```
<style>
body {
   height:100%;
}

html {
   height: 100%;
}
  svg {
      position: absolute;
      top: 0;
   }
   circle {
      stroke-width: 1.5px;
   }
</style>
<body>



<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
</body>
<script src="https://d3js.org/d3.v4.js"></script>
<script src="https://measure-fps.surge.sh/measureFPS.js">

</script>
<script src="https://intervis-projects.ccs.neu.edu/ssvg/ssvg-auto.js"></script>



<script>



var coordinates;
 var clicked_id="Isovist Area";
  var clickable=[];
  var mapName;
  var message;
  var attribute=clicked_id;
  var nowMilliseconds = new Date();
  var milliseconds=nowMilliseconds.getMilliseconds();
  var map="Goldberg";
  var  colorId=["red","orange","yellow","lime","lawngreen","cyan","dodgerblue","navy"];
    var valData=[];
 var id=clicked_id;

mapName="Csv/"+map+".csv";


var margin = {top: 10, right: 0, bottom: 0, left: 0},
   width = 2550 - margin.left - margin.right,
   height = 1550 - margin.top - margin.bottom;
```

```
var svg = d3.select("body")
  .append("svg")
.attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
     .attr("transform",
          "translate(" + margin.left + "," + margin.top + ")");

d3.csv(mapName, function(data) {


var IsovistArea=[];
   var i=0;
   var xVal=[];
   var yVal=[];
   id="Isovist Area";


   for(i=0;i<data.length;i++)
   {
  var Index=[];
  Index=data[i][id];
  IsovistArea.push(Index);
  Index=parseFloat(data[i]["x"]);
  xVal.push(Index);
  Index=parseFloat(data[i]["y"]);
  yVal.push(Index);



   }


data.forEach(function(d) {

 d.IsovistArea = +(d[id]);
 });
 var max = d3.max(data, function(d) { return d.IsovistArea; });
 var min = d3.min(data, function(d) { return d.IsovistArea; });
    var xMax = d3.max(xVal);
 var xMin = d3.min(xVal);
 var yMax = d3.max(yVal);
 var yMin = d3.min(yVal);
 var yPoint;
 var val=min+max/3;


 if(yMax>xMax)
 {
  yPoint=yMax+yMax/5;
 }
 else{
```

```
 yPoint=xMax+xMax/2;
}




 // Add X axis
 var x = d3.scaleLinear()
   .domain([xMin-xMin/2, xMax+xMax/2])
   .range([ 0, width ]);


 // Add Y axis
 var y = d3.scaleLinear()
   .domain([yMin-yMin/5, yPoint])
   .range([ height, 0]);



 var mapped = IsovistArea.map(function(el, i) {
   return { index: i, value: el };
})

// sorting the mapped array containing the reduced values
mapped.sort(function(a, b) {
   return a.value - b.value;
});

// container for the resulting order
var result = mapped.map(function(el){
   return IsovistArea[el.index];
});

for(let i=0;i<10;i++)
{
var a=[];
a=Object.values(mapped[i]);
//console.log(a[0]);

}

 function color(mapped1)
 {
 var color=[]
 var i

 for(i=0;i<mapped1.length;i++)
 {
  var a=[];
  a=Object.values(mapped1[i]);


  if (i>0 &&  i<(Math.floor(mapped1.length/8)))
```

```
 {
  color[a[0]]="navy";
  valData[a[0]]="Very Low";
 }
 else if (i>=(Math.floor(mapped1.length/8)) && i<(Math.floor(mapped1.length/4)))
 {
 color[a[0]]="dodgerblue";
 valData[a[0]]="Low";
 }
 else if (i>=(Math.floor(mapped1.length/4)) && i<(3*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="cyan";
 valData[a[0]]="Low";
 }
 else if (i>=(3*Math.floor(mapped1.length/8)) && i<(4*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="lawngreen";
 valData[a[0]]="Medium";
 }
 else if (i>=(4*Math.floor(mapped1.length/8)) && i<(5*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="lime";
 valData[a[0]]="Medium";
 }
 else if (i>=(5*Math.floor(mapped1.length/8)) && i<(6*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="yellow";
 valData[a[0]]="High";
 }
 else if (i>=(6*Math.floor(mapped1.length/8)) && i<(7*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="orange";
 valData[a[0]]="High";
 }
 else if (i>=(7*Math.floor(mapped1.length/8)) && i<(Math.floor(mapped1.length)))
 {
 color[a[0]]="red";
 valData[a[0]]="Very High";
 }


}
return color;
}


colorData=color(mapped)
 // Create the scatter variable: where both the circles and the brush take place
 var scatter = svg.append('g')


 // Add circles
var circle=  scatter
   .selectAll("circle")
```

```
    .data(data)
    .enter()
    .append("circle")
      .attr("cx", function (d) { return x(d.x); } )
      .attr("cy", function (d) { return y(d.y); } )
      .attr("r", 6)
      .attr("id", function(d,i)
  {

   return String(colorData[i])
  }
  )
  .attr("fill", function(d,i)
  {

   return String(colorData[i])
  }
  )




});
