# Visualizing the convex hull in two and three dimensions

From

*Essentials of Programming in Mathematica*

Exercises from Chapter 8, Graphics and visualization



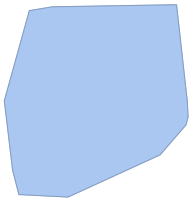
## 8.3 Efficient structures

1. A common problem in computational geometry is finding the boundary of a given set of points. One way to think about the boundary is to imagine the points as nails in a board and then to stretch a rubber band around all the nails. The stretched rubber band lies on a convex polygon commonly called the *convex hull* of the point set.

   Create a function `ConvexHullPlot` for visualizing the convex hull together with the points on the interior of the convex hull. Your function should inherit options for `Graphics`. The built-in function `ConvexHullMesh` can be used to generate the hull polygon:

```
In[2]:= pts = RandomReal[1, {28, 2}];
        ℜ = ConvexHullMesh[pts]
```

Out[3]=



   The zero-dimensional objects in the mesh are points and the one-dimensional objects are lines (two-dimensional objects would be polygons in the plane). These are the points and lines on the convex hull.

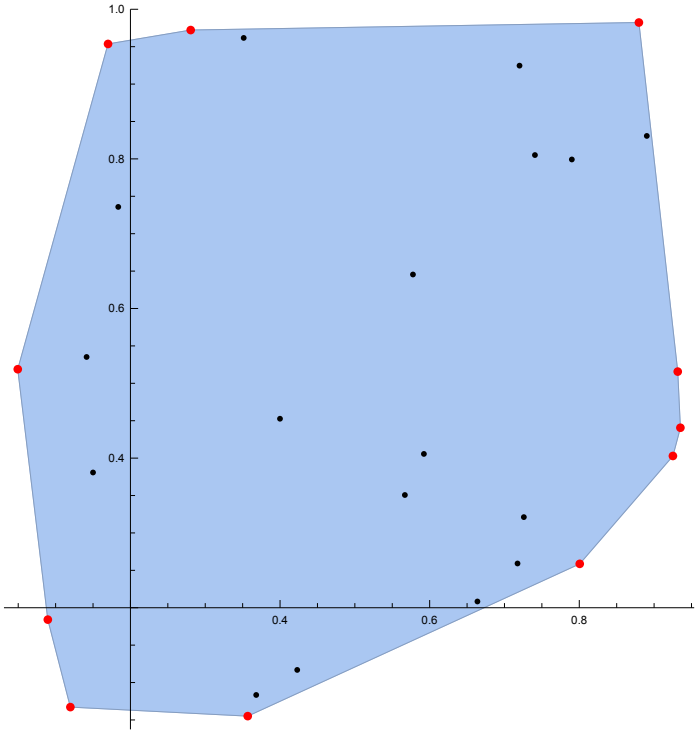
```
In[4]:= MeshPrimitives[ℜ, 0];
        Take[%, 2]
Out[5]= {Point[{0.0493875, 0.518835}], Point[{0.356808, 0.0551065}]}

In[6]:= MeshPrimitives[ℜ, 1];
        Take[%, 2]
Out[7]= {Line[{{0.0493875, 0.518835}, {0.0895672, 0.184101}}],
         Line[{{0.0895672, 0.184101}, {0.119729, 0.0671685}}]}
```

   Your function should generate output similar to the following:

In[8]:=
```
ConvexHullPlot[pts, Axes → Automatic]
```

Out[8]=




*Solution*

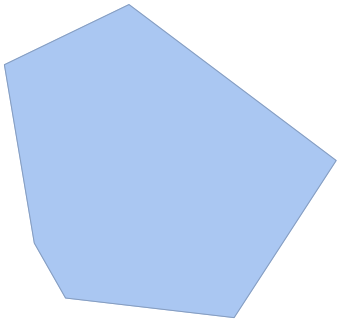
. First, generate some random points in the plane.

In[9]:=
```
pts = RandomReal[1, {8, 2}];
```

Next, generate the mesh using a built-in function.

In[10]:=
```
ℜ = ConvexHullMesh[pts]
```

Out[10]=



Quite a bit of information is embedded in the this `BoundaryMeshRegion`. For example, it is easy to extract the points on the boundary. They are mesh primitives of dimension zero.

In[11]:=
```
Head[ℜ]
```

Out[11]=
```
BoundaryMeshRegion
```

In[12]:=
```
MeshPrimitives[ℜ, 0]
```

Out[12]=
```
{Point[{0.691858, 0.095207}], Point[{0.13393, 0.303382}], Point[{0.398292, 0.969369}],
 Point[{0.2212, 0.14994}], Point[{0.976876, 0.533907}], Point[{0.0503764, 0.801421}]}
```

Similarly, lines are mesh primitives of dimension one.

