ICT in SES

Drag and drop

Lesson №17

Following

Following the mouse

Links graphical object - mouse

- Object follows the mouse
- Object moves with the mouse

Following the mouse

- Easier to implement
- Dow not require selection of current object
- Convenient with orthographic projection

Types of links

Hard link

- Object linked to the mouse
- Moving exactly with the mouse
- Moving only when the mouse is moving

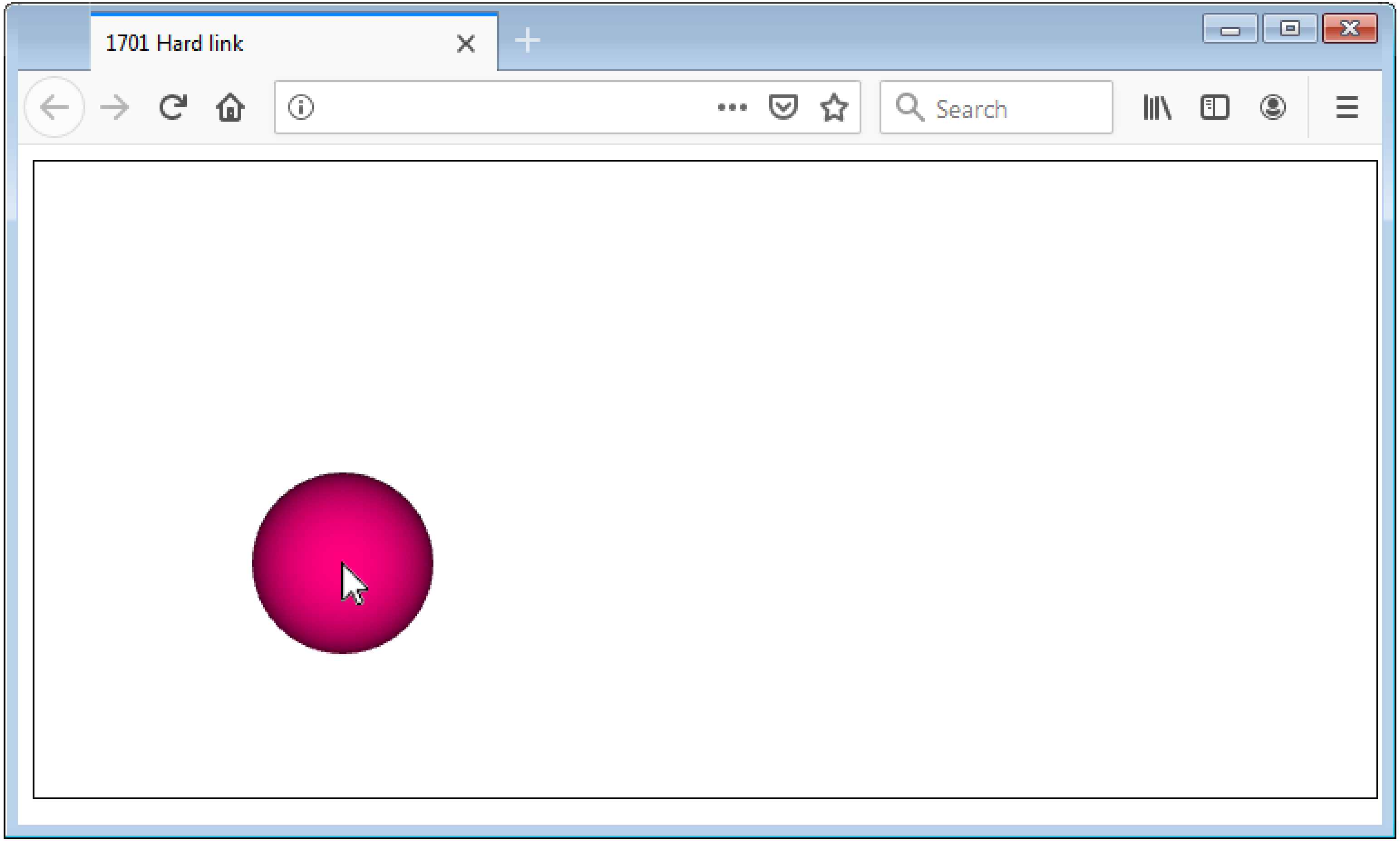
Soft link

- Object linked as with elastic thread
- Moving almost like the mouse
- Moving even if the mouse is not moving

Implementation of hard link

- Converting mouse coordinates to graphical coordinates that define object’s center

```
function mouseMove(event)
{
   var x = event.clientX
           - event.target.offsetLeft
           - event.target.offsetWidth/2;
   var y = -(event.clientY
             - event.target.offsetTop
             - event.target.offsetHeight/2);
   s.center = [x,y,0];
}
```