</script>
<script>
    const ws =new WebSocket("ws://192.168.0.68:8080");


 var ParameterVal;


 ws.addEventListener("open",function(event){

  var colorId=["red","orange","yellow","lime","lawngreen","cyan","dodgerblue","navy"];
  ws.addEventListener('message', function (event) {
     message=event.data;
  console.log(message);
  var received="Received";



if(!(typeof message === "undefined"))
{
console.log(`${received}\n`);
if(message.includes("Parameter 1 Around &"))
{
ParameterVal=message.split("&");
map=ParameterVal[1];
mapName="Csv/"+map+".csv";
}
}
```

```
// Attribute
if(message)
{
if(!(typeof message === "undefined"))
{
d3.csv(mapName, function(data) {
if(message.includes("Parameter 1 Around &"))
{
ParameterVal=message.split("&");

if(!(id=== null))
{

if(!(id===ParameterVal[1] ))
{

    var IsovistArea=[];
   var i=0;
   var xVal=[];
   var yVal=[];
   id=ParameterVal[2];

   for(i=0;i<data.length;i++)
   {
  var Index=[];
  Index=data[i][ParameterVal[2]];
  IsovistArea.push(Index);
  Index=parseFloat(data[i]["x"]);
  xVal.push(Index);
  Index=parseFloat(data[i]["y"]);
  yVal.push(Index);



   }


data.forEach(function(d) {

 d.IsovistArea = +(d[attribute]);
 });
 var max = d3.max(data, function(d) { return d.IsovistArea; });
 var min = d3.min(data, function(d) { return d.IsovistArea; });
    var xMax = d3.max(xVal);
 var xMin = d3.min(xVal);
 var yMax = d3.max(yVal);
 var yMin = d3.min(yVal);
 var yPoint;

 var val=min+max/3;


 if(yMax>xMax)
 {
```

```
 yPoint=yMax+yMax/5;
}
else{
 yPoint=xMax+xMax/2;
}




 // Add X axis



var mapped = IsovistArea.map(function(el, i) {
   return { index: i, value: el };
})

// sorting the mapped array containing the reduced values
mapped.sort(function(a, b) {
   return a.value - b.value;
});

// container for the resulting order
var result = mapped.map(function(el){
   return IsovistArea[el.index];
});

for(let i=0;i<10;i++)
{
var a=[];
a=Object.values(mapped[i]);
//console.log(a[0]);

}

function color(mapped1)
{
var color=[]
var i

for(i=0;i<mapped1.length;i++)
{
 var a=[];
 a=Object.values(mapped1[i]);



 if (i>0 &&  i<(Math.floor(mapped1.length/8)))
 {
  color[a[0]]="navy";
  valData[a[0]]="Very Low";
 }
 else if (i>=(Math.floor(mapped1.length/8)) && i<(Math.floor(mapped1.length/4)))
```

```
 {
 color[a[0]]="dodgerblue";
 valData[a[0]]="Low";
 }
 else if (i>=(Math.floor(mapped1.length/4)) && i<(3*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="cyan";
 valData[a[0]]="Low";
 }
 else if (i>=(3*Math.floor(mapped1.length/8)) && i<(4*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="lawngreen";
 valData[a[0]]="Medium";
 }
 else if (i>=(4*Math.floor(mapped1.length/8)) && i<(5*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="lime";
 valData[a[0]]="Medium";
 }
 else if (i>=(5*Math.floor(mapped1.length/8)) && i<(6*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="yellow";
 valData[a[0]]="High";
 }
 else if (i>=(6*Math.floor(mapped1.length/8)) && i<(7*Math.floor(mapped1.length/8)))
 {
 color[a[0]]="orange";
 valData[a[0]]="High";
 }
 else if (i>=(7*Math.floor(mapped1.length/8)) && i<(Math.floor(mapped1.length)))
 {
 color[a[0]]="red";
 valData[a[0]]="Very High";
 }


}
return color;
}


colorData=color(mapped)
```

```
  // Add a clipPath: everything out of this area won't be drawn.


  // Create the scatter variable: where both the circles and the brush take place
  d3
    .selectAll("circle")
    .data(data)
   .attr("id", function(d,i)
   {

   return String(colorData[i])
   }
   )
   .attr("fill", function(d,i)
   {

   return String(colorData[i])
   }
   )

   .on("click",function(d,i)
   {


   });




}
}
}
if(message.includes("%"))
{

  var res = message.split("%");
 //Change the color of the scatter plot
   var color=[]
d3.selectAll("circle")
       .data(data)
       //.attr("cx", function (d) { return x(d.x); } )
       //.attr("cy", function (d) { return y(d.y); } )
    .attr("fill", function(d,i)
    {

    var filter=String(res[4]);
       if(valData[i]==filter)
    {
    color[i]=String(colorData[i])
    }
    else{
    color[i]="#8B4513";
    }
    if((parseInt(d.x)>=parseInt(res[0]) && parseInt(d.x)<=parseInt(res[1])) &&(parseInt(d.y)>=parseInt(res[2]) &&
```

```
parseInt(d.y)<=parseInt(res[3]))){

    color[i]="black";

   }
   return color[i];
   });

   }

   })




}

}


 }
 );

     });


</script>
```