ICT in SES

User objects

Lesson №12

Objects and styles

Creating graphical objects

Creating an objet

- With a class constructor **new Обект(…)**
- With a function **обект()**

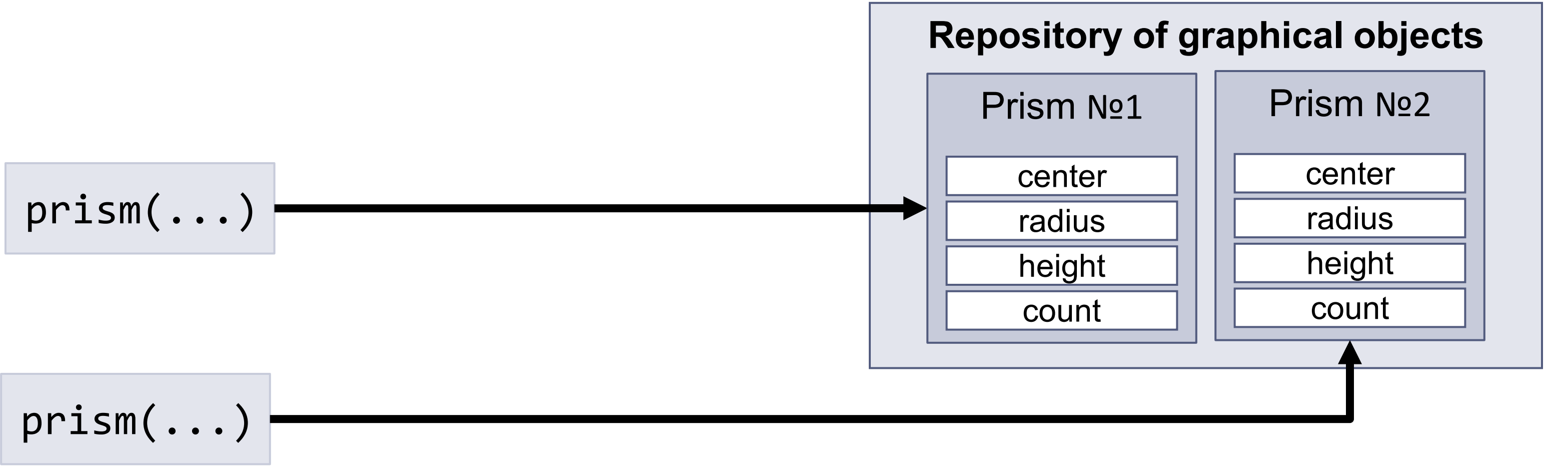
Behind the scene

- Storing created objects
- Using when the frame is generated

Names of objects

Anonymous

- Created without names
- Suitable for fixed objects



`prism(...)`

`prism(...)`

Repository of graphical objects

Prism №1

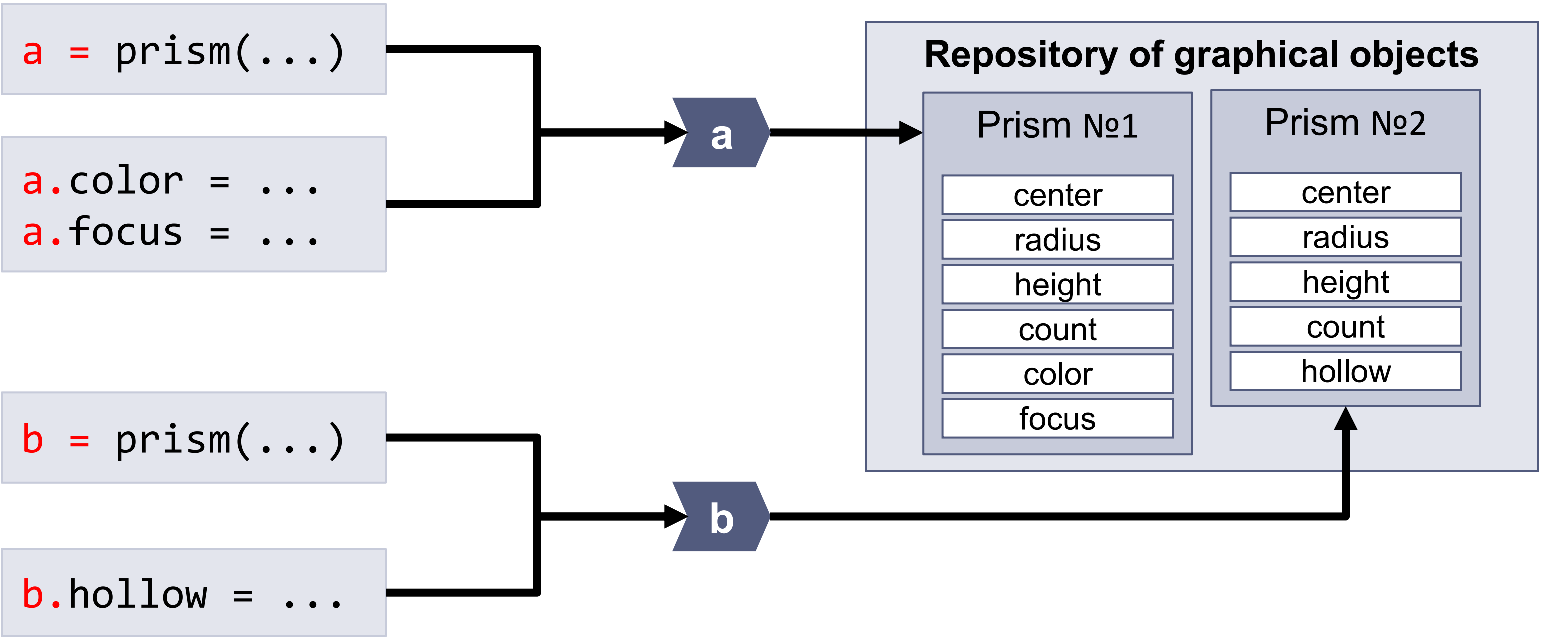
- center
- radius
- height
- count

Prism №2

- center
- radius
- height
- count

Objects with names

- Assigned to variables
- Properties are accessed via the variable



```
a = prism(...)
```

```
a.color = ...
a.focus = ...
```

```
b = prism(...)
```

```
b.hollow = ...
```

Set of properties

Observation

- Object is customized with several properties
- Many objects have the same properties

Goal

- Grouping properties in a style
- Applying a style onto many objects

Implementation

Until now

- Creating objects with names
- Setting properties one by one
- One variable could be reused for several objects

Disadvantages

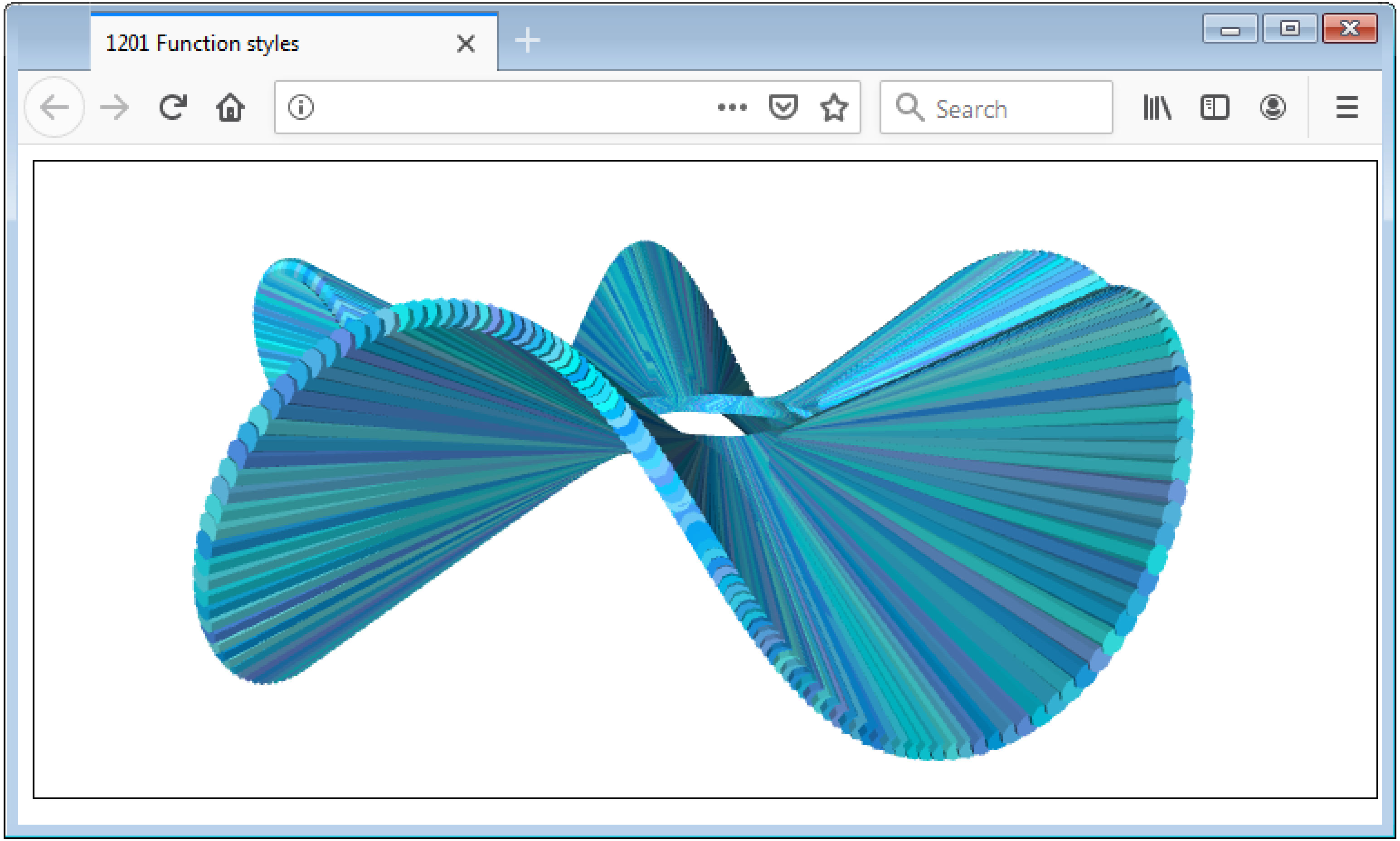
- Writing code for each property
- Not comfortable for managing many objects of the same type