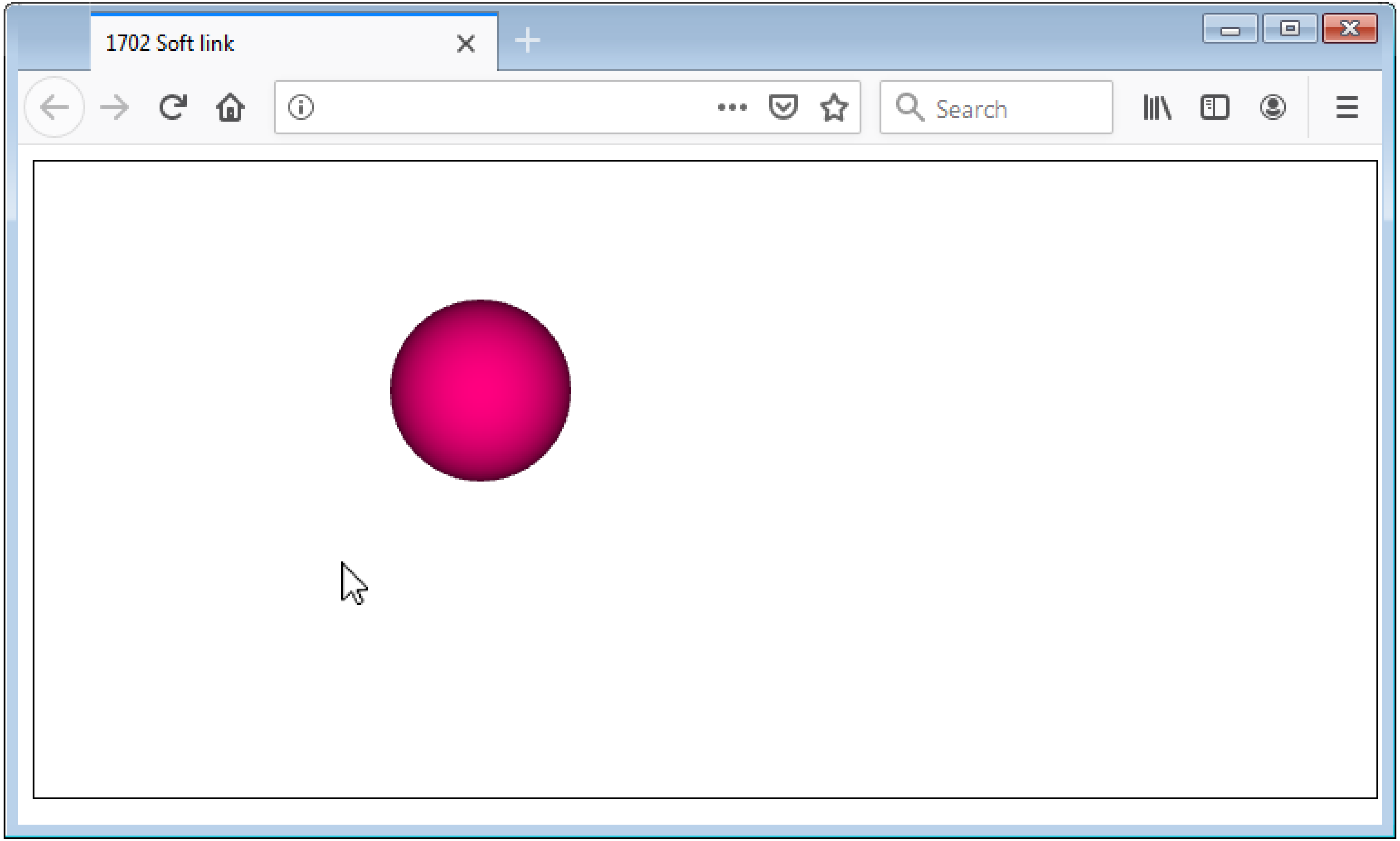


TRY IT

Implementation of soft link

- Graphical coordinates are recorder in mouseMove into global variables x and y
- The loop animate moves the object with linear combination towards the recorded x and y

```
var x=0, y=0;
function mouseMove(event) { x=...; y=...; }
function animate()
{
   var k = 0.92;
   s.center[0] = s.center[0]*k+(1-k)*x;
   s.center[1] = s.center[1]*k+(1-k)*y;
}
```

