ICT in SES

Geometric visualization

Lesson №20

Goals and issues

Goals

Visualization of geometrical objects

- Coordinate system
- Segment and vector
- Sphere, cone and cylinder
- Torus

Issues

Geometry and computer graphics

- A lot of common objects
- Different visualizations
- In educational context some traditional visualizations in computer graphics are not convenient

In this lesson

- Some examples of visualizations
- Examples for learning, not for copy-and-pasting

Coordinate system

Coordinate system

Default visualization

- With command oxyz

Missing

- Long axes
- Arrows
- Grid marks
- Axes in negative directions
- Labels, etc.

Arrows

Coordinate system arrows

- Three styles:

  black – defines the black colour

  ox – orients object as axis X

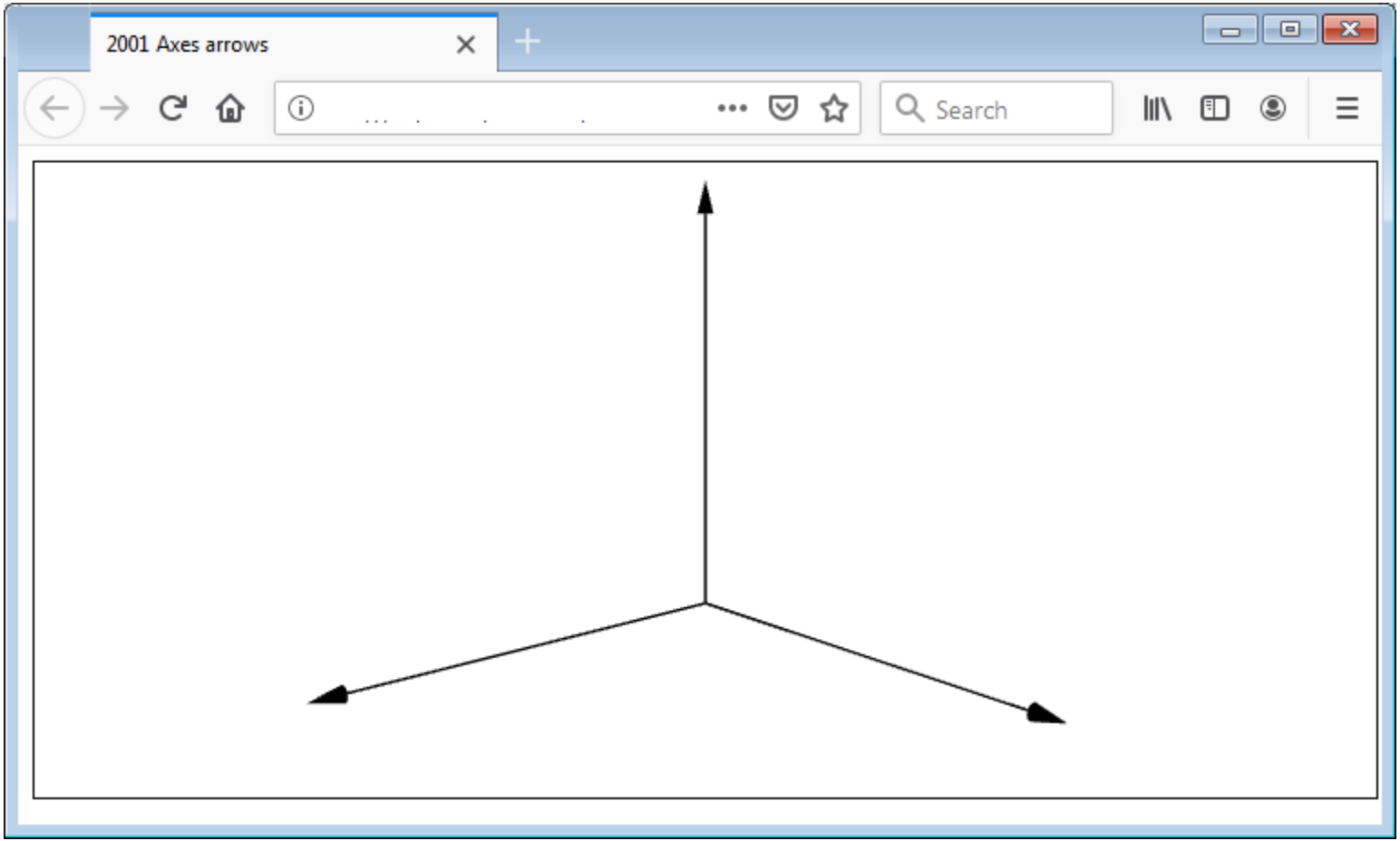
  oy – orients object as axis Y

```
black = {color:[0,0,0]};

ox = {focus:[1,0,0]};
oy = {focus:[0,1,0]};
```

Objects

- Three axes as segments of given length
- Arrows are black unlit cones
- Axes (and arrows) orientations are styled with ox and oy

```
s = segment([0,0,0],[0,0,50]).custom(black);
sameAs(s).custom(ox);
sameAs(s).custom(oy);

cone([0,0,50],1,4).custom(black);
cone([50,0,0],1,4).custom(black).custom(ox);
cone([0,50,0],1,4).custom(black).custom(oy);
```



TRY IT

All axes

Positive and negative axes

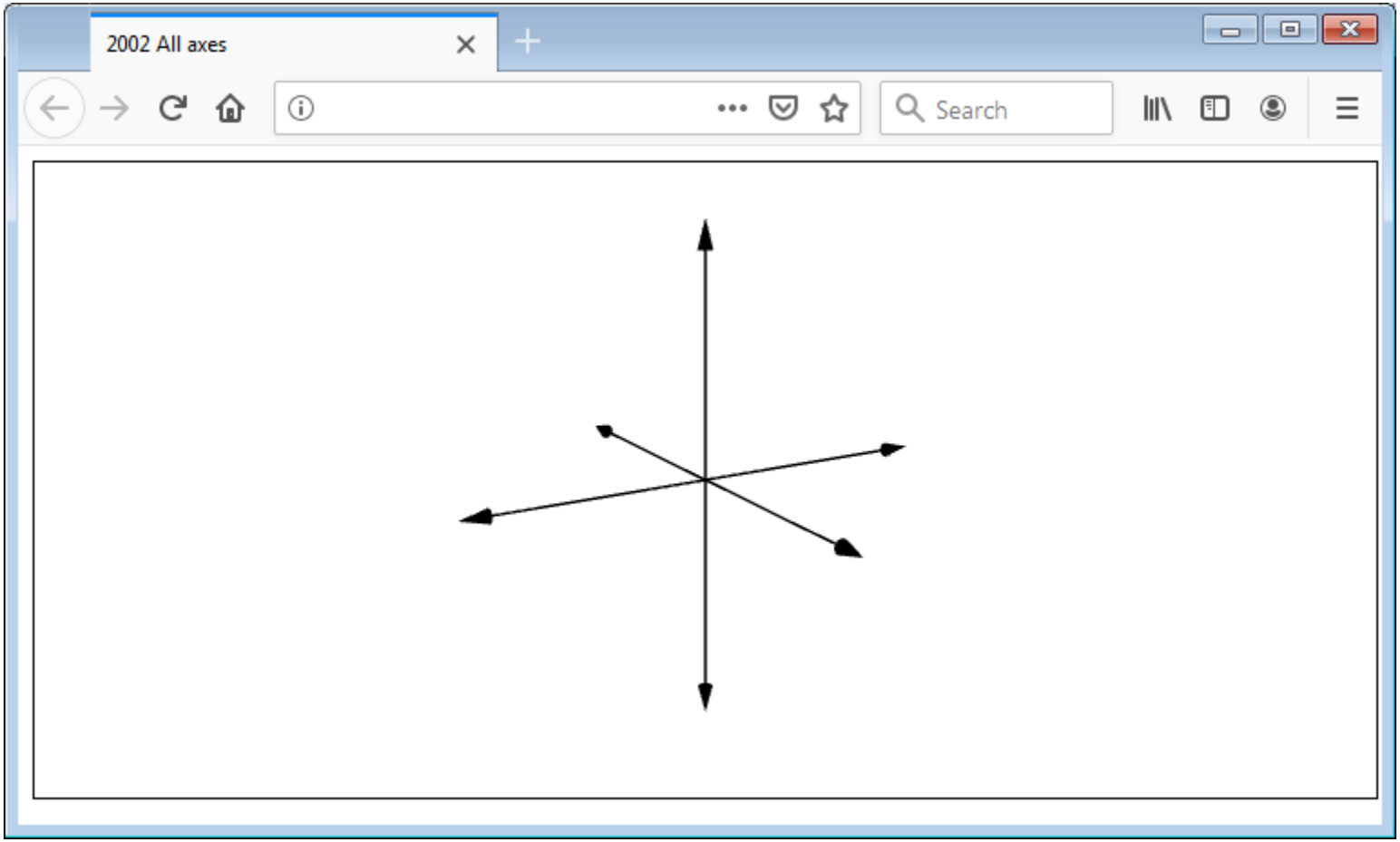
- Alternative solution (because of several same objects)
- Creating a group object
- The group has black colour

```
g = group([
      segment([0,0,0],[0,0,30]),
      cone([0,0,30],1,4)
   ]);
g.color = [0,0,0];
g.mergeColor();
```