Style with a function

Idea №1

- Function receives an object
- Add desired properties
- Returns the object

```
function bluish(object)
{
   object.color = [random(0,0.5),random(0.6,1),1];
   return object;
}
a = bluish(prism(center,1/2,15+5*Math.cos(5*alpha),8));
```



TRY IT

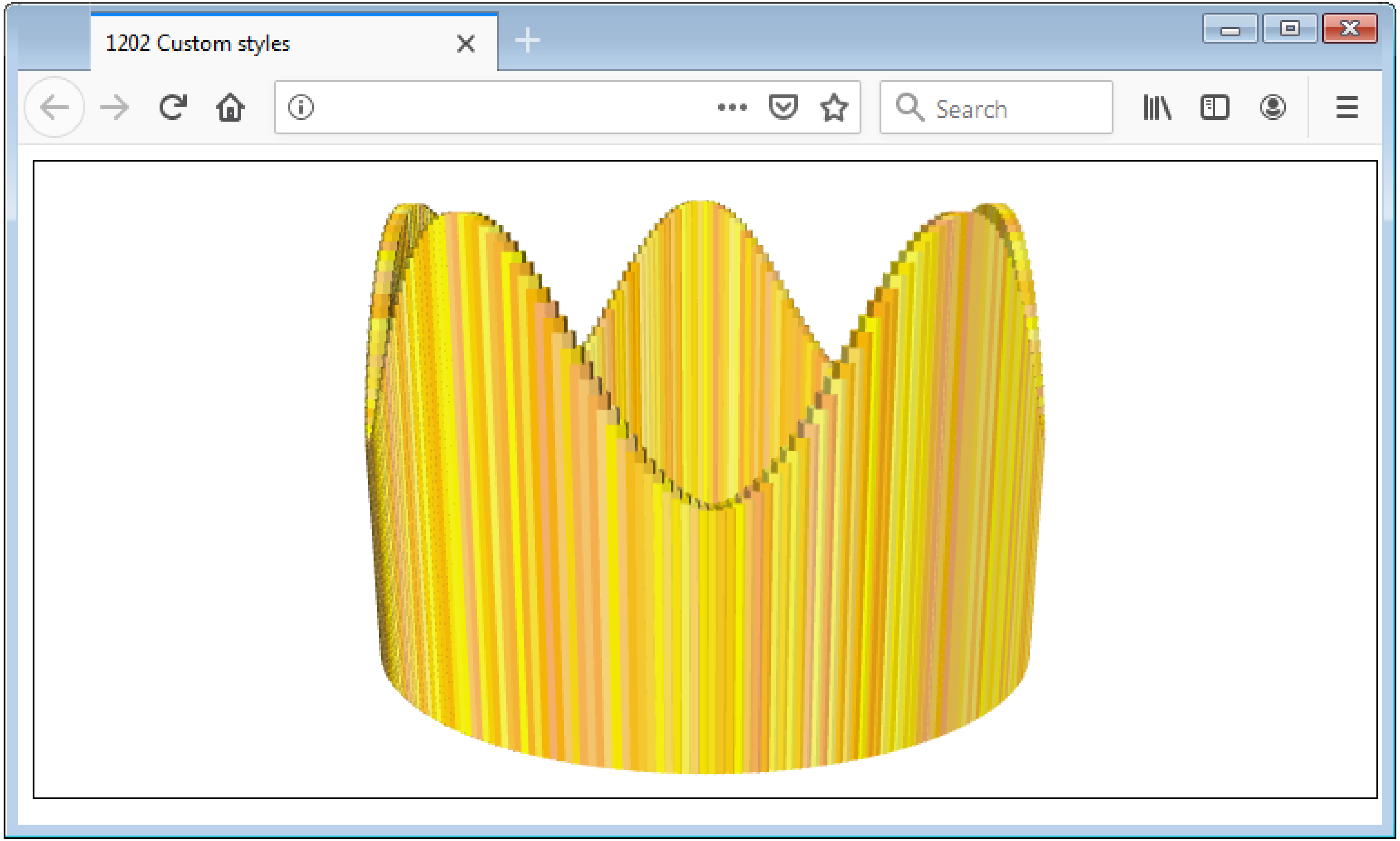
Idea №2

- Using method object.custom({property:value, … })
- Defines a set of properties applied when an object is created even if the object is anonymous

```
prism(center,1/2,15+5*Math.cos(5*alpha),8).custom({
   focus: [0,0,1],
   color: [1,random(0.7,1),random(0,0.5)],
   hollow: true });
```

- If the style is fixed, it can be stored in a variable and reuse as many times as it is needed

```
style = {focus:[0,0,1], ...};
prism(center,...).custom(style);
```



TRY IT

Group objects

Objects in Suica

Library Suica

- Base set of graphical objects
- Sufficient for the course goals

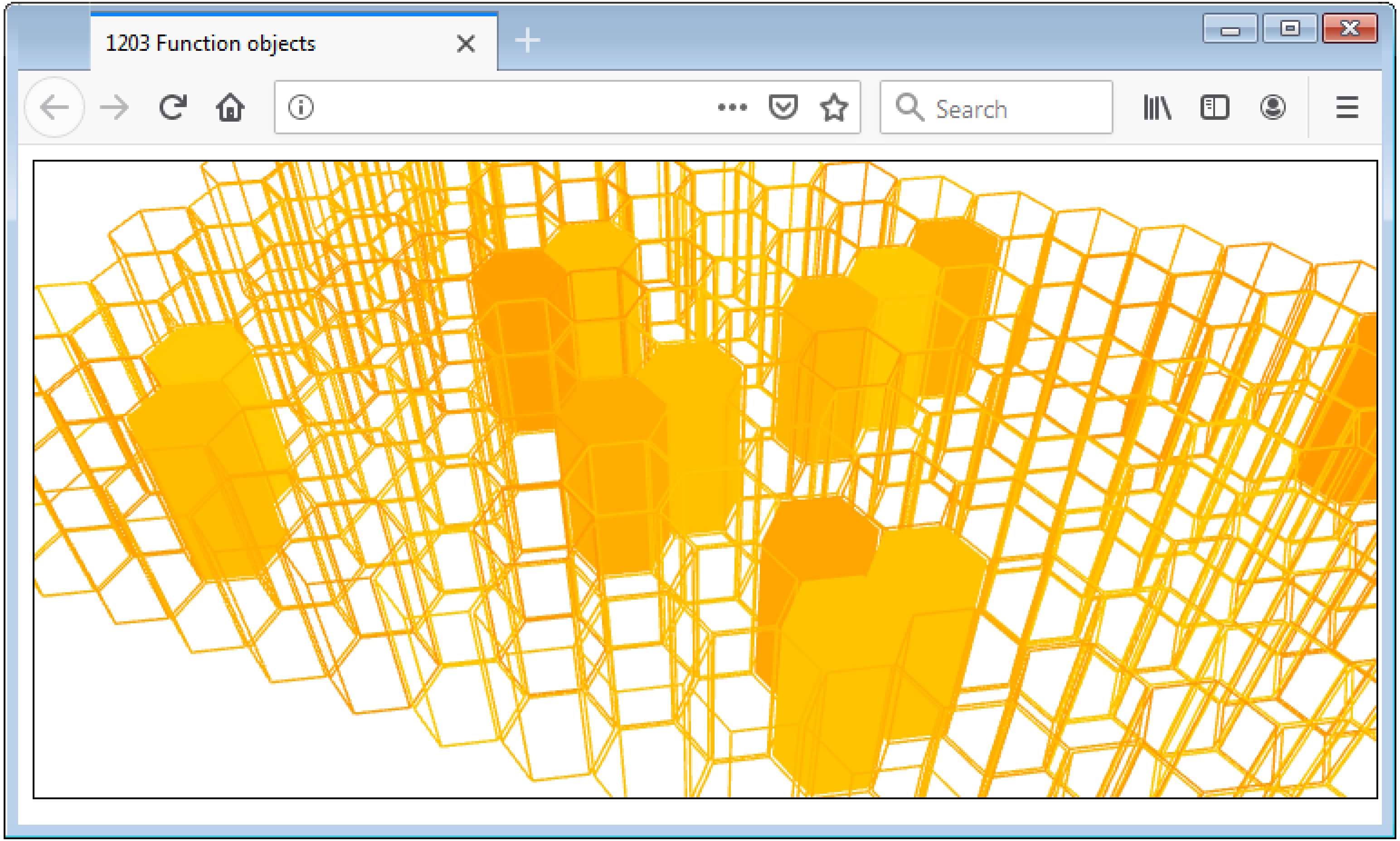
New objects

- Base objects have some degree of customization
- When needed a base object is upgraded
- Often this is done by combining base objects or by clipping objects