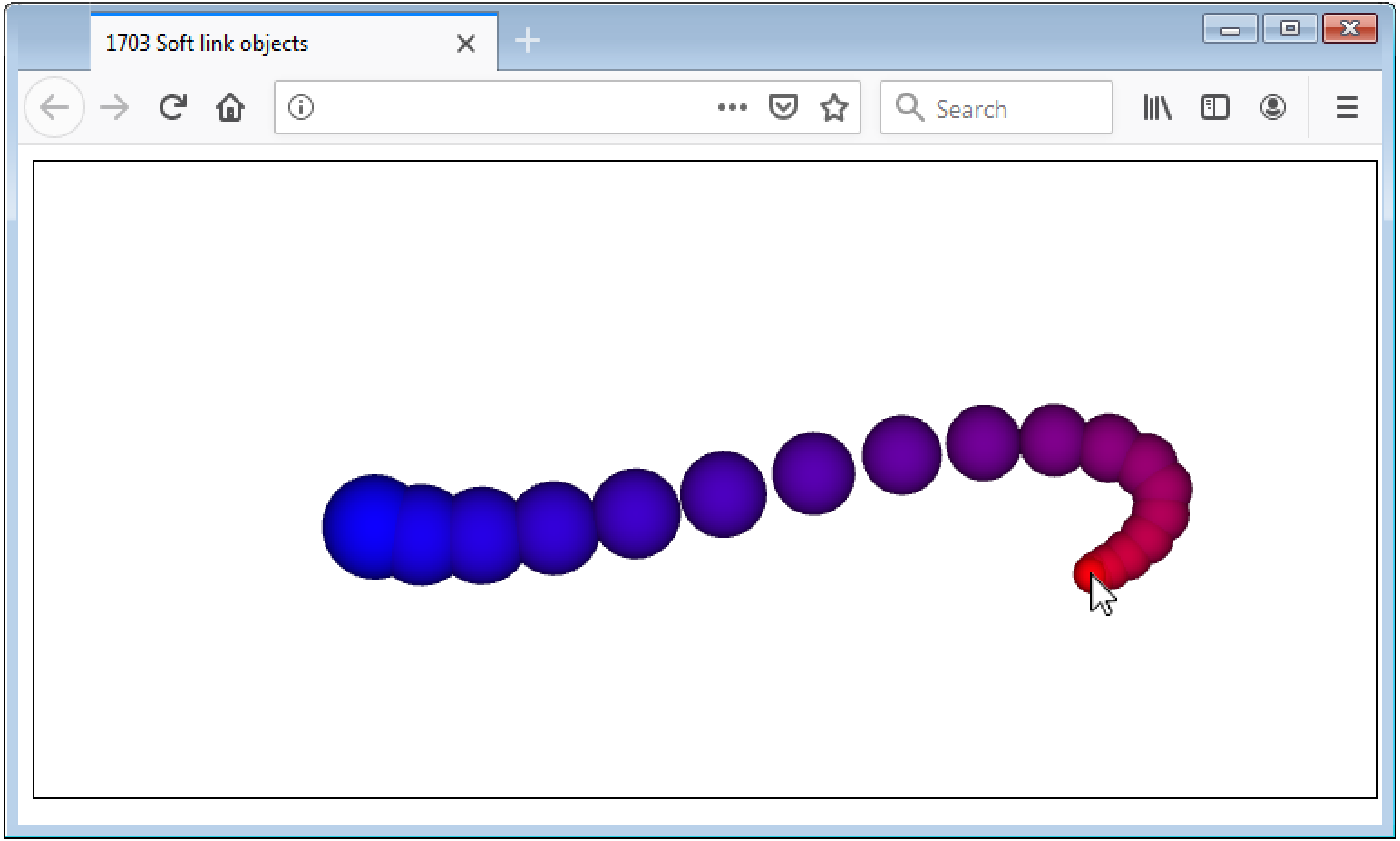


TRY IT

A chain of soft links

- Several soft-chained objects
- Similar implementation, with linear combination

```
function mouseMove(event) {s[0].center = ...; }
function animate()
{
   var k = 0.85;
   for (var i=1; i<n; i++)
   {
      s[i].center[0] = s[i].center[0]*k+(1-k)*s[i-1]...
      s[i].center[1] = s[i].center[1]*k+(1-k)*s[i-1]...
   }
}
```



TRY IT

Object selection

Selection with mouse

Using selection with mouse

- Using an object to manipulate
- This includes selecting an object to drag

Problem

- View point is not fixed
- Objects may have irregular shapes
- Objects may contain other objects

Calculated selection

Idea

- Calculating the screen area of each objects
- Checking whether mouse cursor is in this area

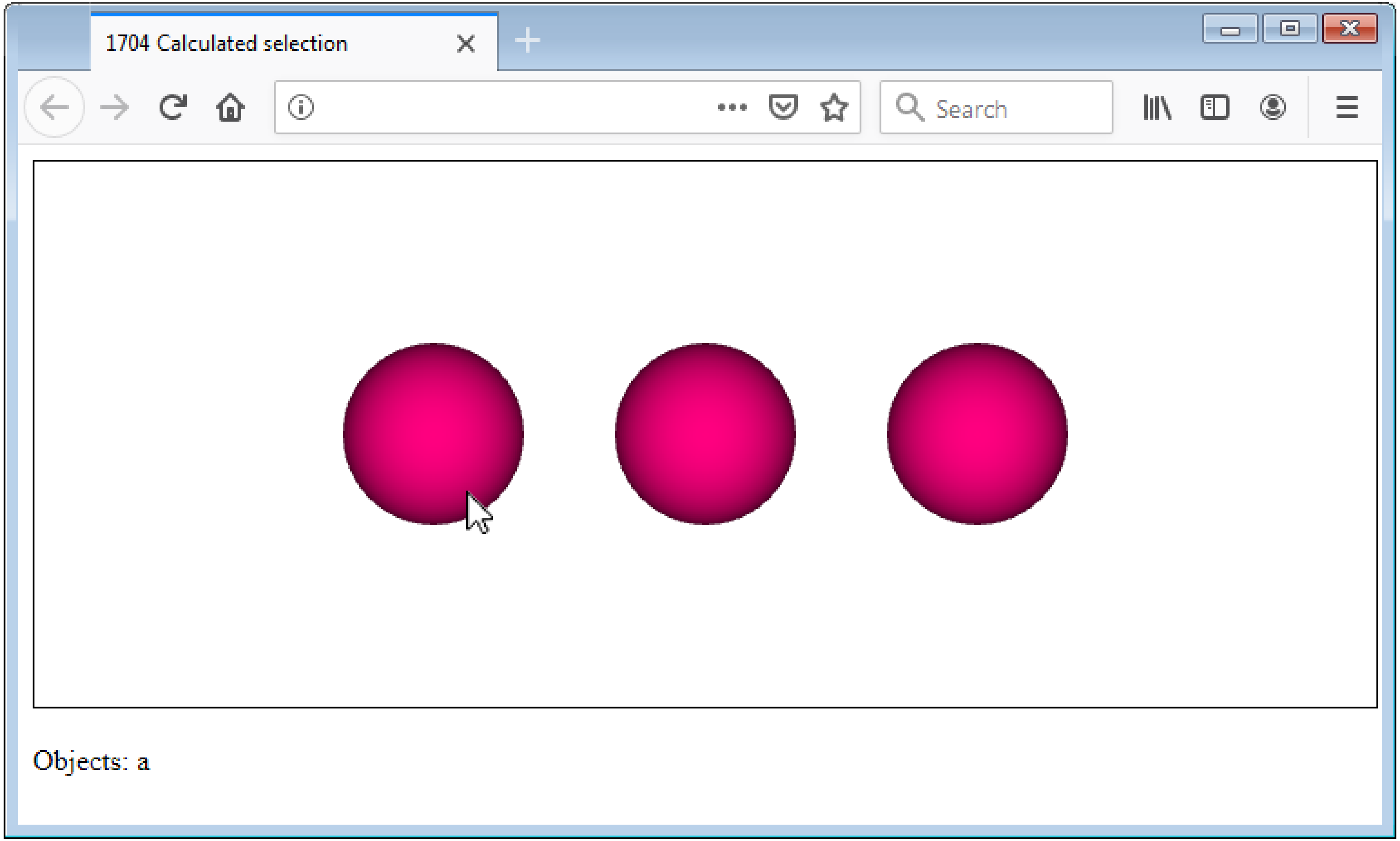
Applicability

- Mostly when calculations are not difficult
- Convenient projection and view point
- Suitable shape, position and orientation of objects

Example

- Three spheres, orthographic projection, view point on Z
- Calculating distances between cursor and spheres’ centers
- Showing the name of the selected sphere

```
if ( distance(a.center,[x,y])<=50 )
   obj.innerHTML = 'a';
else
if ( distance(b.center,[x,y])<=50 )
   obj.innerHTML = 'b';
else
if ( distance(c.center,[x,y])<=50 )
   obj.innerHTML = 'c';
```