Creating axes

- Cloning 5 times the vertical axis
- Changing groups orientations, so that axes point to their directions

```
sameAs(g).custom({focus:[1,0,0]});
sameAs(g).custom({focus:[0,1,0]});

sameAs(g).custom({focus:[-1,0,0]});
sameAs(g).custom({focus:[0,-1,0]});
sameAs(g).custom({focus:[0,0,-1]});
```



TRY IT

Axes with labels

Labels with the name of axis

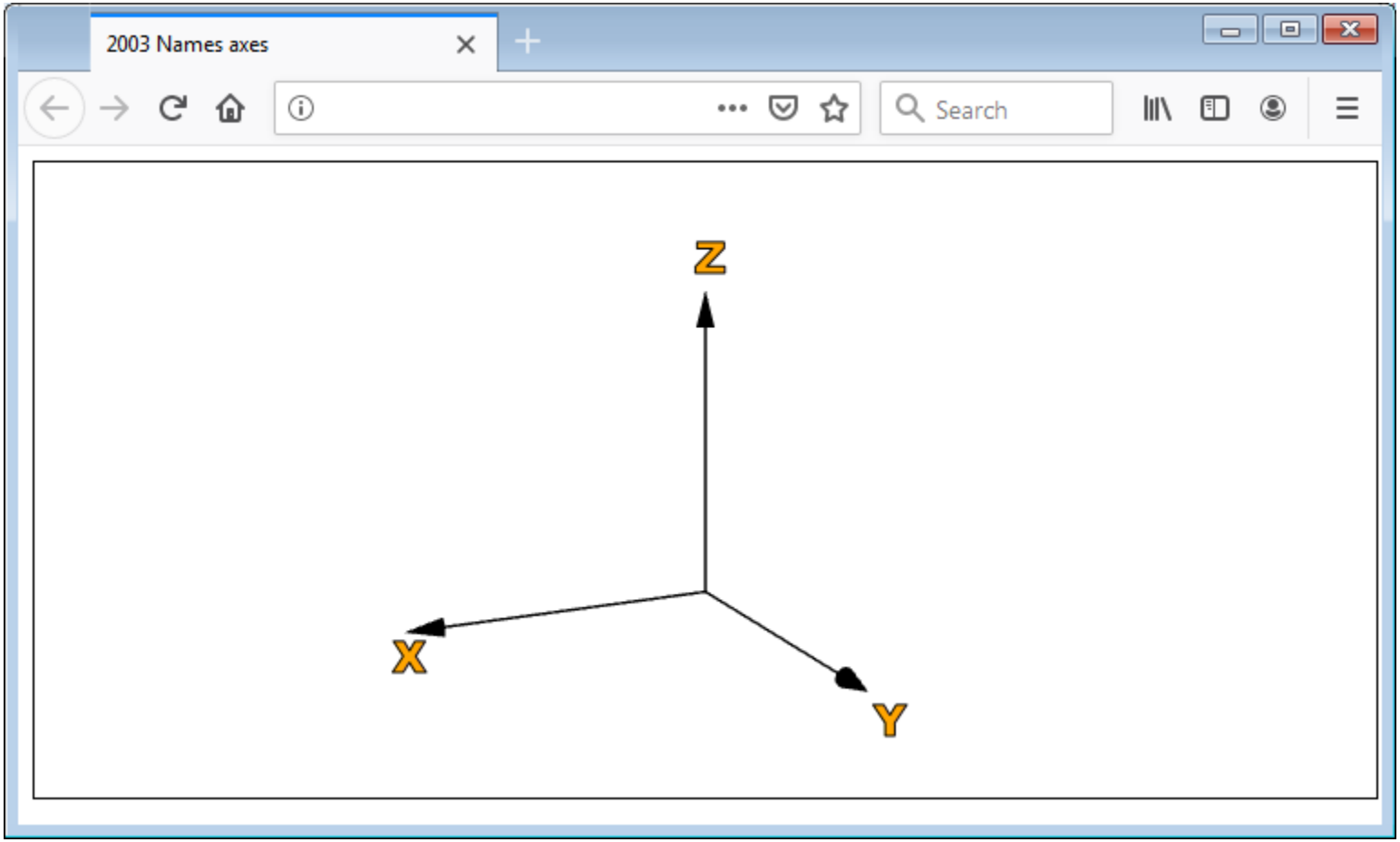
- The label is HTML element with CSS formatting
- Absolute position and a vertical index to position the label above the canvas

```
.label {
   position: absolute;
   z-index: 10;
   background-color: transparent;
   :
}
```

Positioning

- For axis X – finding screen position of 3D point [35,0,0] with function getPosition
- Calculated position is in pixels
- Transferring position to style.left and style.top (+ suffic “px”)
- The other axes are done in the same manner

```
var e = document.getElementById('x');

var pos = getPosition([35,0,0]);

e.style.left = pos[0]+"px";
e.style.top = pos[1]+"px";
```