Upgrading an object

- Function creating an object with desired properties
- Parameters are only the required properties

```
function honeyComb(x,y)
{
   return prism([x,0.85*y,0],0.55,2,6).custom({
      color: [1,random(0.6,0.8),0],
      spin: radians(30),
      light: false,
      mode: (random(0,10)>9?Suica.SOLID:Suica.LINE)
   });
}
```

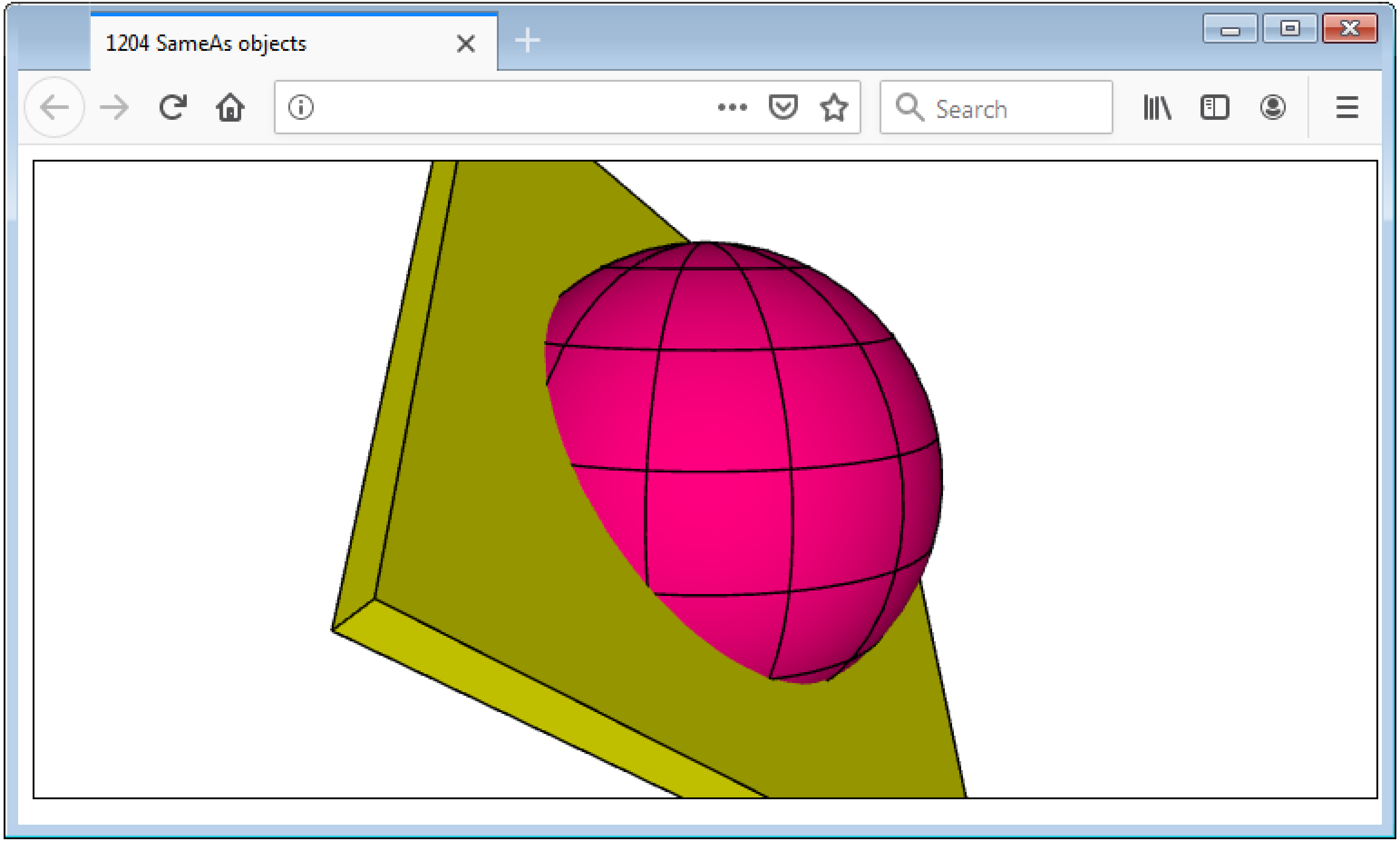

1203 Function objects
Search

TRY IT

Copying an object

- Function sameAs(обект) creates a copy
- Useful to clone an object and its properties
- Example of adding wireframe to objects

```
a = sphere([0,0,0],5);
b = cuboid([0,0,0],[15,15,1]).custom({
       color:[1,1,0],
       focus:[1,1,1],
       spin:Math.PI/4
     });
contour = {color:[0,0,0], mode:Suica.LINE};
sameAs(a).custom(contour);
sameAs(b).custom(contour);
```



TRY IT

Group objects

A group of objects

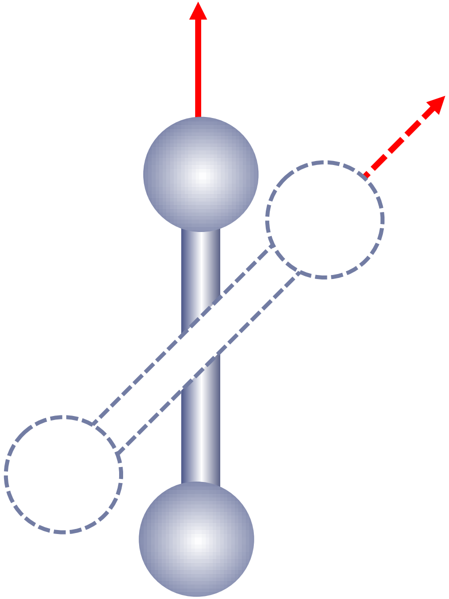
- A problem to change position
- Each object must change its position

Solution

- Objects are grouped in a single object
- The group has its own position and orientation

Example

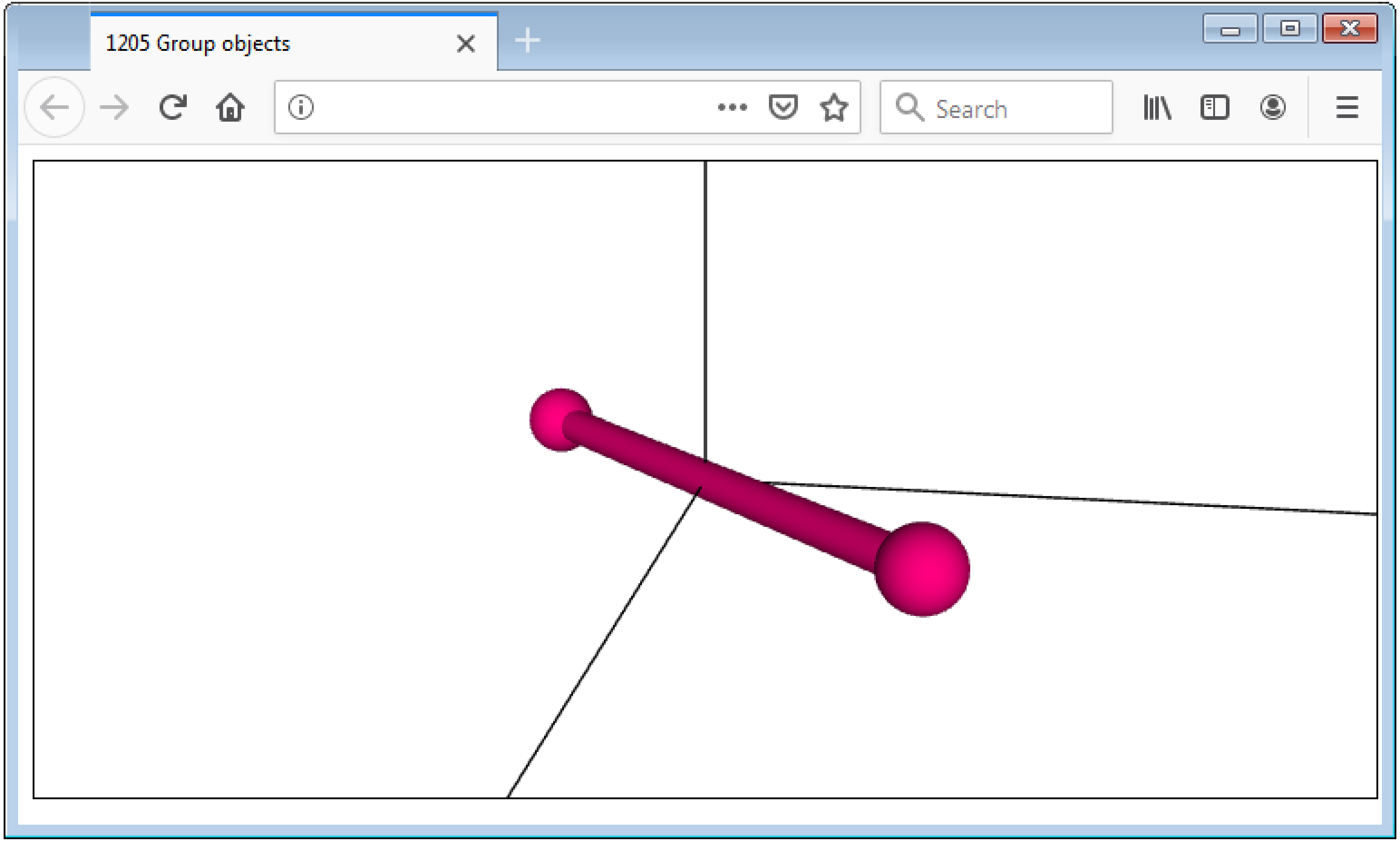
- Vertical cylinder with two spheres
- The group is rotated with focus
- The coordinates of the group remain the same