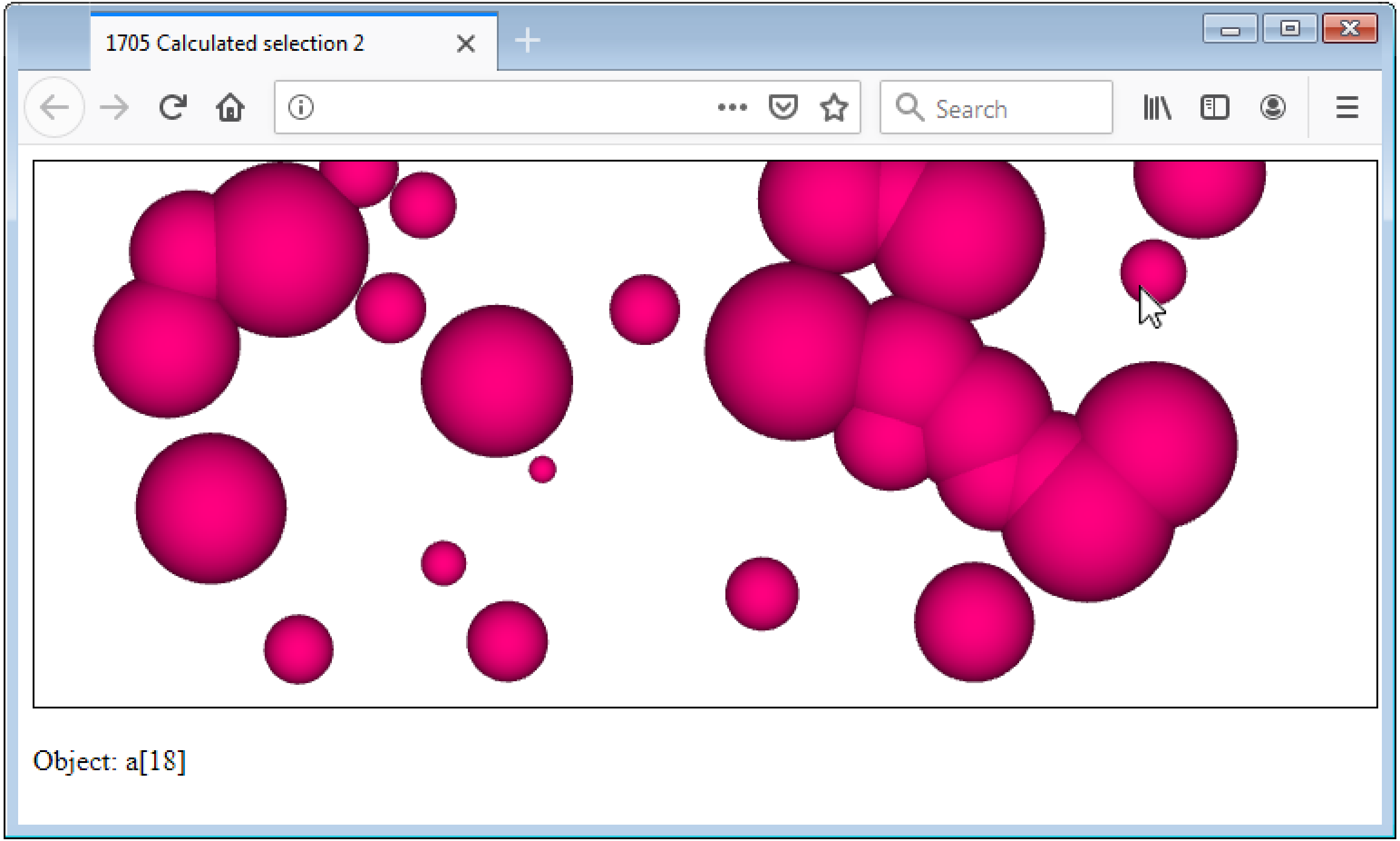
Objects: a

TRY IT

Modified example

- Many spheres, different sizes
- Same idea – calculating distances and comparing with radii

```
for (var i=0; i<n; i++)
  if ( distance(a[i].center,[x,y])<=a[i].radius )
     obj.innerHTML = 'a['+i+']';
```



Object: a[18]

TRY IT

Any shape

Selecting object with any shape

- With objectAtPoint (x, y)
- Returning the object at give pixel or null, if there are no object
- Coordinates x and y are clientX and clientY of mouse events