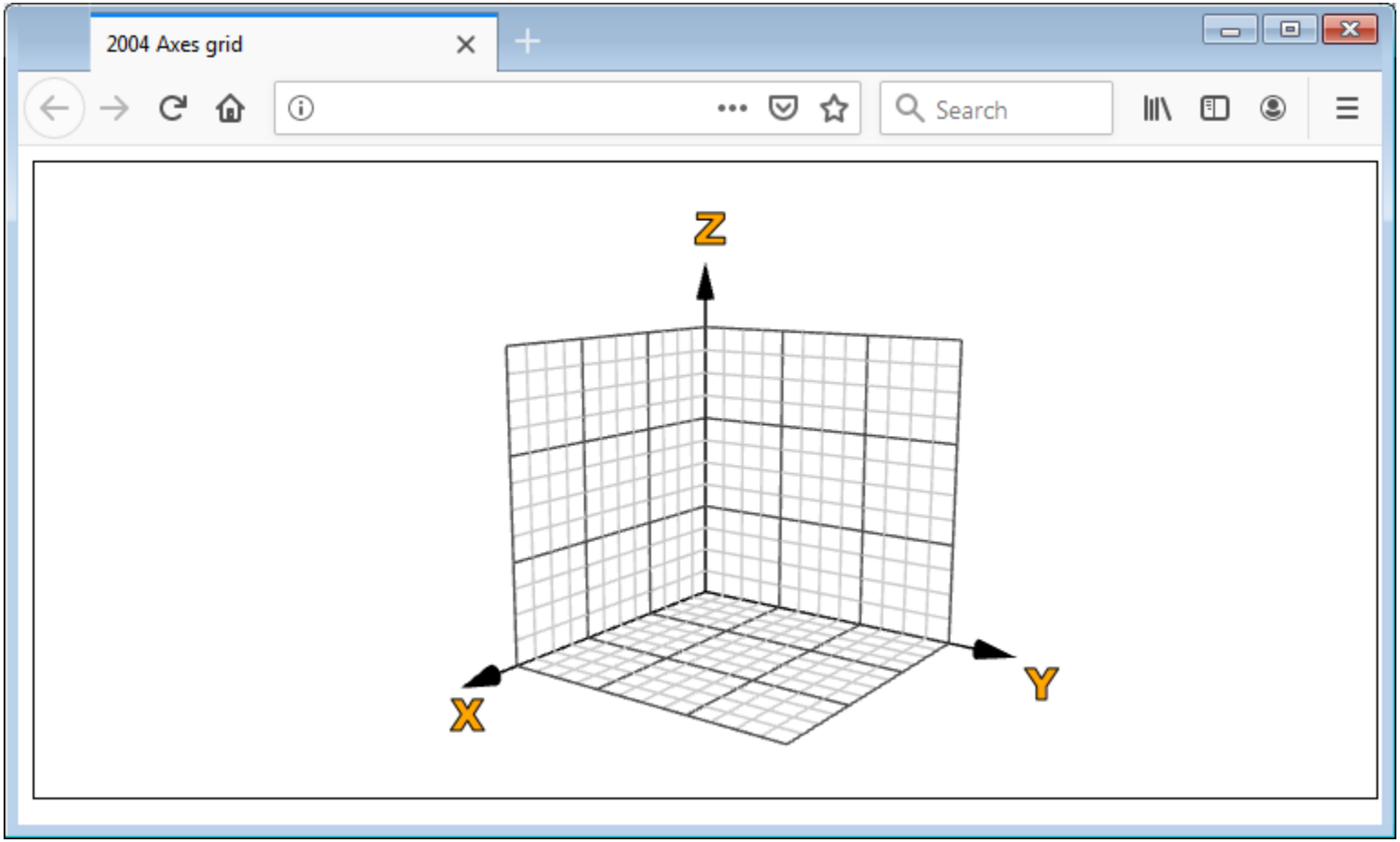
TRY IT

Coordinate grid

A grid of segments

- Generating two groups of segments in plane XY: one group with segment parallel to X axis, and another one – to Y axis
- Similar segments in the other planes

```
for (var i=0; i<=30; i+=2.5)
{
   segment( [i,0,0], [i,30,0]);
   segment( [0,i,0], [30,i,0]);
   :
}
```



TRY IT

Segment and vector

Segment and vector

Default visualization

- With segment

Missing elements

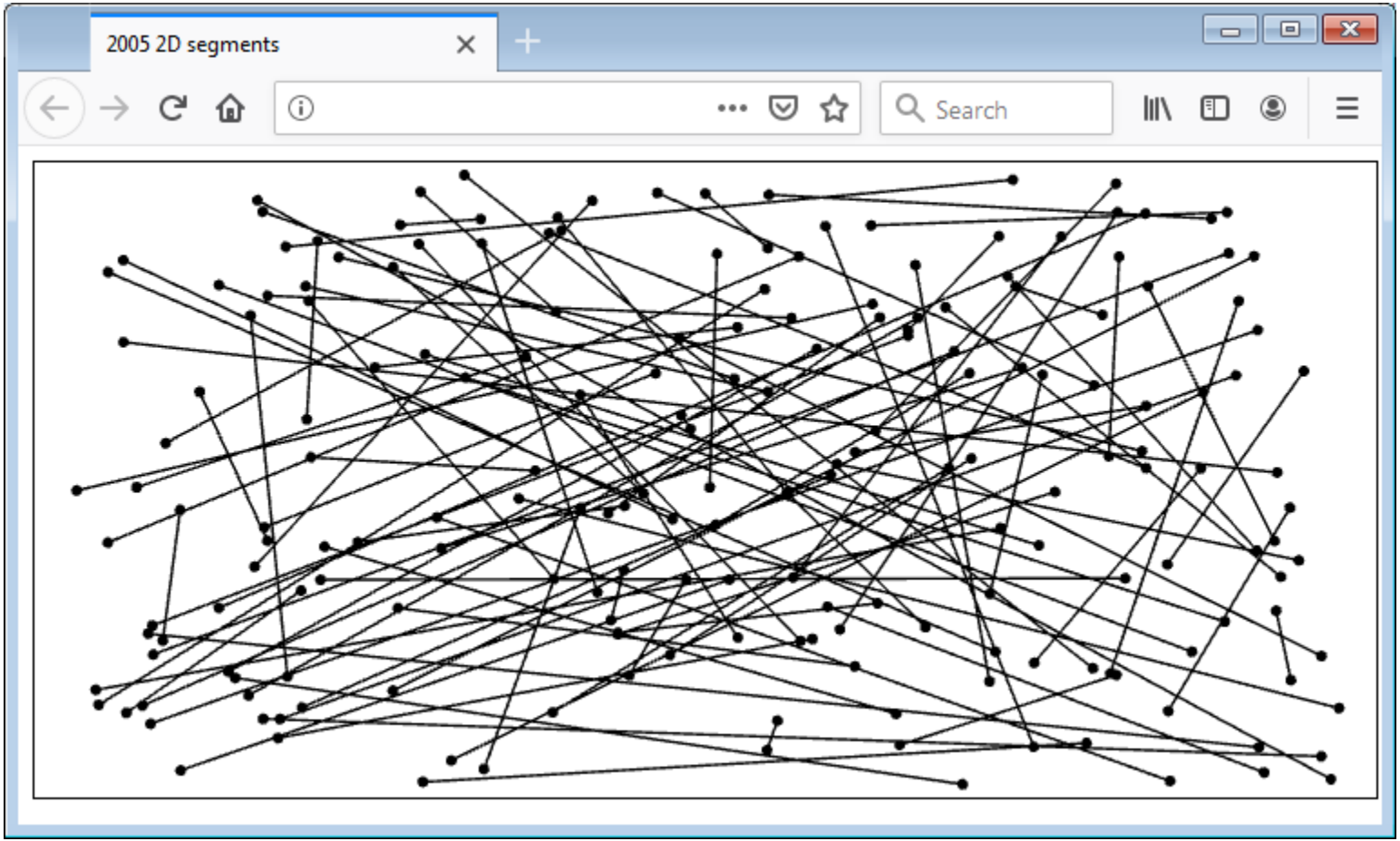
- End points
- Arrow for the vector
- Volume (width)
- Label

Random segments

2D segments

- Lines are standard segments
- End points are thick points (with pointSize) or small circles

```
var black = {color:[0,0,0]};
for (var i=0; i<100; i++)
{
   p=circle([random(-350,350),...],3).custom(black);
   q=circle([random(-350,350),...],3).custom(black);
   segment(p.center,q.center).custom(black);
}
```