Solution

- Using class new Suica.Group or function group
- The parameter is an array of objects
- The result is a group object with own position and orientation

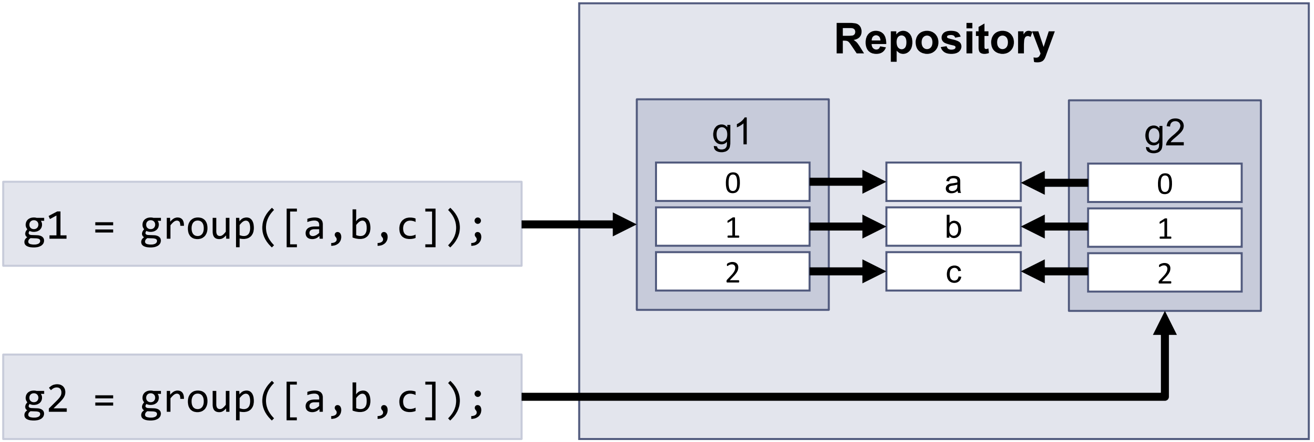
```
a = group ([
         sphere([0,0,-4],1/2),
         sphere([0,0,4],1/2),
         cylinder([0,0,-4],1/4,8)
   ]);
a.focus = [1,1,0];
```



TRY IT

Shared objects

- Graphical objects are JS objects
- Same subobjects of groups become shared objects
- Changing a shared object changes all it appearances in all groups where it blongs



```
g1 = group([a,b,c]);
```

```
g2 = group([a,b,c]);
```

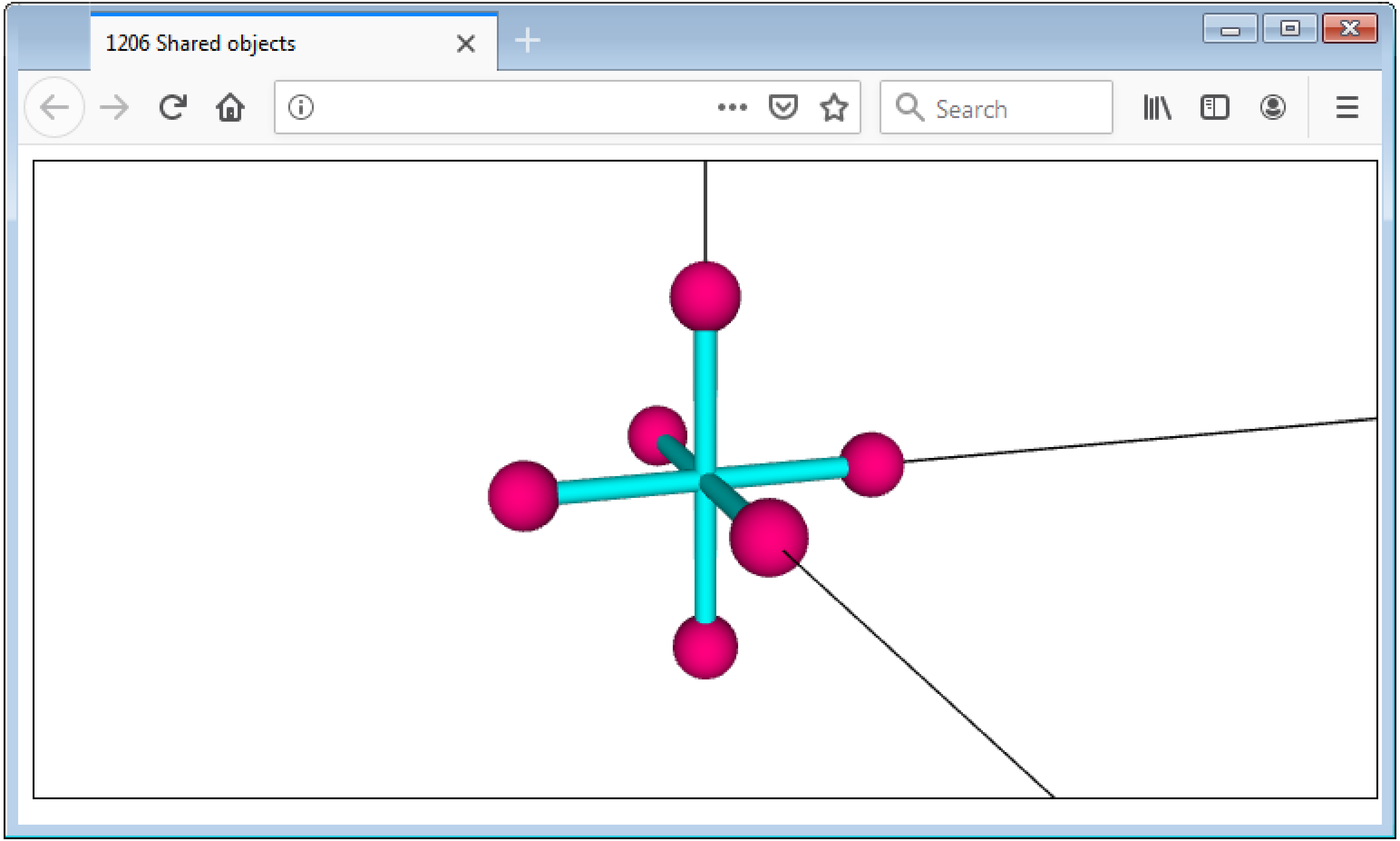
Example

- Building three groups with shared objects
- If the colour of an object is changed, this is seen in all groups

```
a = sphere([0,0,-4],3/4),
b = sphere([0,0,4],3/4),
c = cylinder([0,0,-4],1/4,8)

g1 = group([a,b,c]);
g2 = group([a,b,c]).custom({focus:[1,0,0]});
g3 = group([a,b,c]).custom({focus:[0,1,0]});;

c.color = [0,1,1];
```