Important

- The method checks only objects which property interactive is set to true

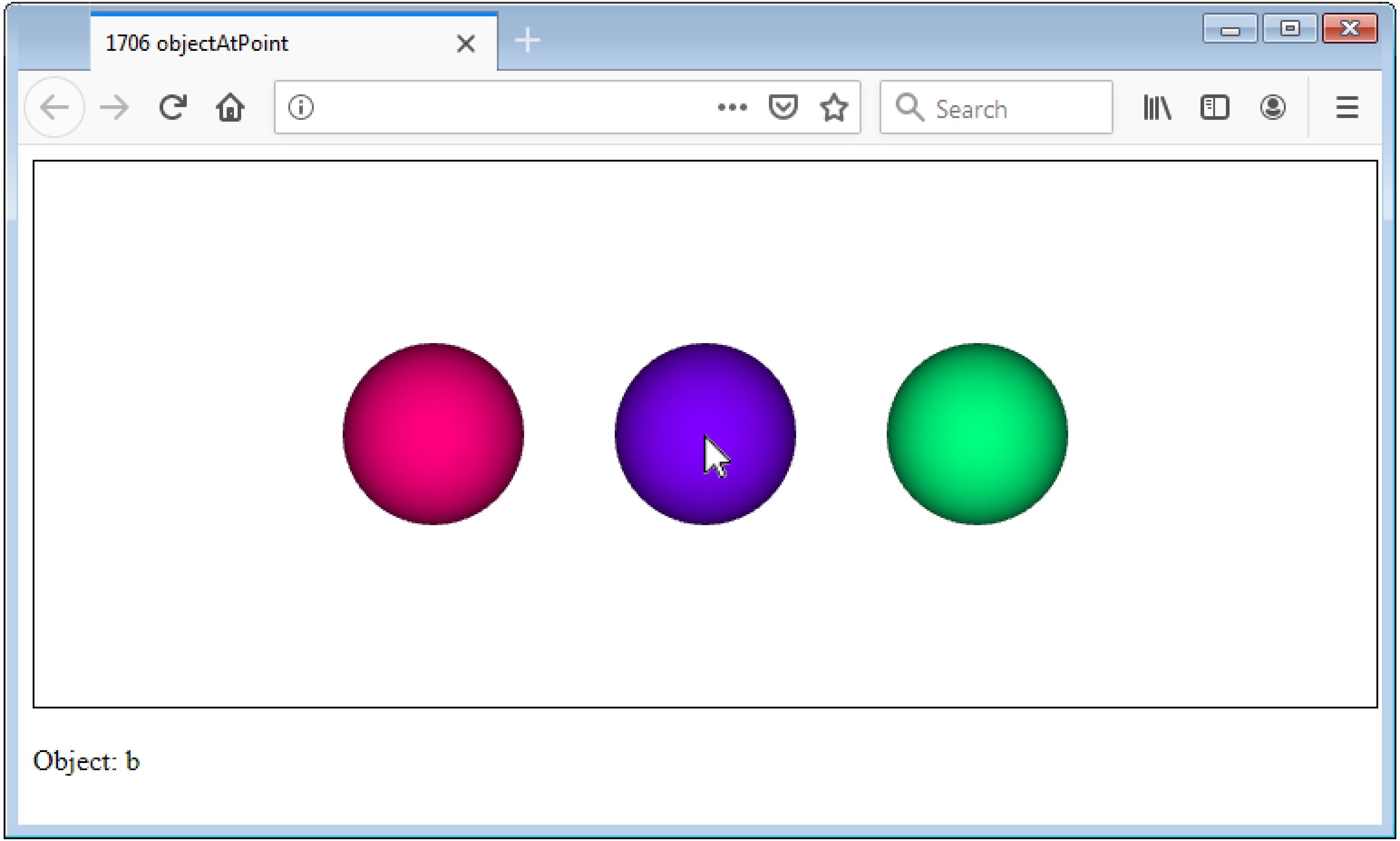
Note

- Result null is also produces when it is not possible to uniquely identify a single object
- Happens when pixel's colour is generated from several sources:
  - A contour pixel partly inherits the colour from the background
  - A colour of pixel on the boundary of two objects contains portions of both their colours

Example

- Three sphere with turned on interactive
- Selecting object with objectAtPoint and coordinates from the event e

```
p = new Suica();
...
a = sphere([-150,0,0],50).custom({
     info: 'a',
     interactive: true});
...
function mouseMove(e)
{  var o = p.objectAtPoint(e.clientX,e.clientY);
   if (o) obj.innerHTML = o.info;
   ...}
```