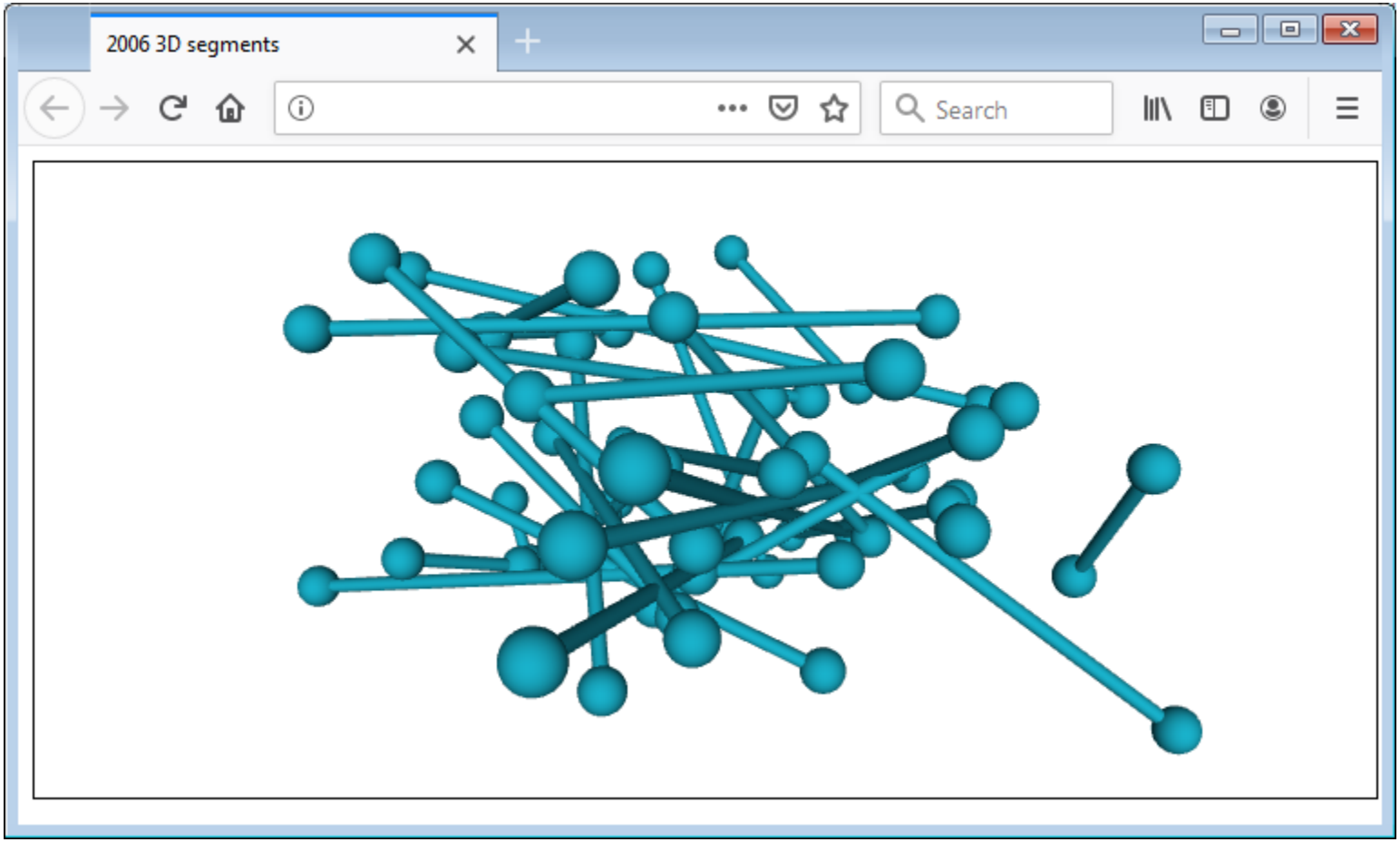


TRY IT

3D segments

- Lines are cylinders of sufficient length
- End points are spheres

```
var color = {color:[0.1,0.7,0.8]};
for (var i=0; i<30; i++)
{
   p = sphere([random(-50,50),...],3).custom(color);
   q = sphere([random(-50,50),...],3).custom(color);

   v = vectorPoints(q.center,p.center);
   cylinder(p.center,1,Math.sqrt(scalarProduct(v,v)))
                        .custom(color).custom({focus:v});
}
```



TRY IT

Random vectors

Note

- End objects are at the ends of a segment
- For vectors the arrow must be shifted, so the sharp vertex is at the end

2D and 3D vectors

- Arrow drawn with cone
- Cone offset makes its origin in the vertex , i.e. origin = [0,0,1]

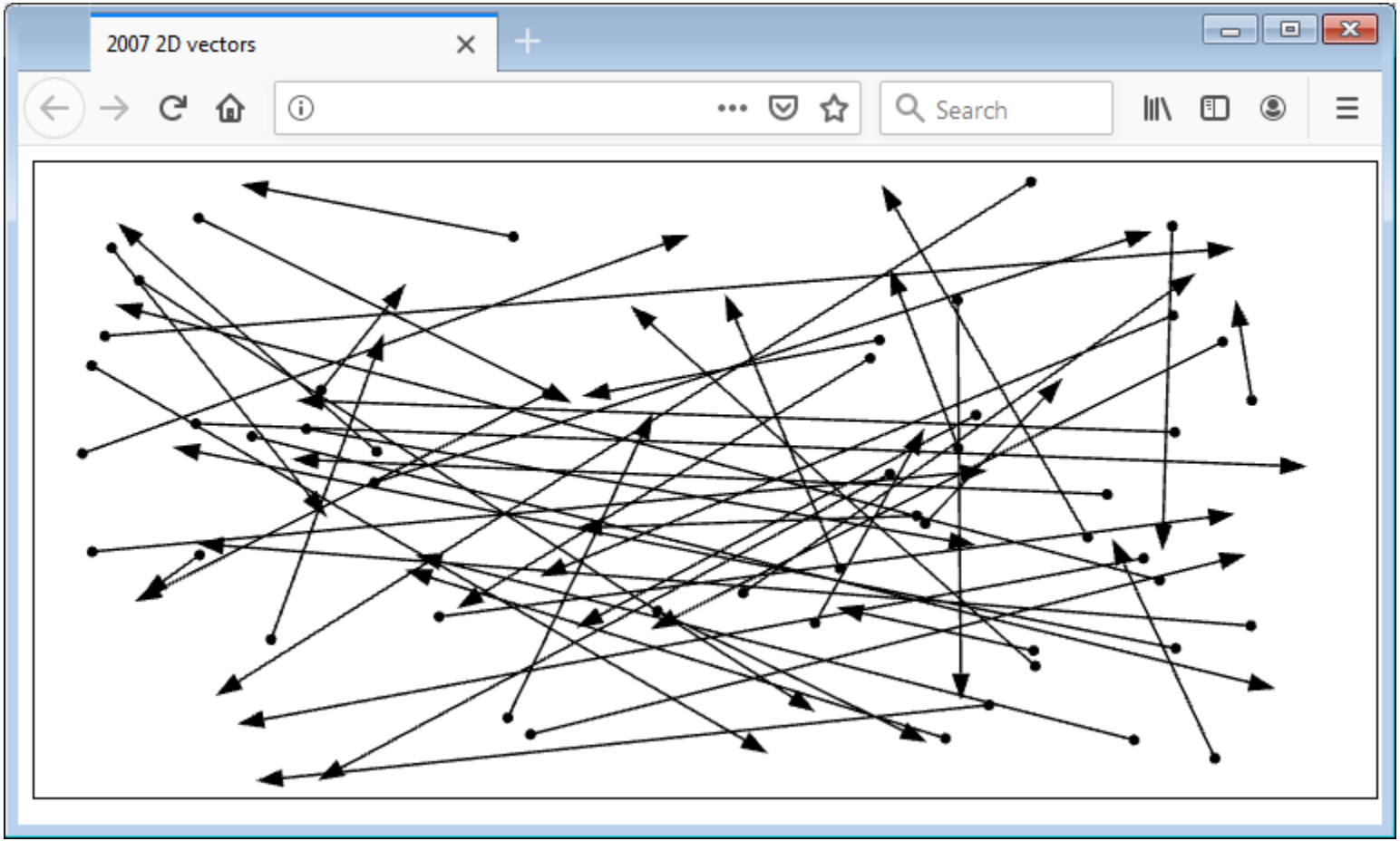
Vector body

- If the body is wide, it will show up where the arrow is

Implementation

- Starting point is circle p and, end point is vector q
- Arrow is a cone with changed origin and orientation along vector v between the center of p and q
- Coefficient l shrinks the body by 15 units (that is the length of the arrow)

```
p = circle([...],3);
q = [...,0];
v = vectorPoints(q,p.center);
cone(q,5,15).custom({origin:[0,0,1], focus:v});
l = 15/Math.sqrt(scalarProduct(v,v));
q = [q[0]-l*v[0],q[1]-l*v[1],0];
segment(p.center,q);
```