In[13]:=
```
MeshPrimitives[ℜ, 1]
```

Out[13]=
```
{Line[{{0.691858, 0.095207}, {0.976876, 0.533907}}], Line[{{0.976876, 0.533907}, {0.398292, 0.969369}}],
 Line[{{0.398292, 0.969369}, {0.0503764, 0.801421}}], Line[{{0.0503764, 0.801421}, {0.13393, 0.303382}}],
 Line[{{0.13393, 0.303382}, {0.2212, 0.14994}}], Line[{{0.2212, 0.14994}, {0.691858, 0.095207}}]}
```

In fact, the polygon itself is a mesh primitive of dimension two.

```
In[14]:= MeshPrimitives[ℛ, 2]
Out[14]= {Polygon[{{0.691858, 0.095207}, {0.976876, 0.533907},
    {0.398292, 0.969369}, {0.0503764, 0.801421}, {0.13393, 0.303382}, {0.2212, 0.14994}}]}
```

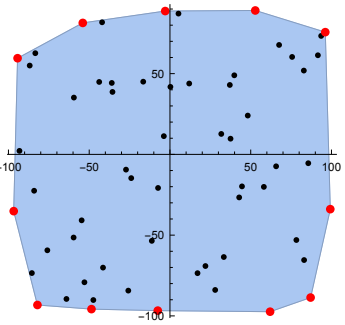
So the pieces of the mesh that we want to show are the region itself (bounded polygon, the original points, and the points on the boundary. We will highlight the points on the boundary by making them large and red. And we will pass options from `Graphics` on to the function.

```
In[15]:= ConvexHullPlot[pts_, opts : OptionsPattern[Graphics]] := Module[{ℛ}, ℛ = ConvexHullMesh[pts];
           Show[ℛ, Graphics[{Point[pts], Red, PointSize[Medium], MeshPrimitives[ℛ, 0]}], opts]]
```

Try it out with a larger point set.

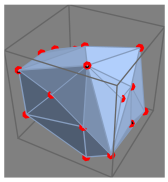
```
In[16]:= pts = RandomReal[{-100, 100}, {60, 2}];
In[17]:= ConvexHullPlot[pts, Axes → Automatic, ImageSize → Small]
```

Out[17]=




2. Modify `ConvexHullPlot` from the previous exercise to accept an option `Dimension`. With default value of two, `ConvexHullPlot` should produce output like that above. For `Dimension → 3`, it should generate the three-dimensional convex hull together with large red points at each vertex of the hull.

Figure 8.1. *Three-dimensional convex hull with hull points highlighted.*



## *Solution*

. First, set up the options framework.

```
In[18]:= Options[ConvexHullPlot] = Join[{Dimension → 2}, Options[Graphics], Options[Graphics3D]];
```

Here is a message that will be issued if the `Dimension` option is set to anything other than 2 or 3.

```
In[19]:= ConvexHullPlot::baddim = "The value `1` of the Dimension option should be either 2 or 3";
In[20]:= ConvexHullPlot[pts_, opts : OptionsPattern[]] := Module[{ℛ, dim = OptionValue[Dimension]},
           ℛ = ConvexHullMesh[pts];
           Which[
            dim == 2, Show[{ℛ, Graphics[{Point[pts], Red, PointSize[Medium], MeshPrimitives[ℛ, 0]}]},
             FilterRules[{opts}, Options[Graphics]]],
            dim == 3, Show[{ℛ, Graphics3D[{Point[pts], Red, PointSize[Medium], MeshPrimitives[ℛ, 0]}]},
             FilterRules[{opts}, Options[Graphics3D]]],
            True, Message[ConvexHullPlot::baddim, dim]
           ]
          ]
```

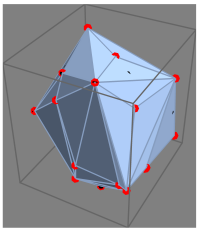
Here is a larger point set.

```
In[21]:= pts = RandomReal[1, {40, 3}];
```

Exercise some options.

In[22]:=
```
ConvexHullPlot[pts, Dimension → 3, Boxed → True, Background → Gray]
```

Out[22]=



Giving a bad value for the `Dimension` option should produce an error message.

In[23]:=
```
ConvexHullPlot[pts, Dimension → 4]
```

ConvexHullPlot: The value 4 of the Dimension option should be either 2 or 3