Object: b

TRY IT

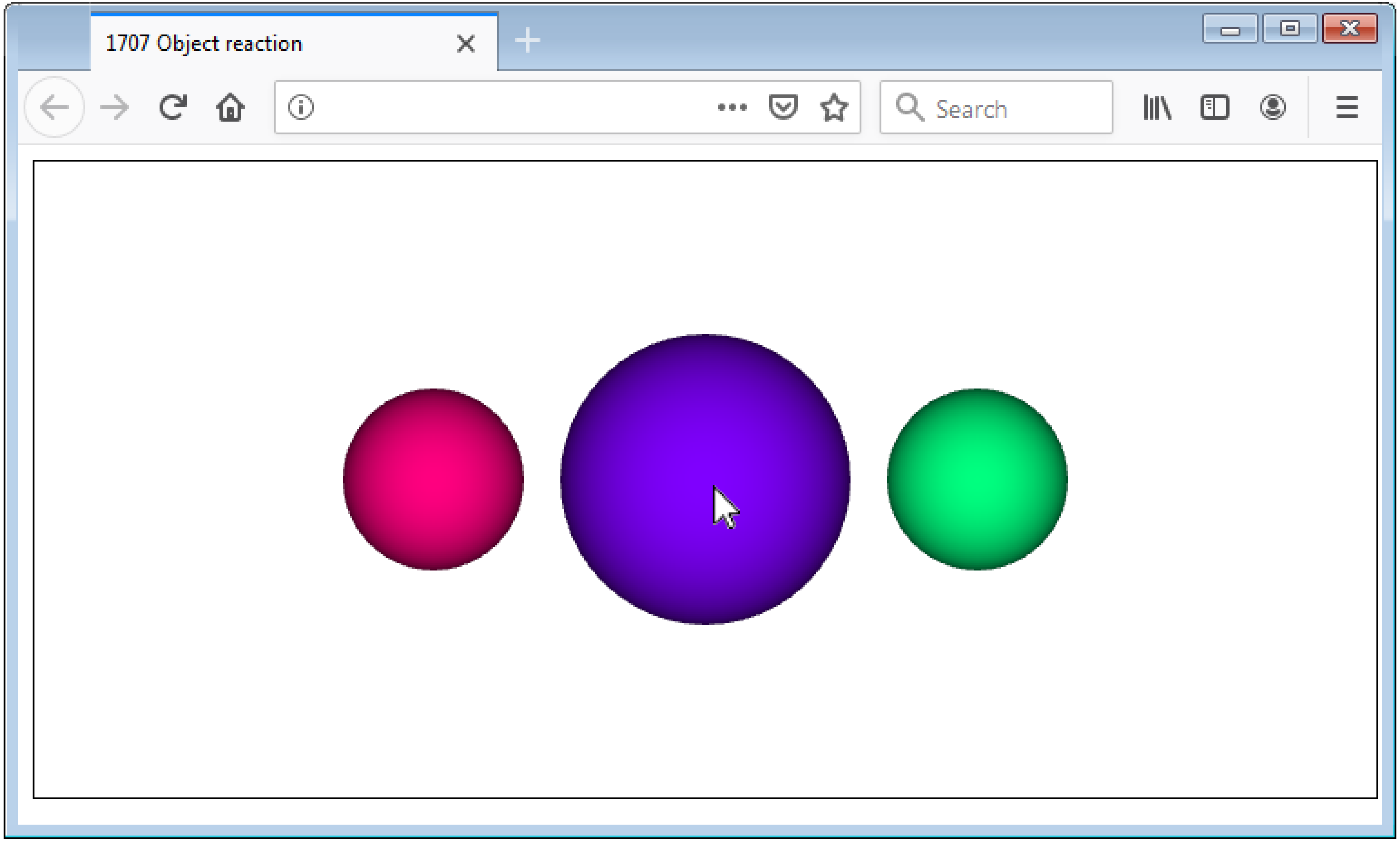
Object reaction

- Currently selected sphere lastObj is larger
- When a new sphere is selected in newObj, the old one reverts its size

```
var lastObj;

function mouseMove(event)
{
   var newObj = p.objectAtPoint(...);

   if (lastObj) lastObj.radius = 50;
   lastObj = newObj?newObj:null;
   if (lastObj) lastObj.radius = 80;
}
```


1707 Object reaction
Search

TRY IT

Example

Ring of columns

- Cuboids in a circle
- All have the same height
- When the mouse cursor hover over a cuboid, it becomes short
- All short cuboids grow to their initial height
- The scene is rotating continuously

Implementation

- Columns are cuboids with modified origin
- Rotation with spin places them on a circle
- All objects are interactive and can be selected by objectAtPoint

```
n = 50;
a = [];
for (var i=0; i<n; i++)
  a.push( cuboid([0,0,-5],[1,1,15]).custom({
             interactive: true,
             origin: [10,0,-0.5],
             spin: i/n*2*Math.PI,
       }));
```

- Scene rotation with lookAt (time slowed down 4 times)

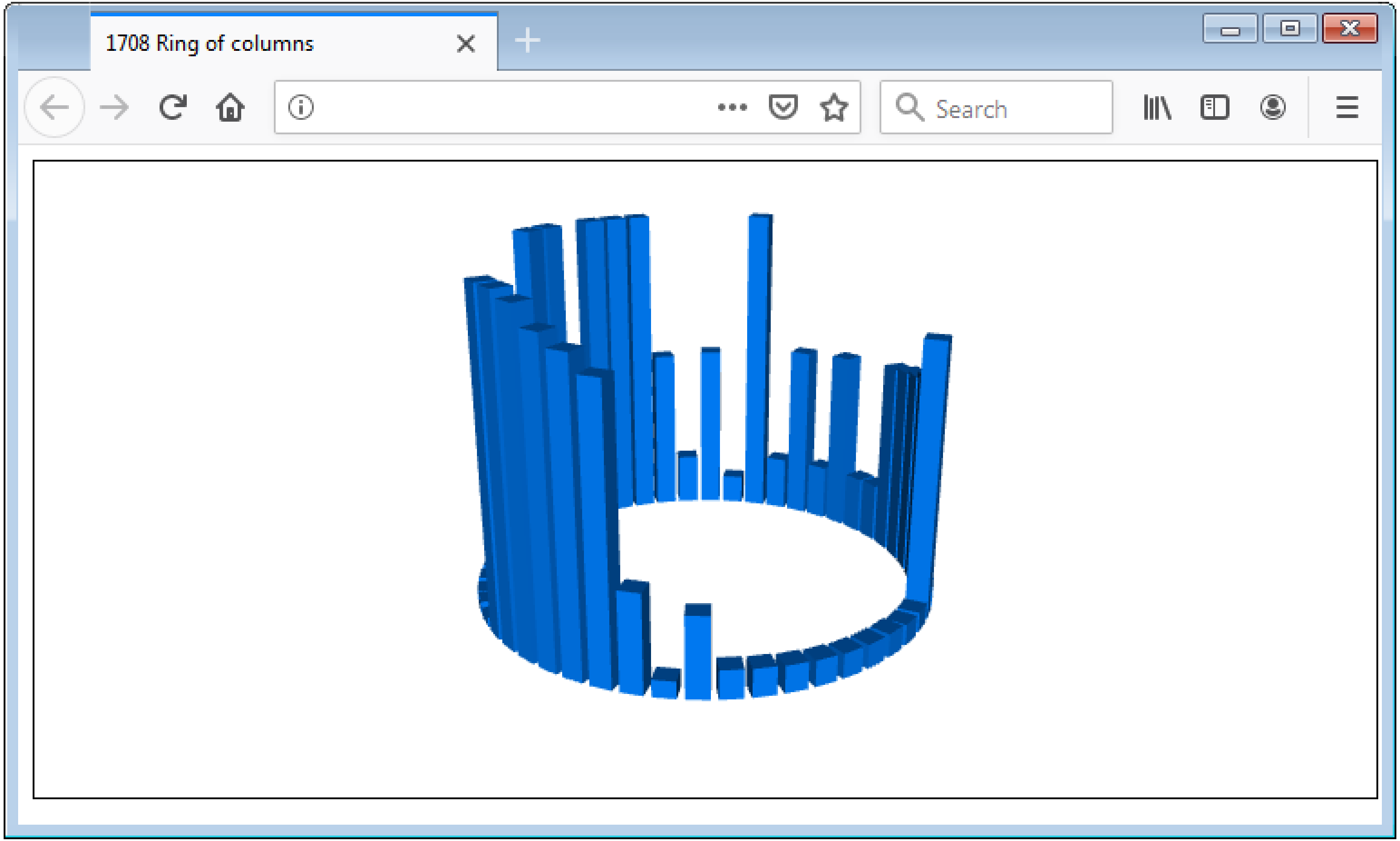
```
var t = Suica.time/4;
lookAt ([50*Math.cos(t),50*Math.sin(t),20],...);
```

- Gradual grow of cuboids by 2% per frame, applied for heights less than 15 units

```
for (var i=0; i<n; i++)
   if (a[i].sizes[2]<15)
      a[i].sizes[2] *= 1.02;
```