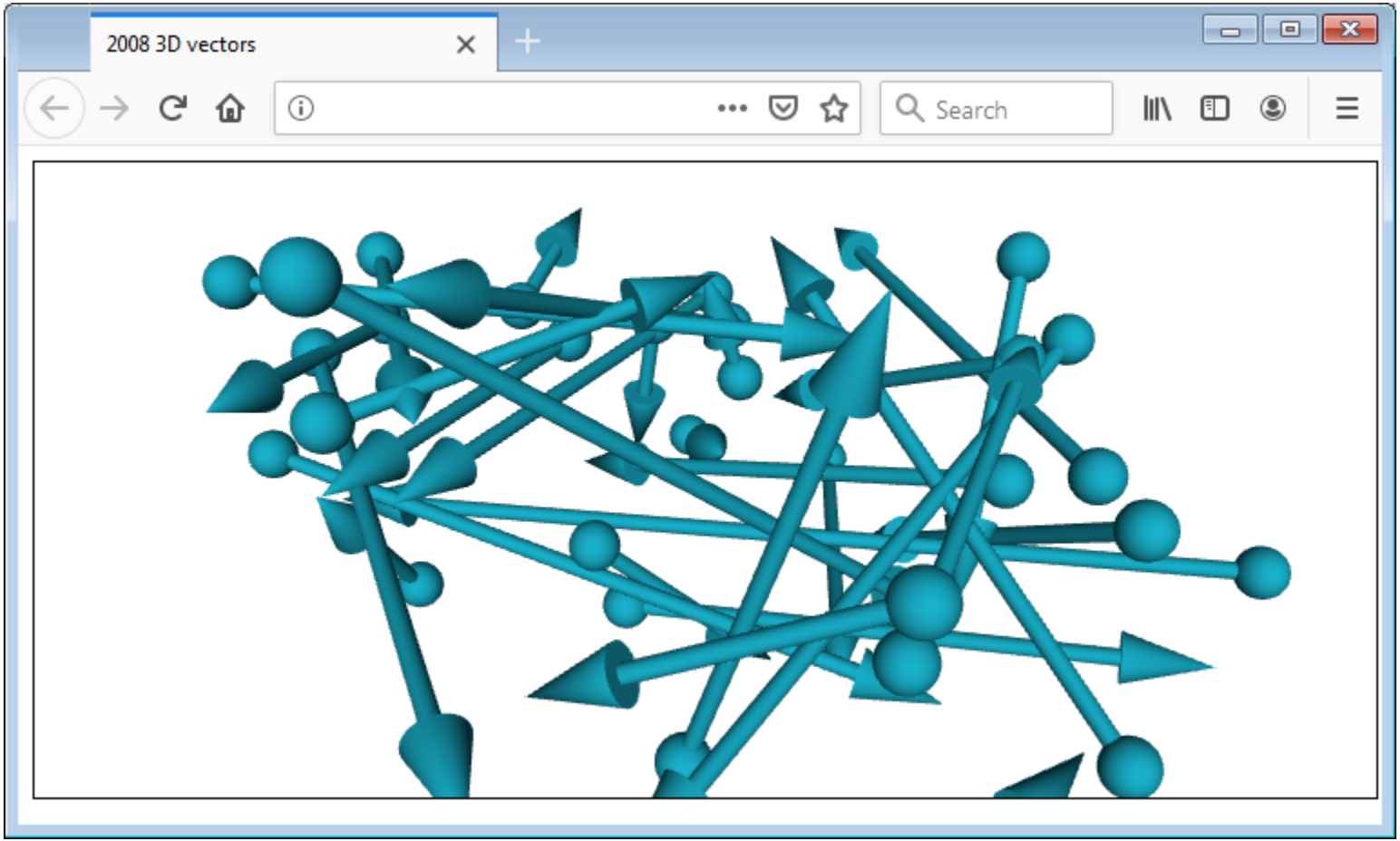


TRY IT

Alternative solution (with 3D vector)

- Directly creating the initial circle p and the ending correctly positioned non-oriented cone q
- Then fixing cone orientation (via vector v)
- Creating vector body with the correct length and orientation

```
p = sphere([...],3);
q = cone([...],3,10).custom({origin:[0,0,1]});
v = vectorPoints(q.center,p.center);
q.focus = v;
cylinder( p.center, 1,
          Math.sqrt(scalarProduct(v,v))-10
        ).custom(color).custom({focus:v});
```



TRY IT

Vertices of objects

Triangular prism with shapes on sides

- Models of several objects, more complex than a segment
- Drawn with lines, but have points at ends

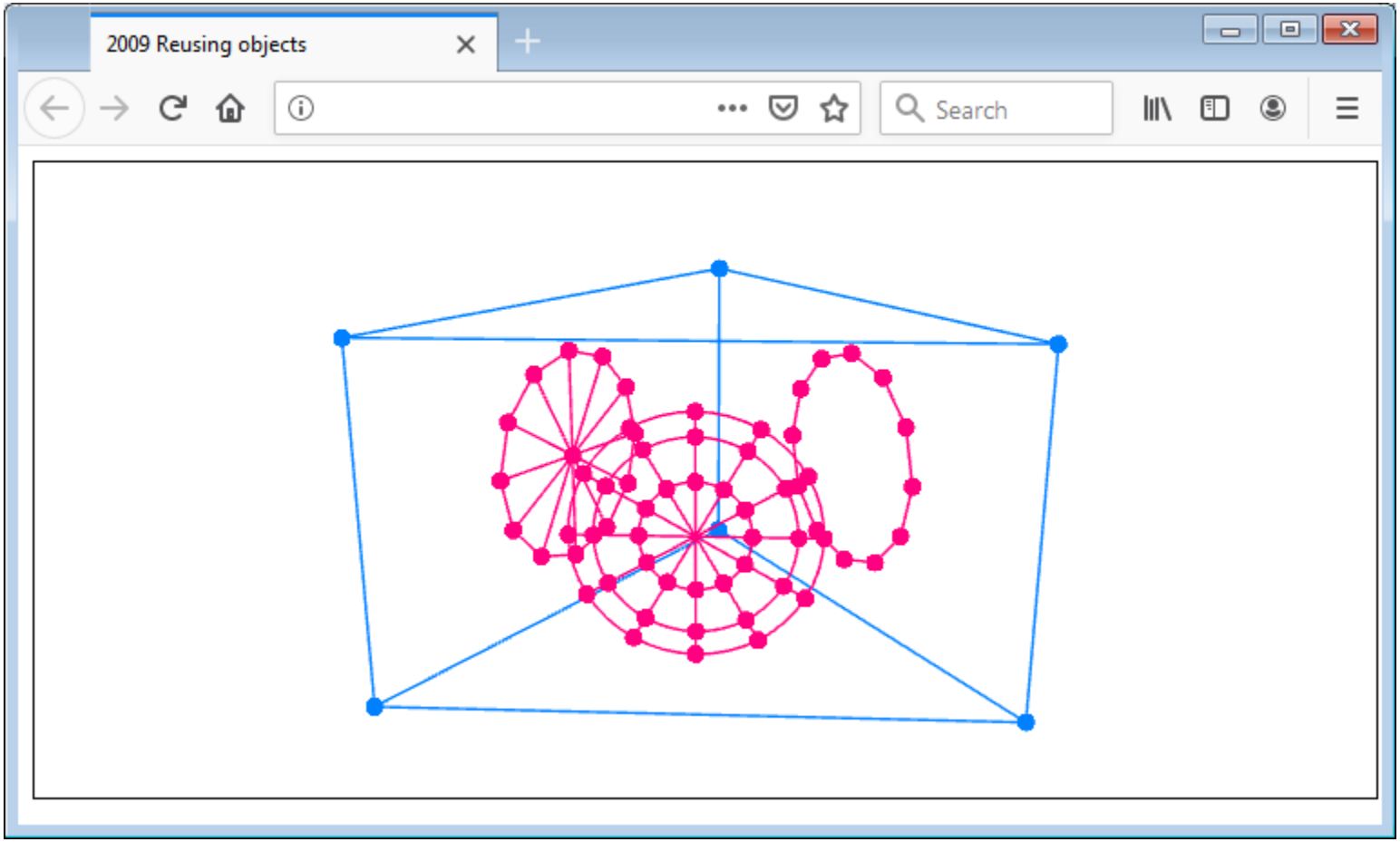
Implementation idea №1

- Done as segments with small circles
- Easy implementation
- Too many objects

Implementation idea №2

- Reusing the object
- Object is drawn in line mode
- Then is cloned in point mode

```
a = prism(...).custom({mode:Suica.LINE,...});
sameAs(a).custom({mode:Suica.POINT,pointSize:10});

a = cylinder(...).custom({mode:Suica.LINE,...});
sameAs(a).custom({mode:Suica.POINT,pointSize:10});
```