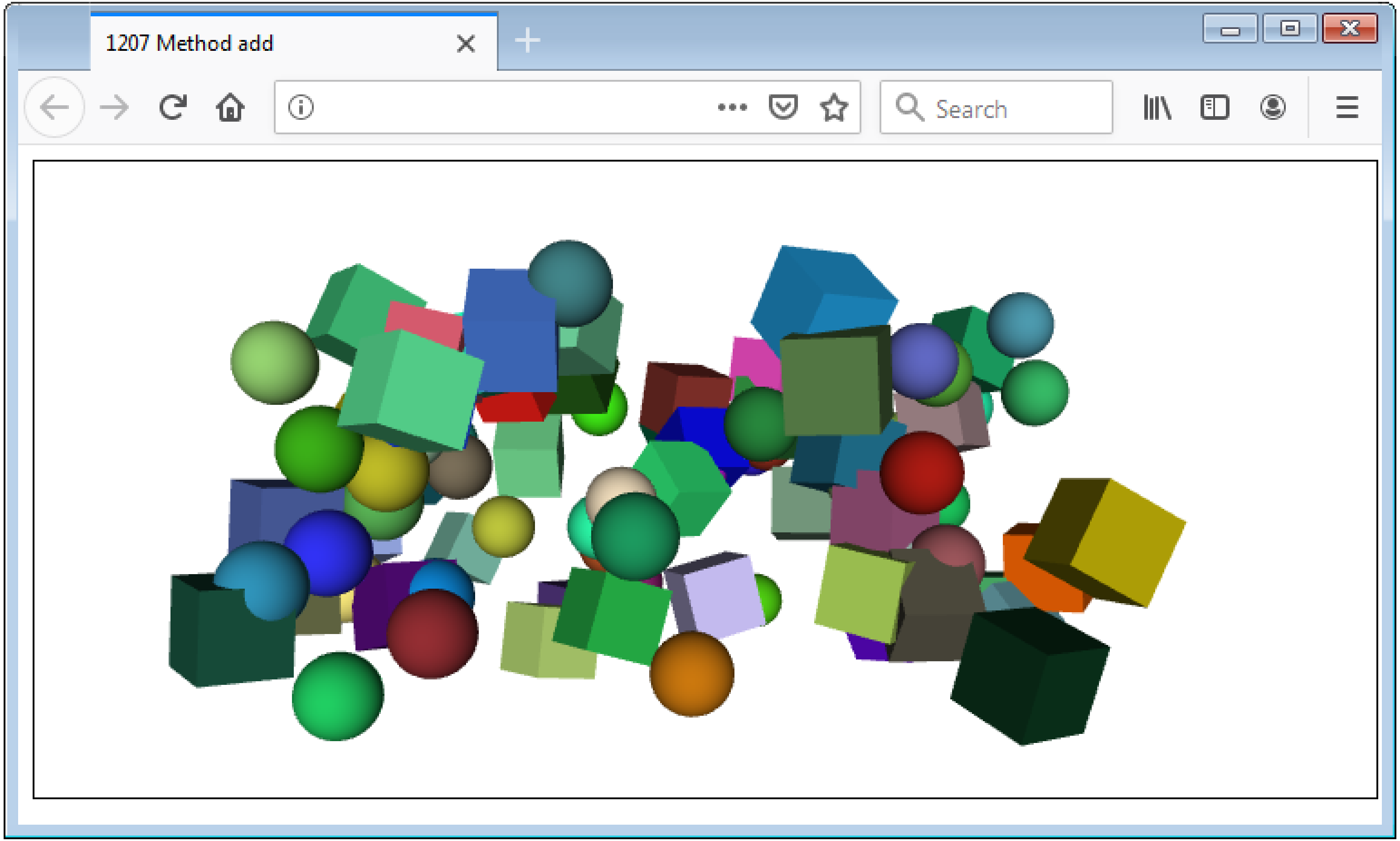


TRY IT

Adding elements

- Adding objects with add(object)
- Objects could be different graphical objects

```
a = group([]);
for (var i=0; i<100; i++)
{
   var style = {...};
   if (random(-1,1)>0)
      a.add(cube([0,0,0],2).custom(style));
   else
      a.add(sphere([0,0,0],1).custom(style));
}
```

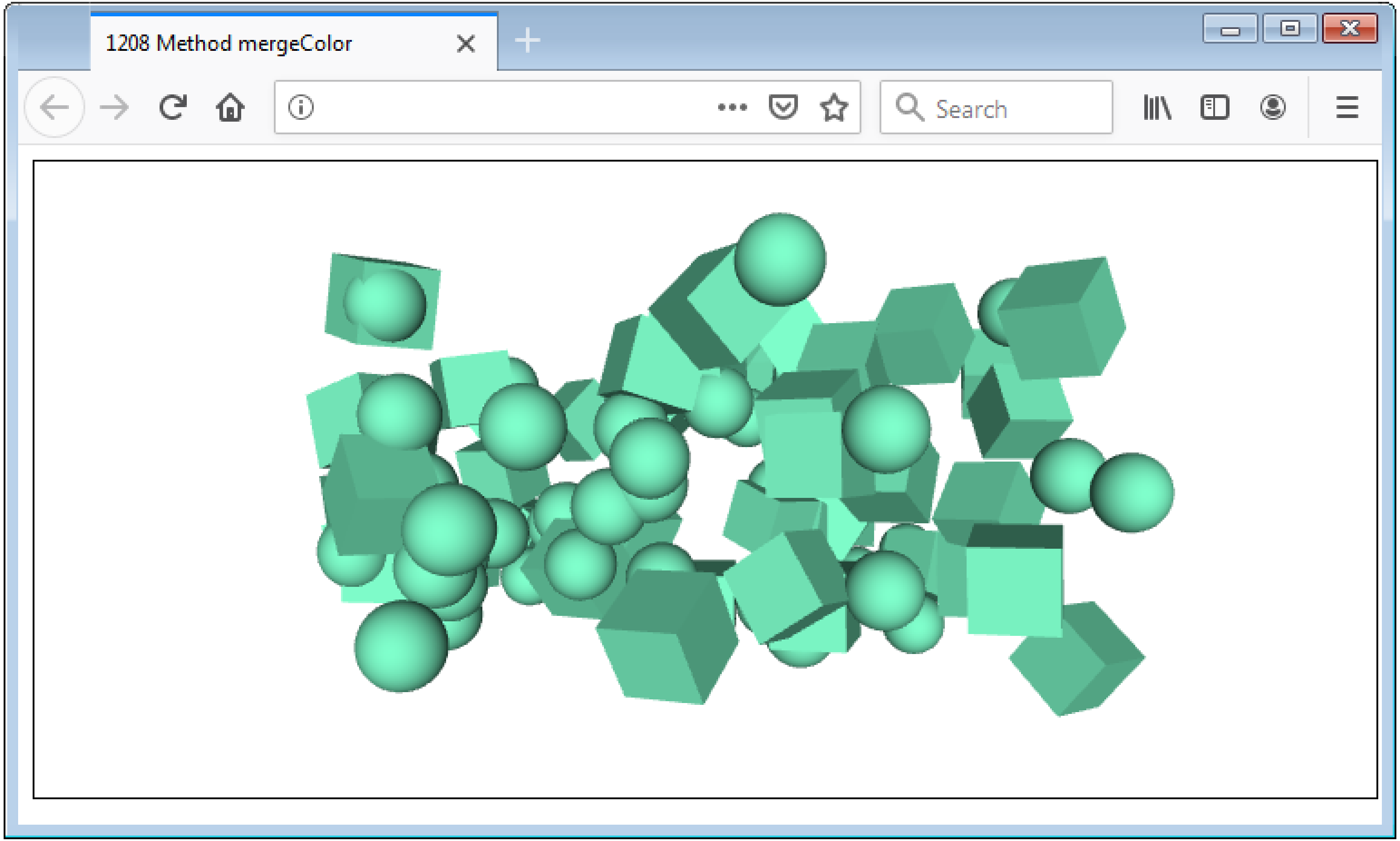


TRY IT

Shared colour

- Each object in a group has own colour
- Method mergeColor() removes the individual colours, objects use the group colour

```
a = group([]);
for (var i=0; i<100; i++)
{
  var style = {color: ...];
  a.add(cube([0,0,0],2).custom(style));
  ...
}
a.mergeColor();
a.color = [0.5,1,0.8];
```



TRY IT

Example

Brick chimney

- A row of brownish bricks in a circle
- The another row and so on
- Upper rows are shrinked