- Looking for object `obj` when mouse moves
- If there is object – make it short

```
var obj = p.objectAtPoint(event.clientX,...);
if (obj) obj.sizes[2] = 0.1;
```



TRY IT

Drag and drop

Phases

Phases of dragging and dropping

- Pressing a button – selecting an object
- Mouse motion – changing an object
- Releasing a button – dropping an object

Combining actions at button pressing

- If an object is grabbed, start its dragging
- If no object is grabbed, start rotating the scene

Naïve implementation

Events

- Capturing events mousedown, mouseup and mousemove
- Selecting object in obj when mouse button is pressed

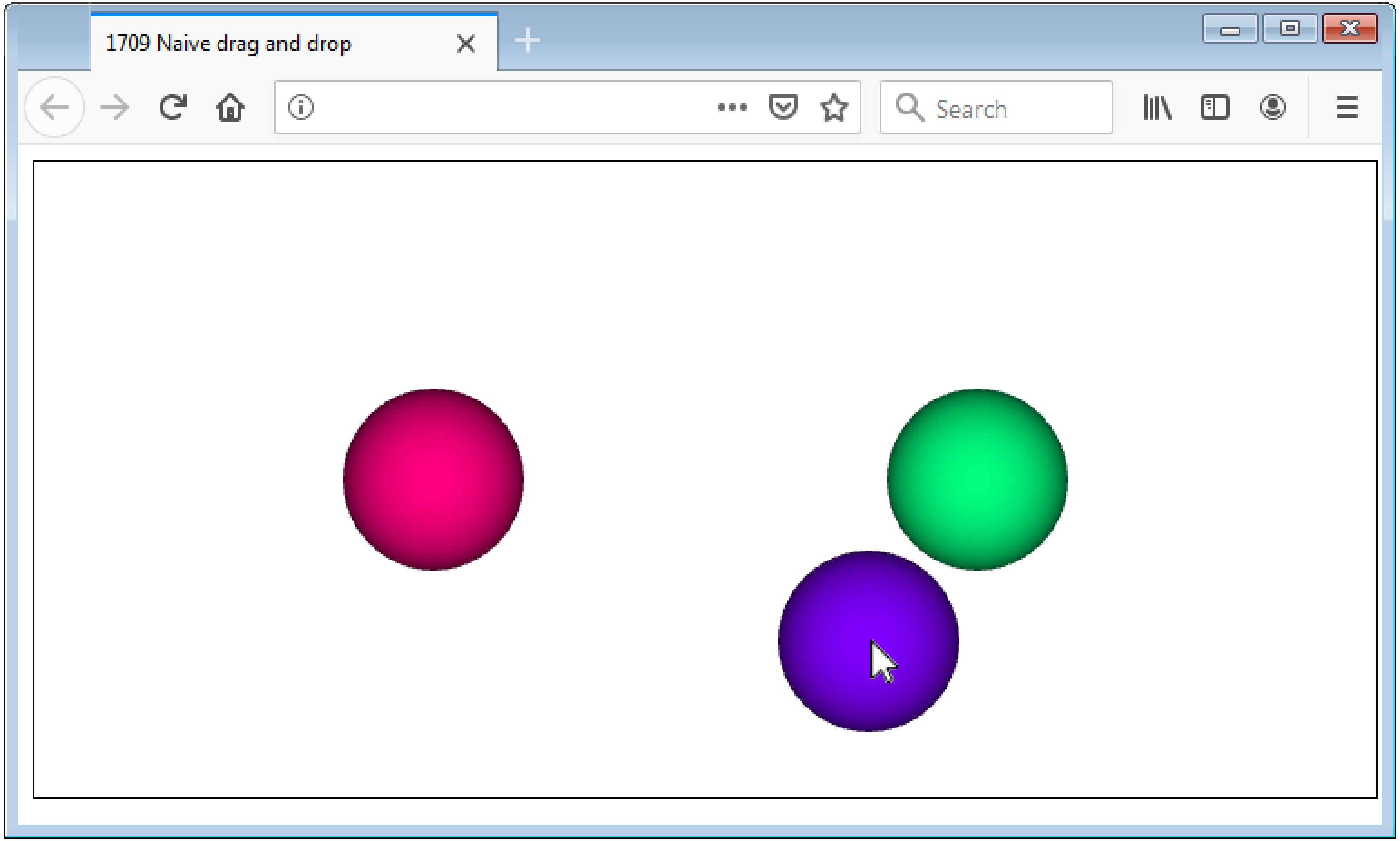
```
p.gl.canvas.addEventListener('mousedown',...);
p.gl.canvas.addEventListener('mouseup',...);
p.gl.canvas.addEventListener('mousemove',...);

function mouseDown(event)
{
   obj=p.objectAtPoint(event.clientX,event.clientY);
}
```

Events

- Forgetting selected object when mouse button is released
- While moving, if there is selected object, change its center

```
function mouseUp(event)
{
   obj = undefined;
}
function mouseMove(event)
{
   var x = ...;
   var y = -(...);
   if (obj) obj.center = [x,y,0];
}
```