TRY IT

Sphere, cylinder and cone

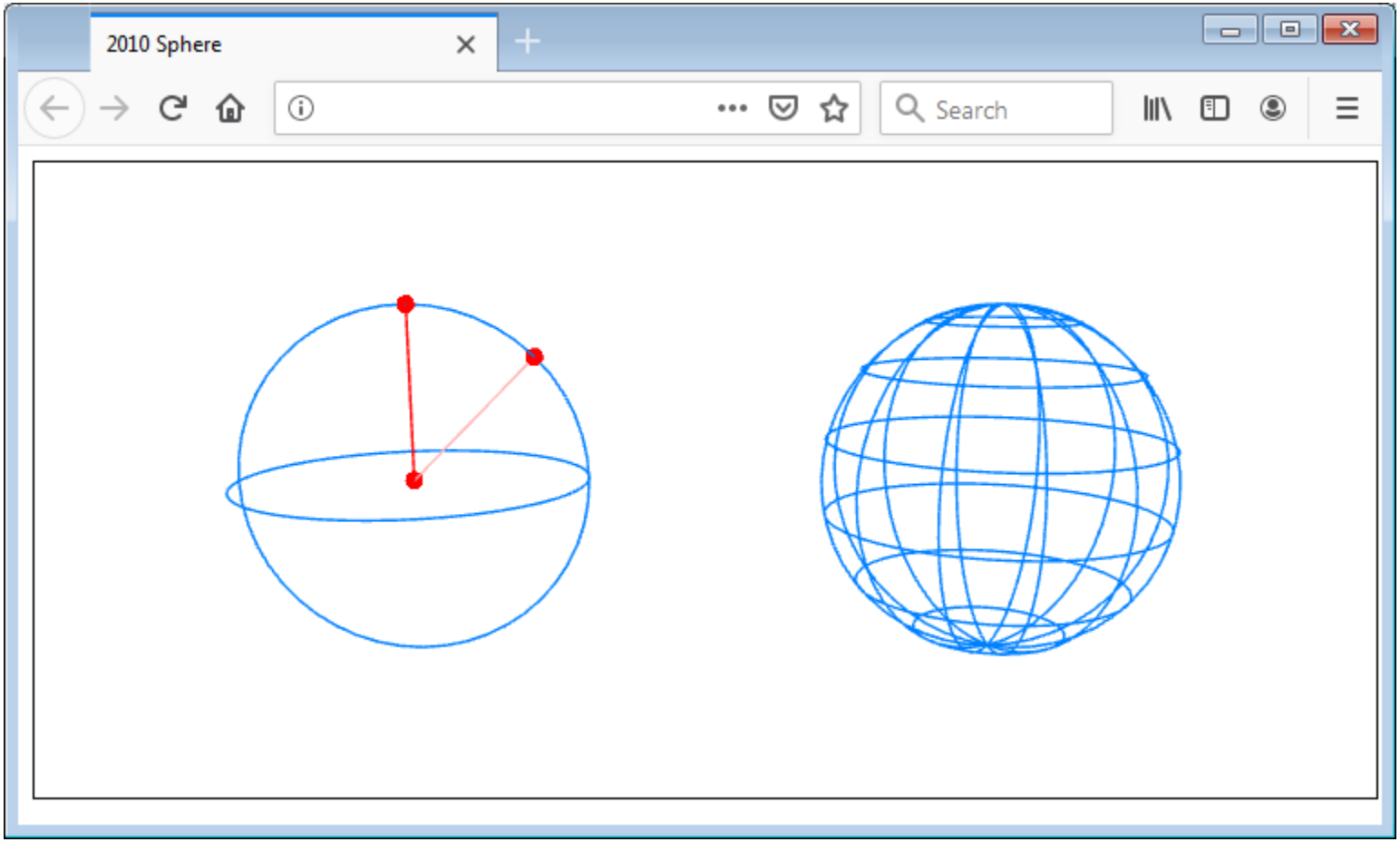
Sphere, cylinder and cone

Default visualization

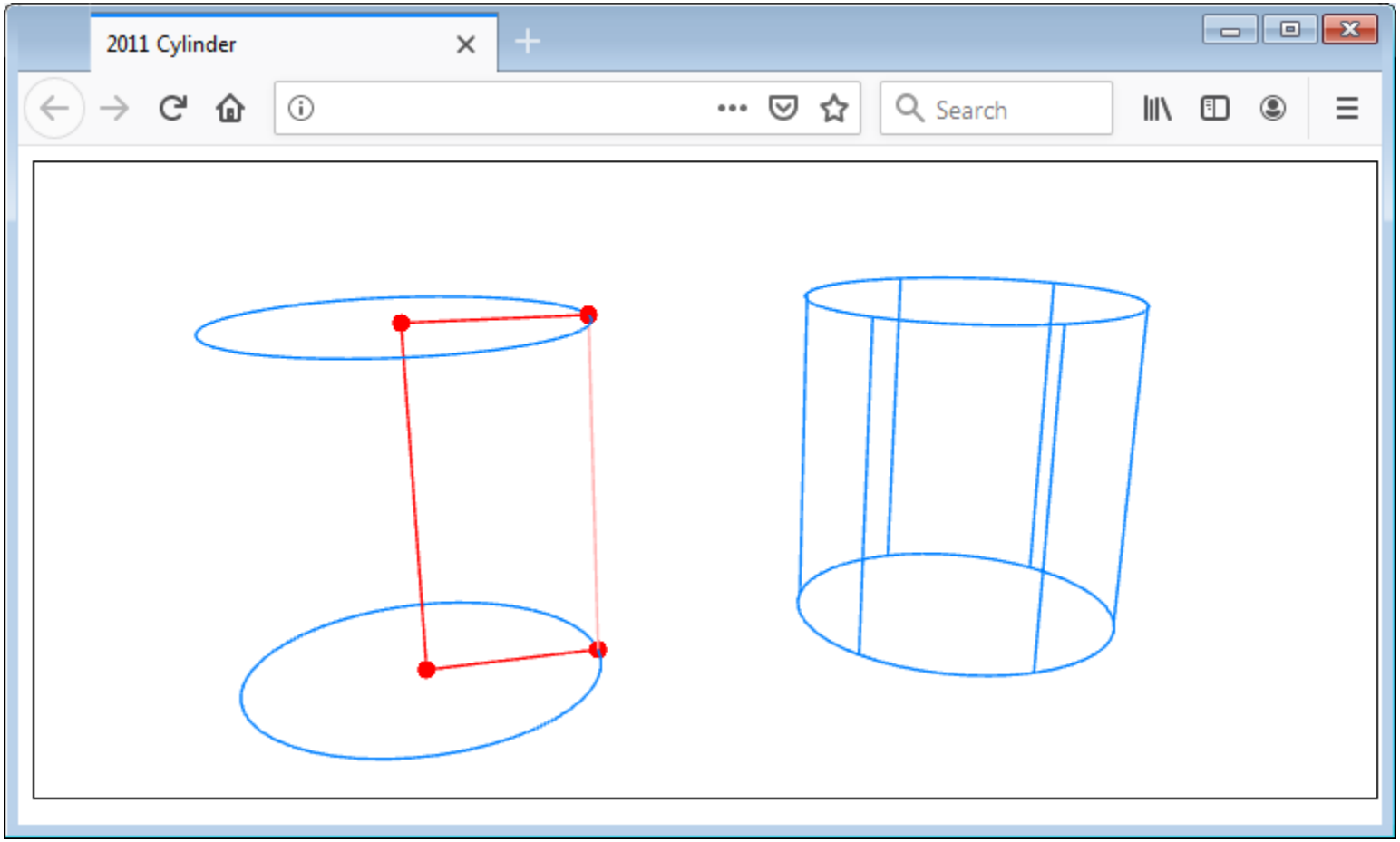
- With sphere, cylinder, and cone
- With segment

