccess.item);
85          }
86
87          /// <summary>
88          /// This is the method that actually does the work.
89          /// </summary>
```

```
90          /// <param name="DA">The DA object can be used to retrieve data from
              input parameters and
91          /// to store data in output parameters.</param>
92          protected override void SolveInstance(IGH_DataAccess DA)
93          {
94              List<Point3d> pointsList = new List<Point3d>();
95              List<Point3d> ListP = new List<Point3d>();
96
97
98              if (!DA.GetDataList(0, ListP)) return;
99
100             //Algorithm
101
102             if (ListP.Count == 0)
103             {
104                 AddRuntimeMessage(GH_RuntimeMessageLevel.Warning, "There Are No
                      Data Yet > Please Input a List Of Points");
105                 return;
106             }
107             else
108             {
109                 Point3d avg = new Point3d();
110
111                 // Get An Average Point Between A List Of Points (Start)-------
                    >
112                 for (int i = 0; i < ListP.Count; i++)
113                 {
114                     if (ListP.Count == 1 || ListP.Count == 0)
115                     {
116                         AddRuntimeMessage(GH_RuntimeMessageLevel.Warning,
                          "you're not able to get a result from less than 2 points");
117                         return;
118
119                     }
120                     else
121                     {
122                          avg += new Point3d((ListP[i].X / ListP.Count), (ListP
                          [i].Y / ListP.Count), (ListP[i].Z / ListP.Count));
123
124                     }
125
126                 }
127
128                 // Get An Average Point Between A List Of Points (End)---------
                    >
129
130                 // compute victors btween 2points & dot-product values and get
                    sorted angles!--------start
131
```

```
132                 List<Vector3d> Vectors = new List<Vector3d>();
133                 List<double> angles = new List<double>();
134
135                 for (int i = 0; i < ListP.Count; i++)
136                 {
137                     Vector3d vecs = ListP[i] - avg;
138                     Vectors.Add(vecs);
139                     double Dx = Vector3d.Multiply(Vectors[i],
                        Plane.WorldXY.XAxis);
140                     double Dy = Vector3d.Multiply(Vectors[i],
                        Plane.WorldXY.YAxis);
141                     // compute victors btween 2points & dot-product values and
                        get sorted angles!--------end
142
143                     // Compute Angles (keys)
                        (Start)------------------------------------>
144
145                     double a = Math.Atan2(Dx, Dy);
146                     double angle = a * (180 / Math.PI);
147                     // Compute Angles (keys)
                        (End)------------------------------------->
148
149                     angles.Add(angle);
150
151                 }
152
153                 // Compute a Dot Product Values of 2Vectors
                      (End)------------------>
154
155                 // Sorting DataKeys
                      (Start)---------------------------------------->
156
157                 double[] k = angles.ToArray();
158                 angles.Sort();
159
160                 // Sorting DataKeys
                      (End)------------------------------------------>
161
162                 // Sorting Data
                      (Start)------------------------------------------->
163                 RhinoList<double> sortPointsX = new RhinoList<double>();
164                 RhinoList<double> sortPointsY = new RhinoList<double>();
165                 RhinoList<double> sortPointsZ = new RhinoList<double>();
166
167                 double[] X0 = new double[ListP.Count];
168                 double[] Y0 = new double[ListP.Count];
169                 double[] Z0 = new double[ListP.Count];
170
171
```



```
172                 for (int i = 0; i < ListP.Count; i++)
173                 {
174
175                     Point3d pxyz = ListP[i];
176                     X0[i] = pxyz.X;
177                     sortPointsX.Add(pxyz.X);
178                     Y0[i] = pxyz.Y;
179                     sortPointsY.Add(pxyz.Y);
180                     Z0[i] = pxyz.Z;
181                     sortPointsZ.Add(pxyz.Z);
182
183                 }
184
185                 sortPointsX.Sort(k);
186                 sortPointsY.Sort(k);
187                 sortPointsZ.Sort(k);
188
189                 List<int> Pointsindices = new List<int>();
190
191                 for (int i = 0; i < ListP.Count; i++)
192                 {
193                     Point3d PL = new Point3d(sortPointsX[i], sortPointsY[i],
                        sortPointsZ[i]);
194
195                     Pointsindices.Add(i);
196                     pointsList.Add(PL);
197
198                 }
199
200                 // Sorting Data
                      (End)------------------------------------------------>
201
202                 //output
203                 DA.SetDataList(0, pointsList);
204                 DA.SetDataList(1, k);
205                 DA.SetDataList(2, Pointsindices);
206                 DA.SetData(3, avg);
207
208
209             }
210
211         }
212         /// <summary>
213         /// Provides an Icon for every component that will be visible in the
              User Interface.
214         /// Icons need to be 24x24 pixels.
215         /// You can add image files to your project resources and access them
              like this:
216         /// return Resources.IconForThisComponent;
```