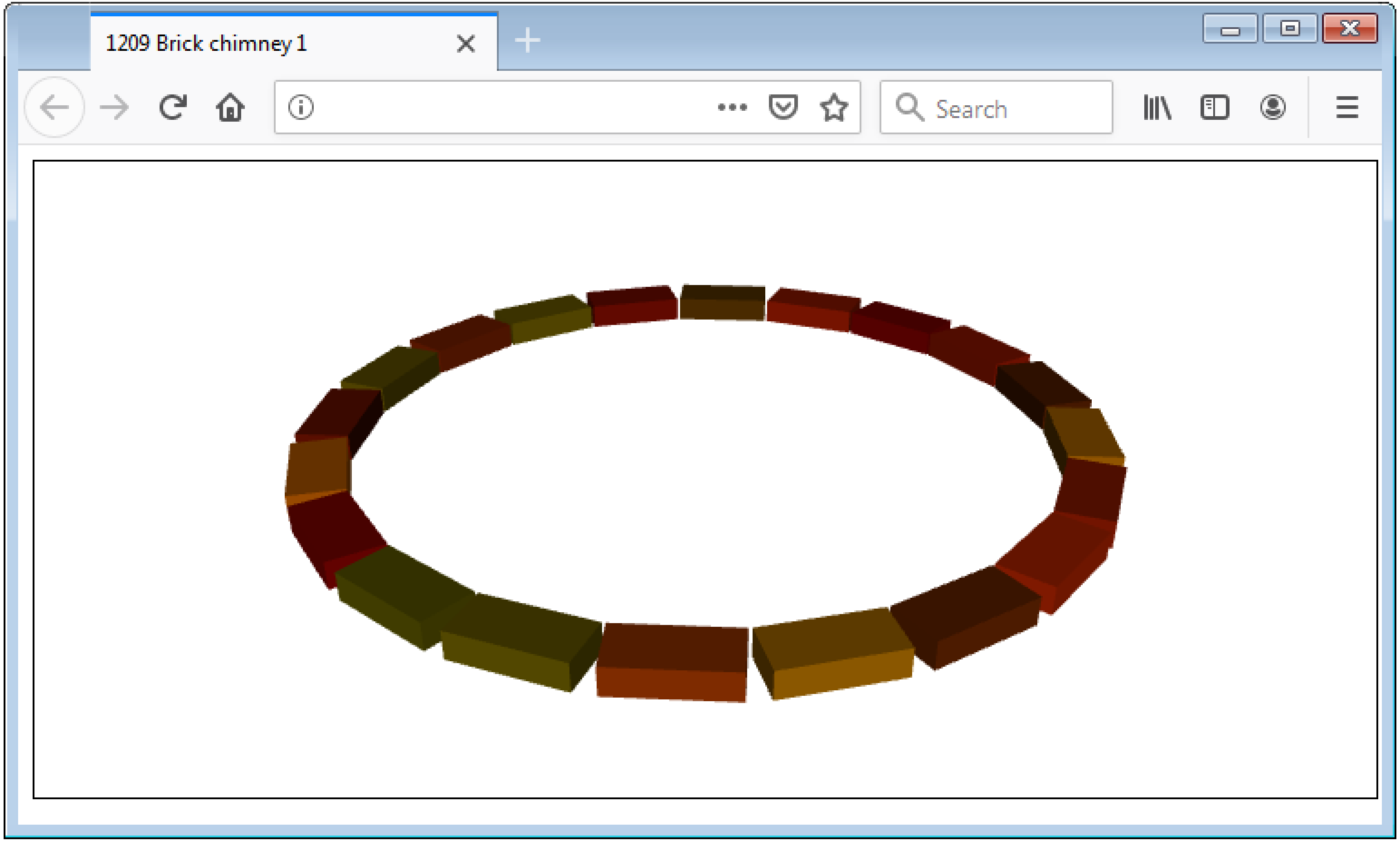
Idea

- Using a group object (only one)

Creating a row of bricks

- The row has 20 brownish bricks
- They all have modified centers (shifted 7 units)
- Rotated at 18° from one anothe

```
row = group([]);
for (var i=0; i<20; i++)
   row.add(
      cuboid([0,0,0],[2,4,1]).custom({
         origin: [7,0,0],
         color: [random(0.3,0.7),random(0,0.4),0],
         spin: 2*Math.PI*i/20
      })
   );
```

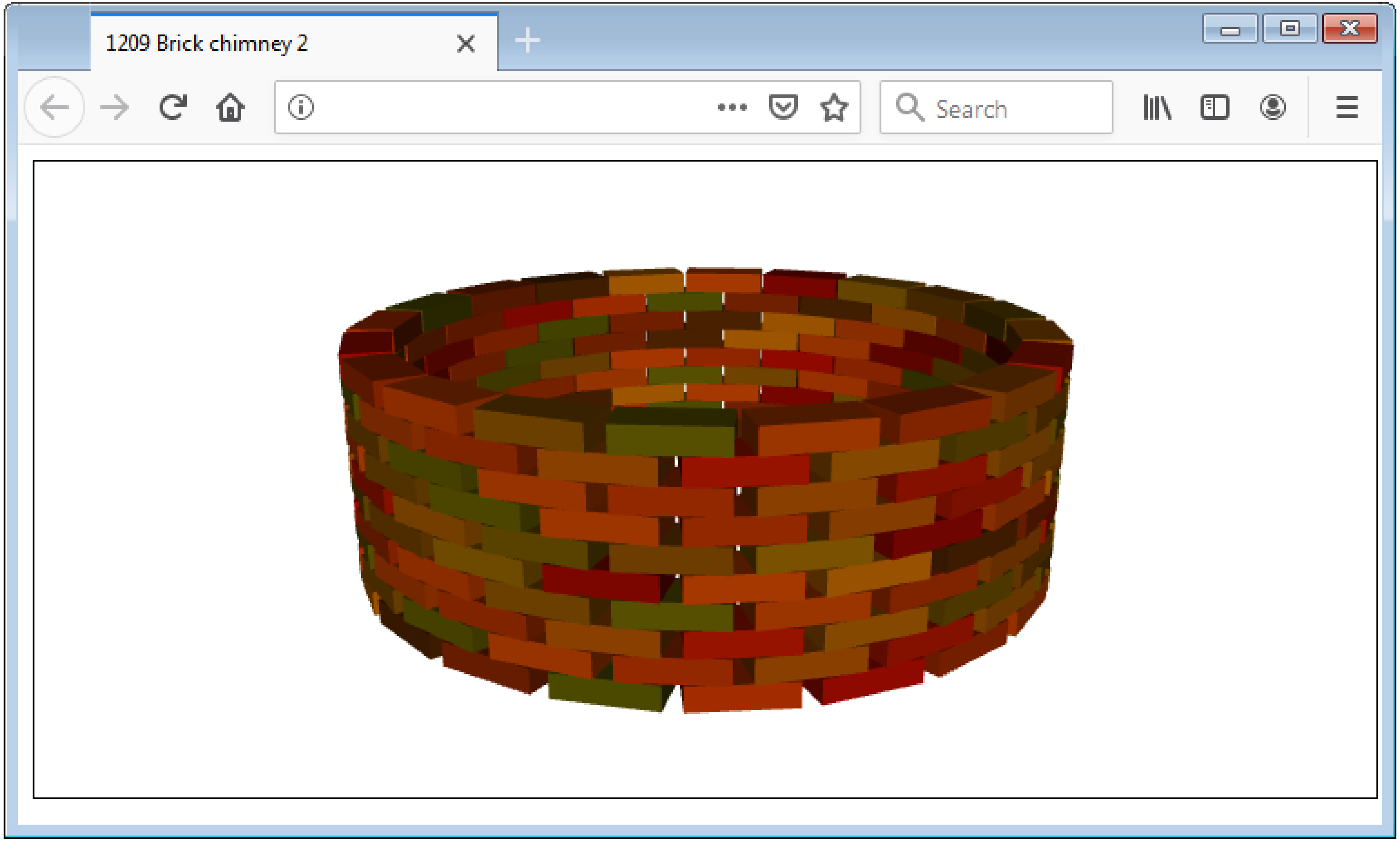


TRY IT

Building other rows

- Creating a copy of the row with sameAs
- Each copy has different z coordinate
- Each row is rotated on a multiple of 18°
- For overlapping each other row is rotated on additional 9°

```
for (var i=1; i<10; i++)
  sameAs(row).custom({
     center: [0,0,i],
     spin: radians(18*Math.round(random(0,20))+9*(i%2))
  });
```