Missing elements

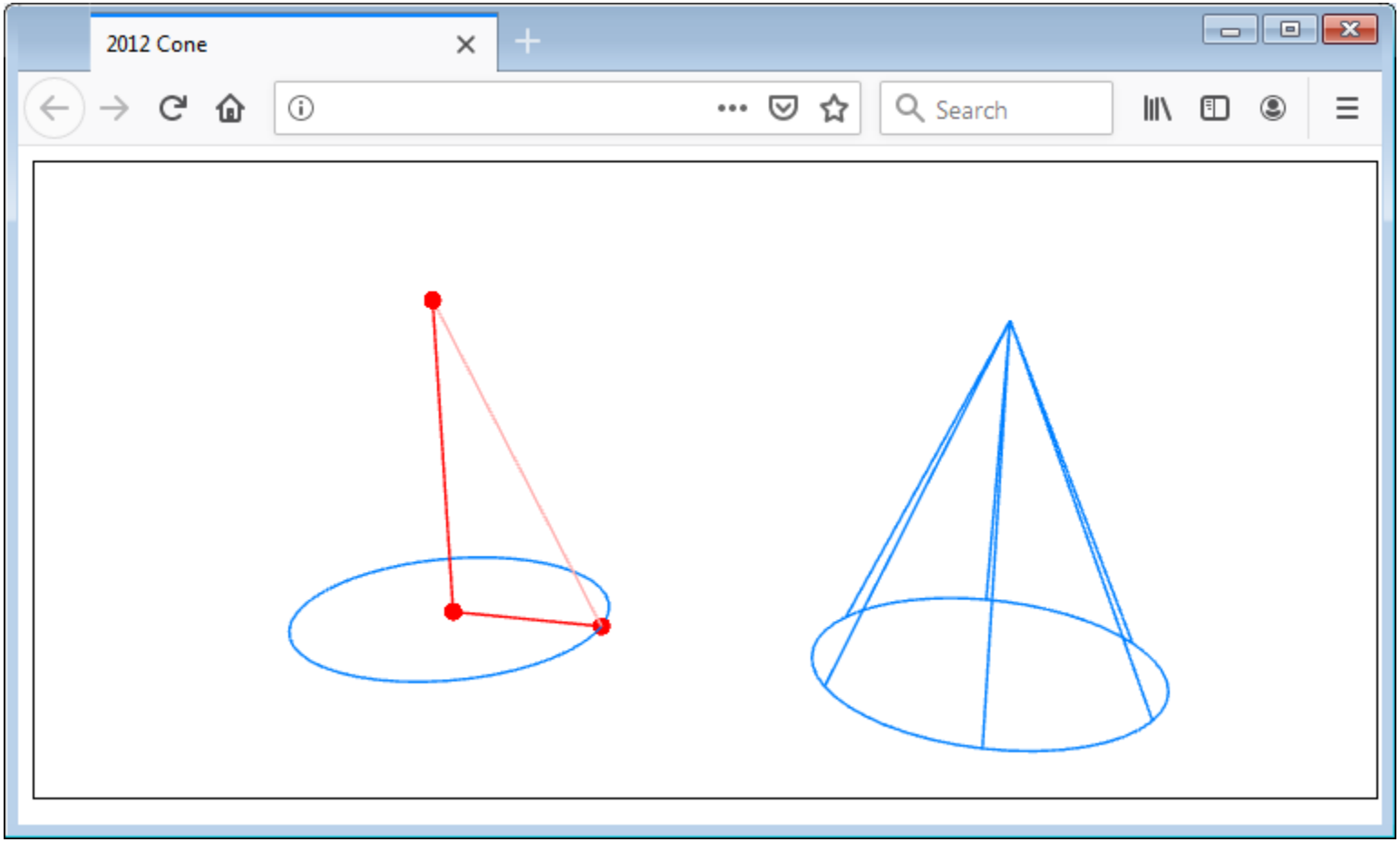
- Nothing is missing, there are extra elements
- Manually drawing only necessary elements
- Feci quod potui, faciant meliora potentes



TRY IT



TRY IT



TRY IT

Torus

Torus visualization

General problems

- No such object in Suica
- No object, that can be configured to look like a torus
- Torus must be defined by the user

Creating a torus

Approach №1 – vertical circles

- Torus is a circle, rotated around an axis
- Defining and rotating an offset circle

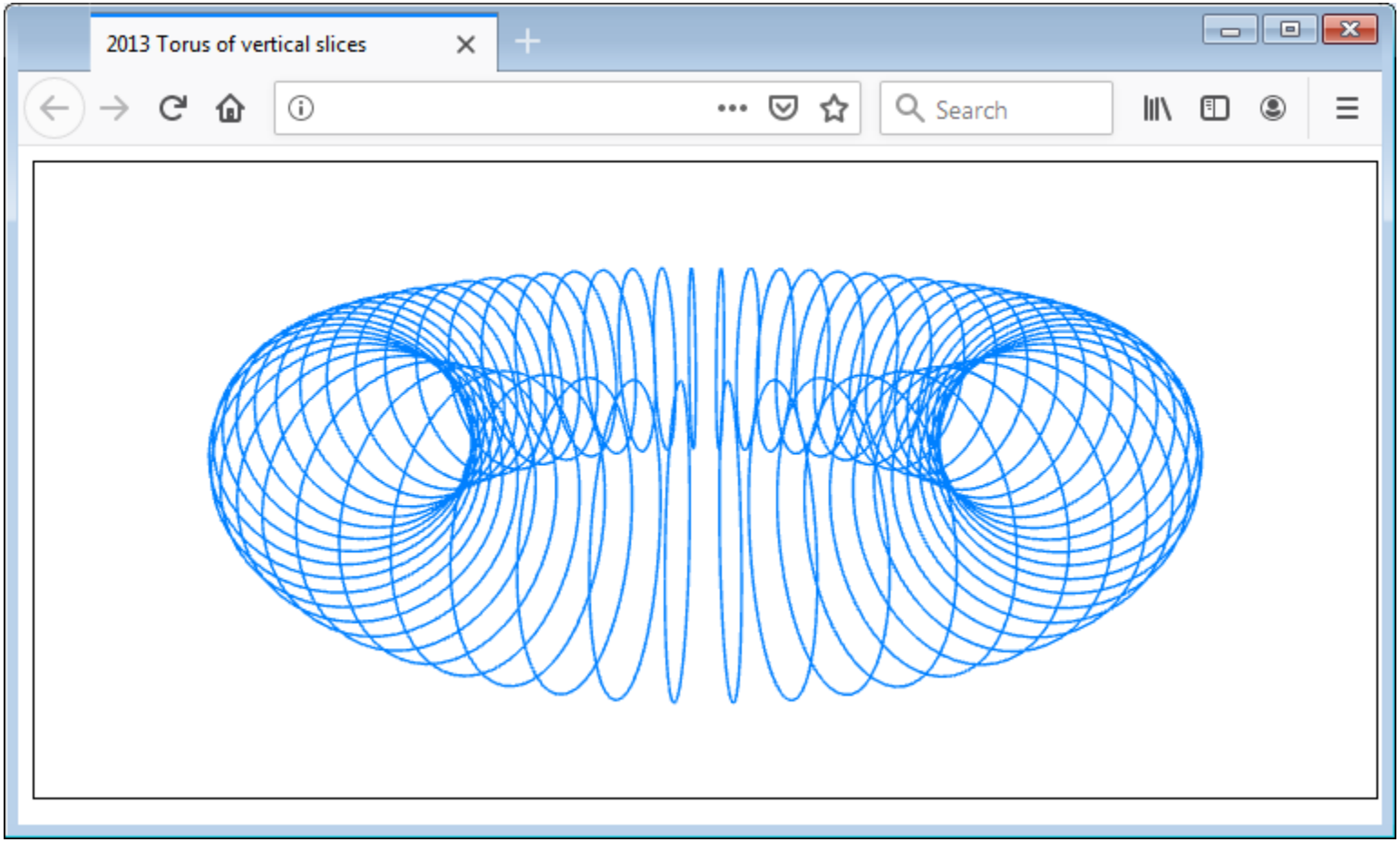
Problem

- If offset with center, then spin will rotate it around the center, not around the axis
- Property spin should be unaware, that center is modified
- If offset is done with origin, there will be a problem with focus

Solution

- Hiding well oriented circle in a group object
- Thus both the circle and the group has own focus, spin and center, independent on each other
- Cloning and rotating the group

```
a = group([circle([0,30,5],10).custom({
                mode:Suica.LINE,
                color:[0,0.5,1],
                focus:[1,0,0]})
        ]);

for (var i=0; i<60; i++)
   sameAs(a).custom({spin:i*2*Math.PI/60});
```