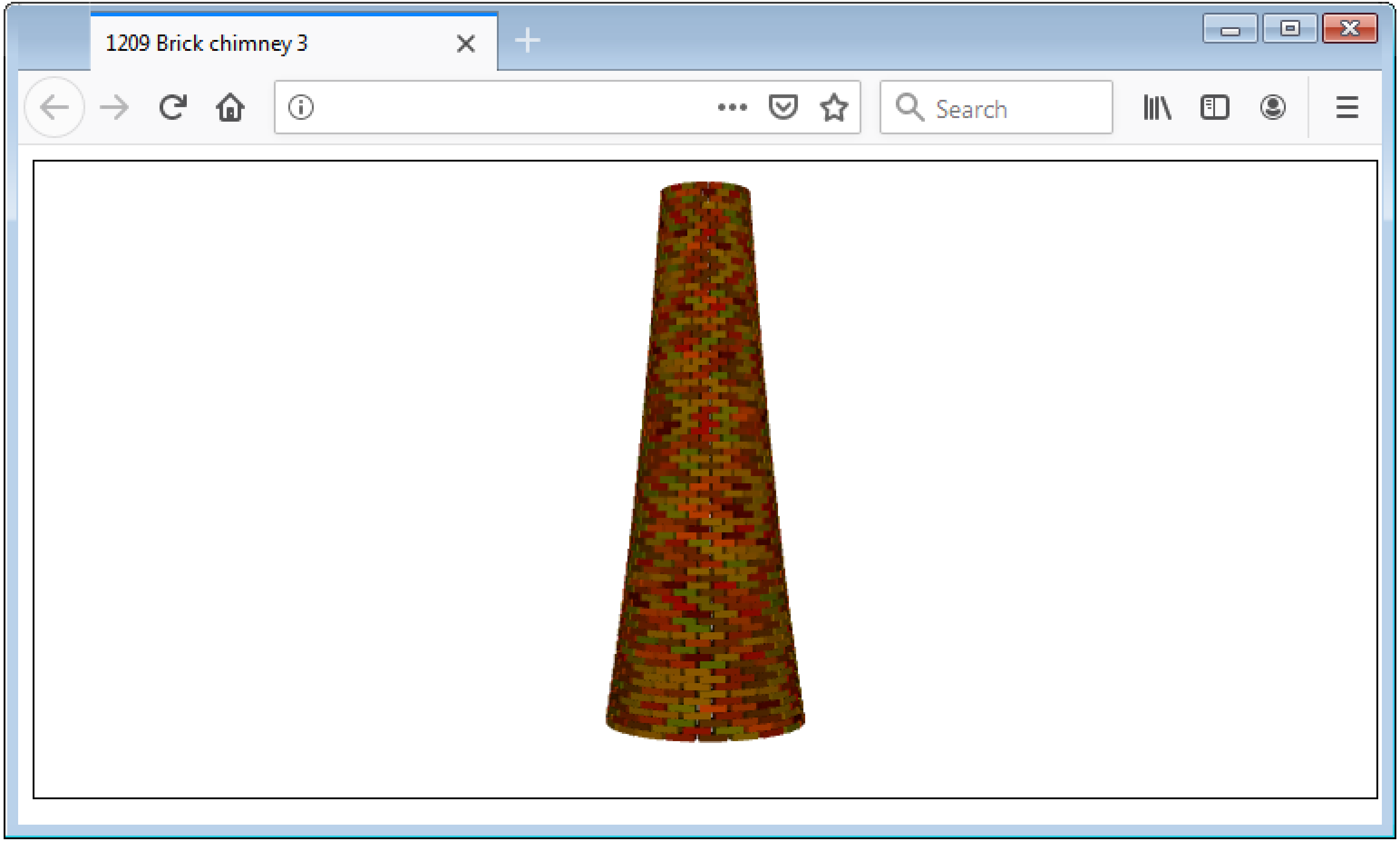


TRY IT

Shaping a chimney

- Each row of bricks is a group object
- Using property sizes to change the size
- Starting with scale k=1 and reducing it by 1% for each next row

```
var k = 1;
for (var i=1; i<80; i++)
{
   k = k*0.99;
   sameAs(row).custom({
      ...
      sizes: [k,k,1]
   });
}
```


1209 Brick chimney 3
Search

TRY IT

Clipping planes

Clipping planes

Parts of objects

- Some geometrical objects are part of other objects
- Examples
  - Truncated cone is a part of a cone
  - Semisphere is a part of a sphere

Generating

- With small triangles and other faces (slow)
- With extending the library (difficult)
- With intersecting an object with planes

Clipping planes

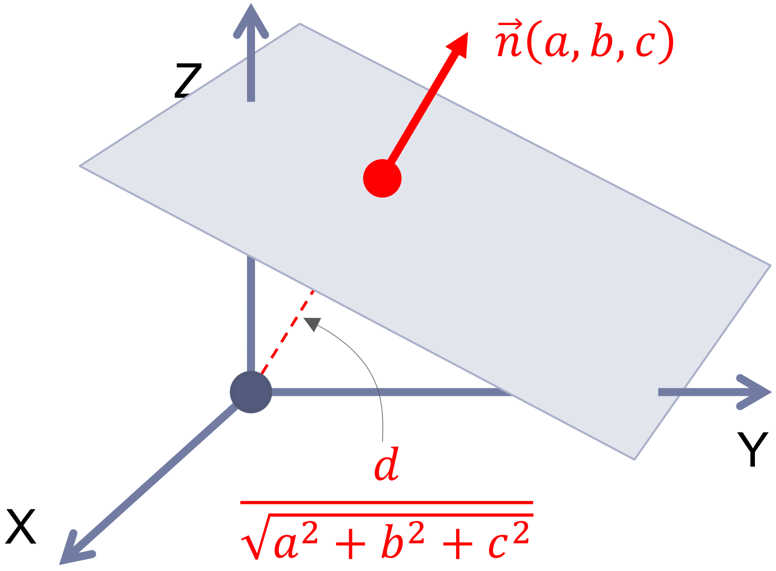
- Defining a plane with equation **ax+by+cz+d=0**
- The plan split the space in two supspaces
- Onle the part of the object in the positive subspace is drawn, this is **ax+by+cz+d>0**

Property clipPlanes

- Property clipPlanes is an array of 4 subarrays
- Each subarray defines one clipping plane $[[a_0,b_0,c_0,d_0], [a_1,b_1,c_1,d_1], [a_2,b_2,c_2,d_2], [a_3,b_3,c_3,d_3]]$
- The property applies only to base (i.e. non-group) objcts

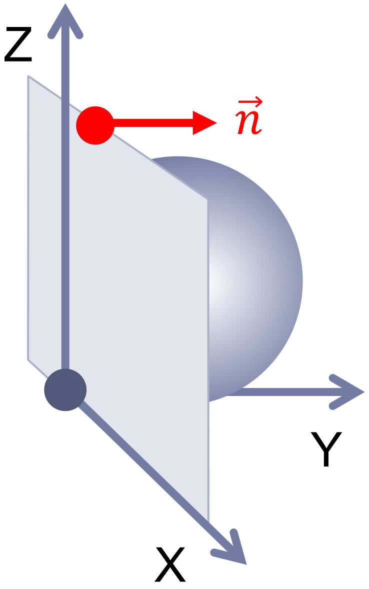
Coefficients

- Relative to the local coordinate system
- Define the normal vector
- Define the distance to the clipping plane



Example

- A scene of random spheres
- Clipping with a vertical XZ plane
- Drawing only what is in the +Y subspace



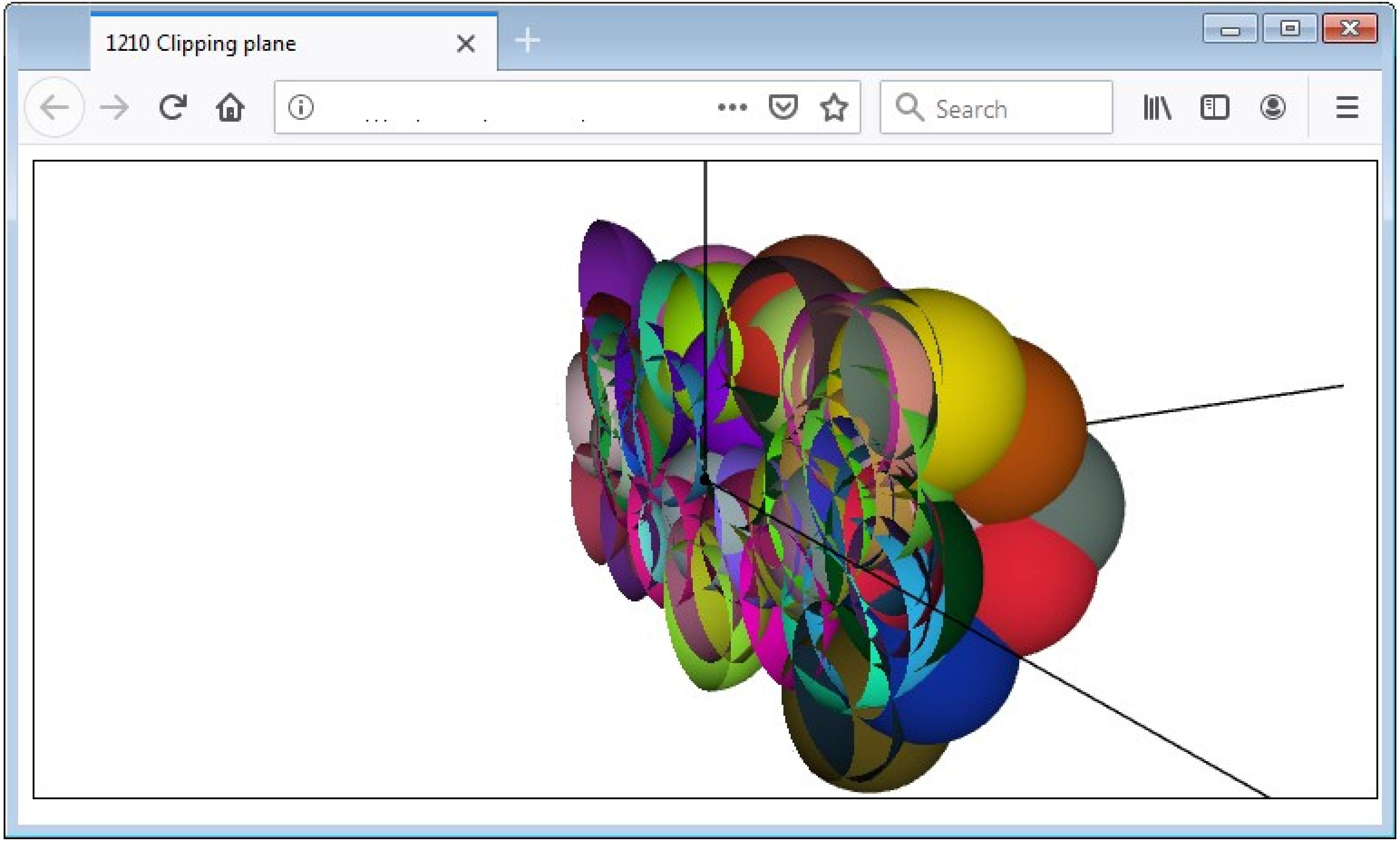
Solution

- The normal vector is +Y, i.e. (a,b,c)=(0,1,0)
- The plane goes through (0,0,0), i.e. d=0, however…