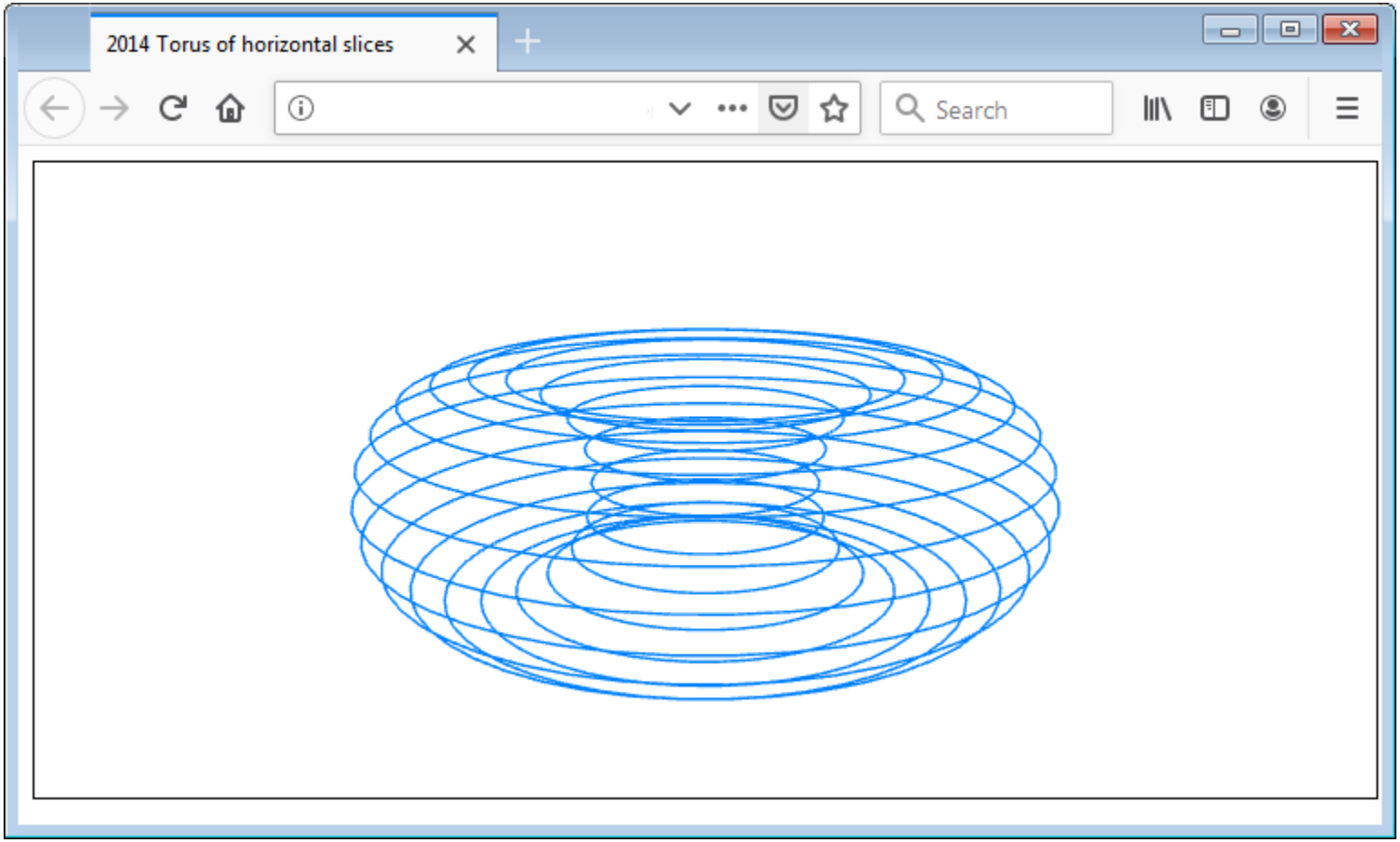


TRY IT

Approach №2 – horizontal circles

- Creating horizontal circles, their Z coordinates and radii traverse … a circle

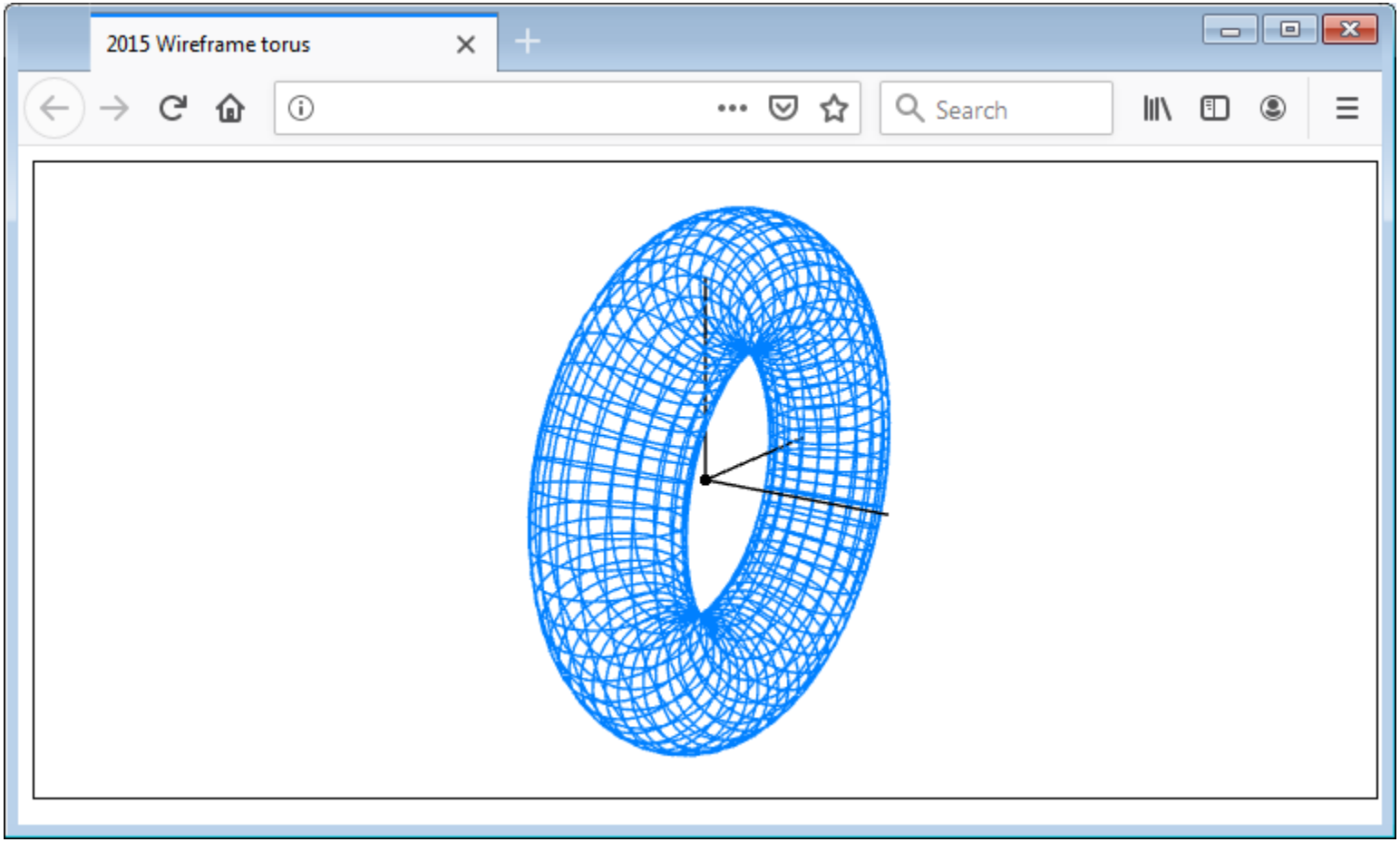
```
for (var i=0; i<20; i++)
{
   var a = 2*Math.PI*i/20;
   circle( [0,0,10*Math.sin(a)],
           20+10*Math.cos(a) ).custom({
        mode:Suica.LINE,
        color:[0,0.5,1]});
}
```



TRY IT

Approach №3 – combined circles

- Torus appears to be made of plates
  - Creating vertical circles
  - Creating horizontal circles
- All circles are in a group – thus the torus could be manipulated as a single object



TRY IT

Summary

Visualization

Visualization of geometrical objects

- Images of geometrical objects are not always the same as the images of graphical objects
- Composing images from several objects
- Creating images from scratch
- Suica defines only a few objects, all the rest are to be constructed by the user

Note

- Every object could be represented graphically in several different ways
- It is a matter of design, style and skills which one to use

The end

Comments, questions