The local coordinates must be used :

- The sphere center is (0,0,0), its diameter is 1, the distance to the plane is y/5, thus d=y/5, instead of 0

```
var y = random(-5,5);
sphere([0,0,0],2.5).custom({
   center: [random(-10,10),y,random(-5,5)],
   color: [random(0,1),random(0,1),random(0,1)],
   clipPlanes: [[0,1,0,y/5]]
});
```



TRY IT

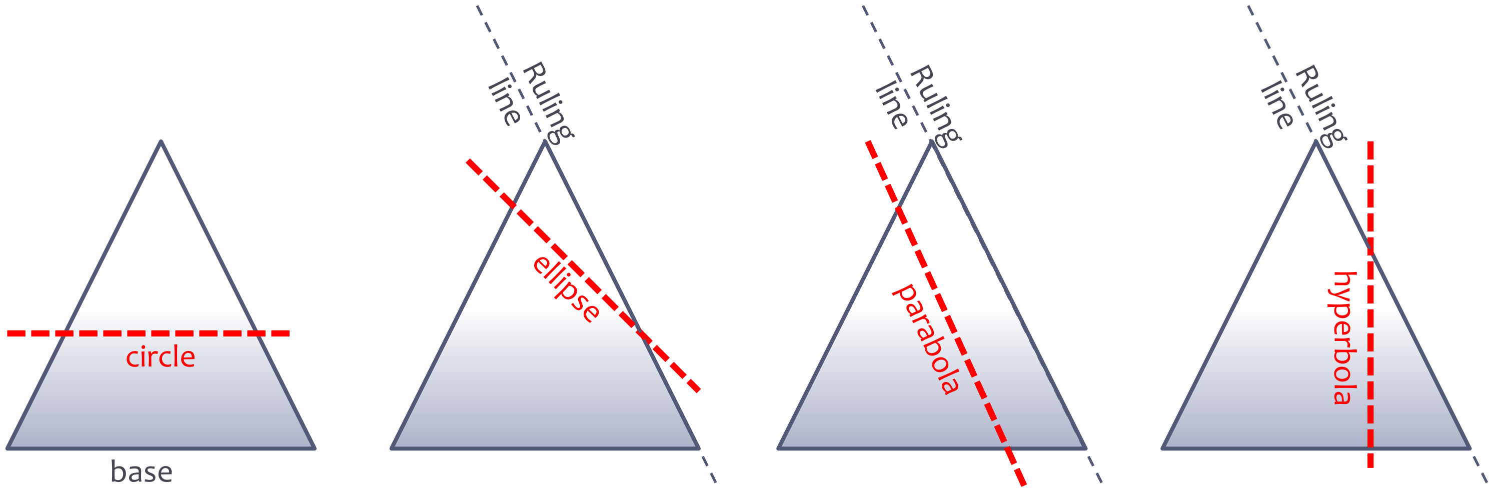
Conic sections

Illustration of conic sections

- Four cones
- Each is truncated with a plane
- The intersections are:
  - Circle
  - Ellipse
  - Parabola
  - Hyperbola

Clipping plane

- Circle – the plane is parallel to the base
- Ellipse – the plane is at a smaller angle than the ruling line
- Parabola – the plane is at the same angle as the ruling line
- Hyperbola – the plane is at a larger angle than the ruling line

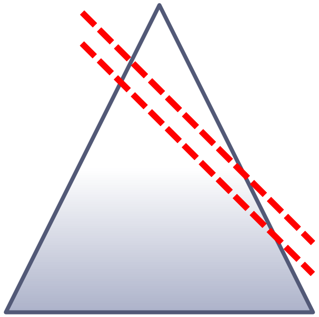


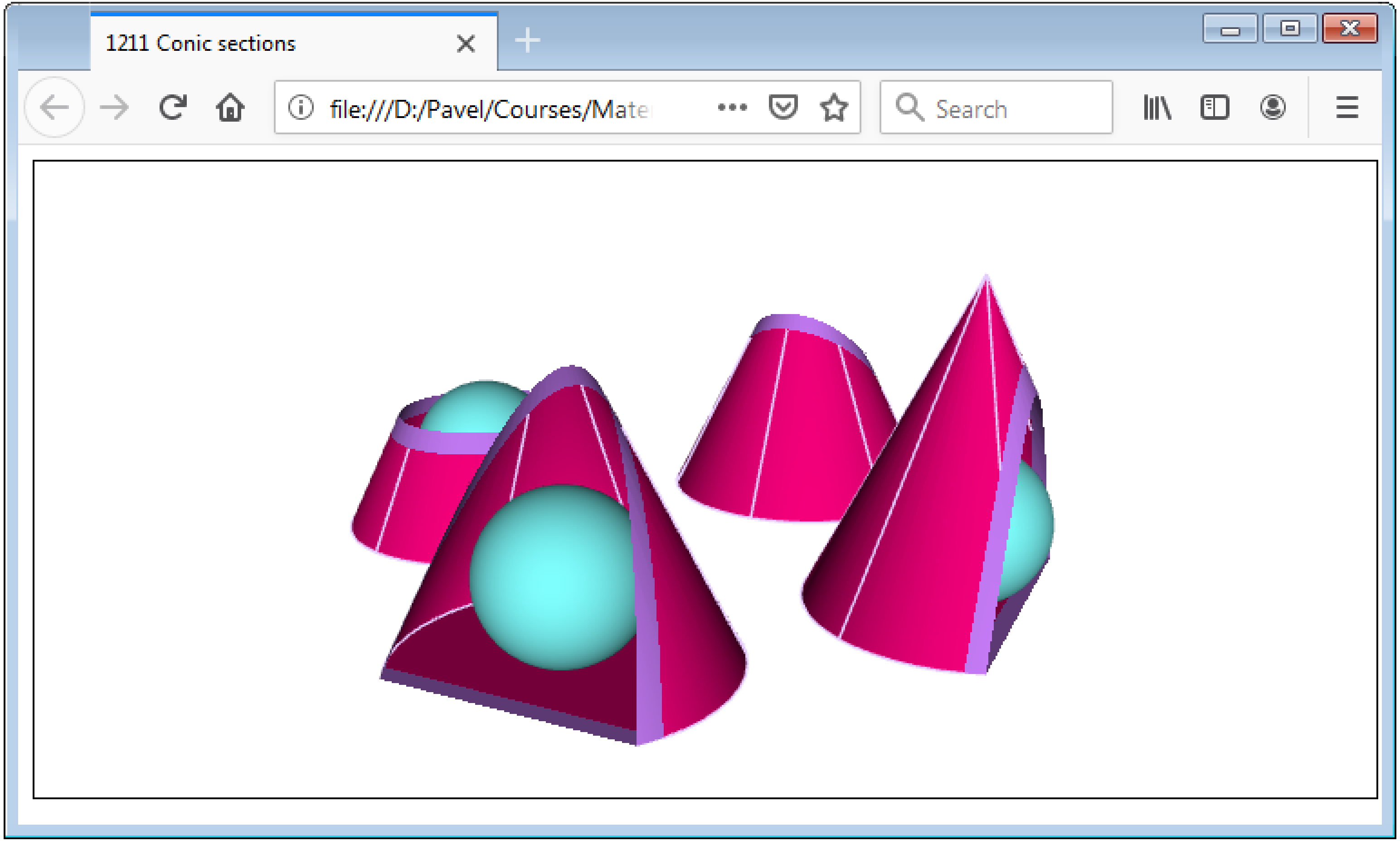
Solution

- Selecting coefficients of the clipping planes

```
cone(...).custom({clipPlanes: [[0,0,-1,1/3]]});
cone(...).custom({clipPlanes: [[0.8,0,-1,1/2]]});
cone(...).custom({clipPlanes: [[2,0,-1,1/2]]});
cone(...).custom({clipPlanes: [[1,0,0,1/8]]});
```

- Cloning objects, but in mode Suica.LINE, to draw the edges
- Cloning again + applying two clipping planes to draw a strip around the intersection





TRY IT

Summary

Graphical objects

Anonymous and named

- Anonymous, when no changes are needed
- Names, when their properties will be changed
- The same name can be reused for several objects

Creating styles

- Assigning values of object properties
- With custom, and a style as a set of pairs {name:value, …}

Copying / cloning

- With sameAs

Group objects

- With class new Suica.Group or function group
- Method add for adding objects to existing group
- Method mergeColor to define common colour

Clipping planes

- Property clipPlanes with coefficients of 1 to 4 planes

ICT in SES

The end

Comments, questions