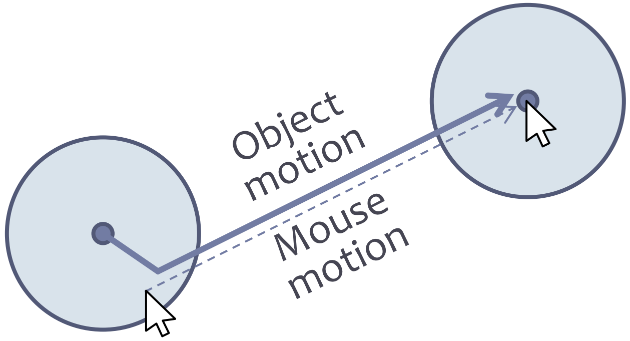


TRY IT

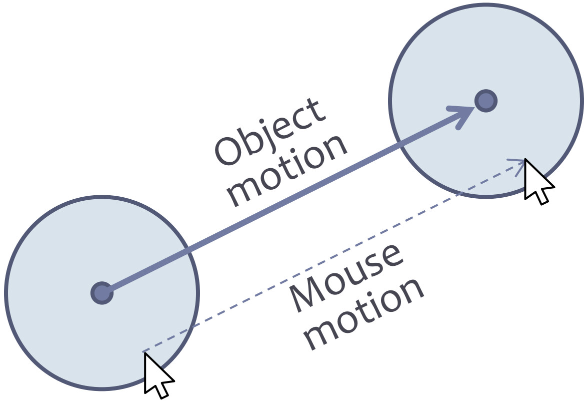
Problem

Dragging is not natural

- Object always centered on the cursor



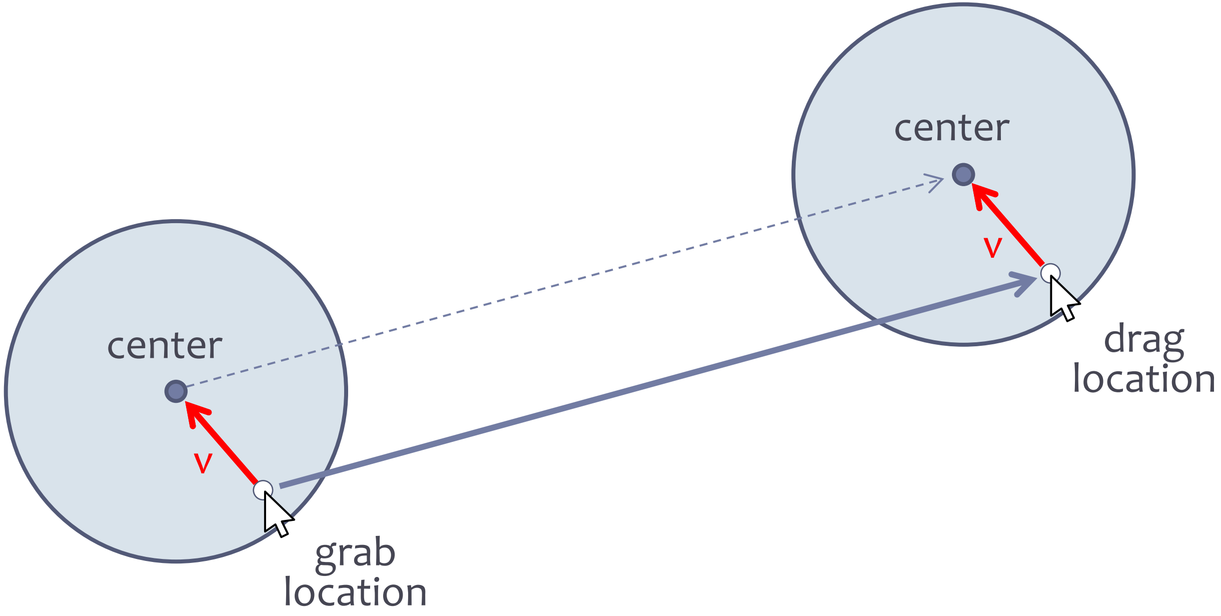
Naïve dragging



Desired dragging

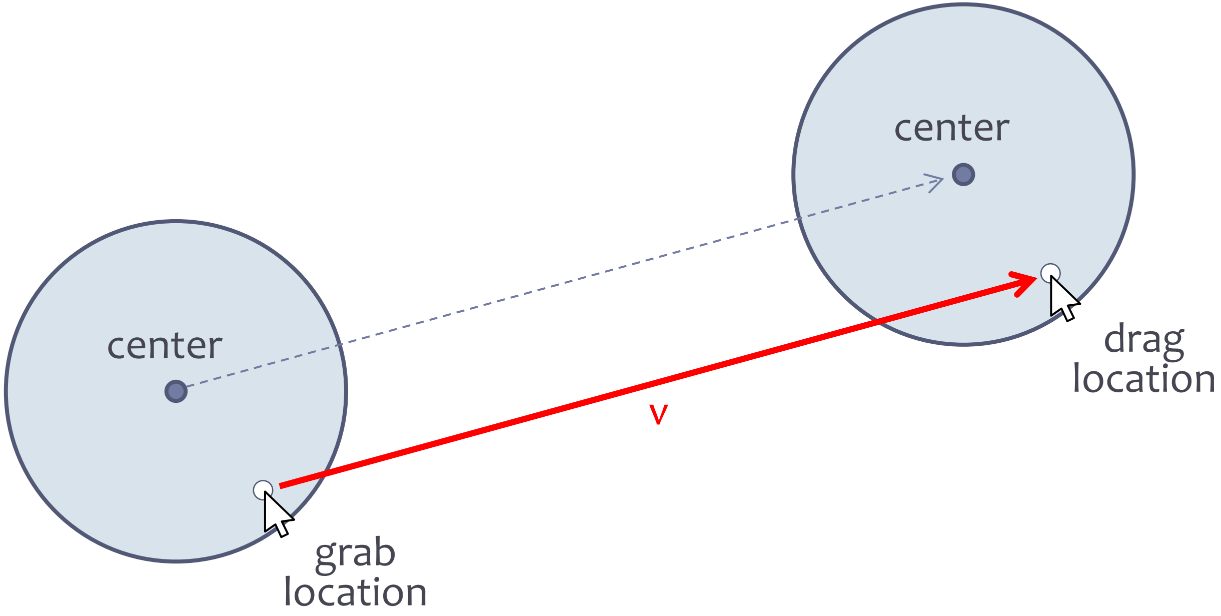
Solution №1

- Vector v from the grab location to the center
- Calculating the center from the drag location using this vector



Solution №2

- Vector v from the grab location to drag location
- Moving the center with this vector



Comparison

Solution №1	Solution №2
Vector v is calculated once at the beginning of the drag	Vector v is calculated at every step of the drag
No need to remember the last coordinates	There is need to remember the last coordinates
Useful for dragging that depends on the overall offset	Useful for dragging that depends on relative offset
Makes the traditional dragging easier	Makes additional effects easier (e.g. inertia)

Dragging

Implementation of solution №2

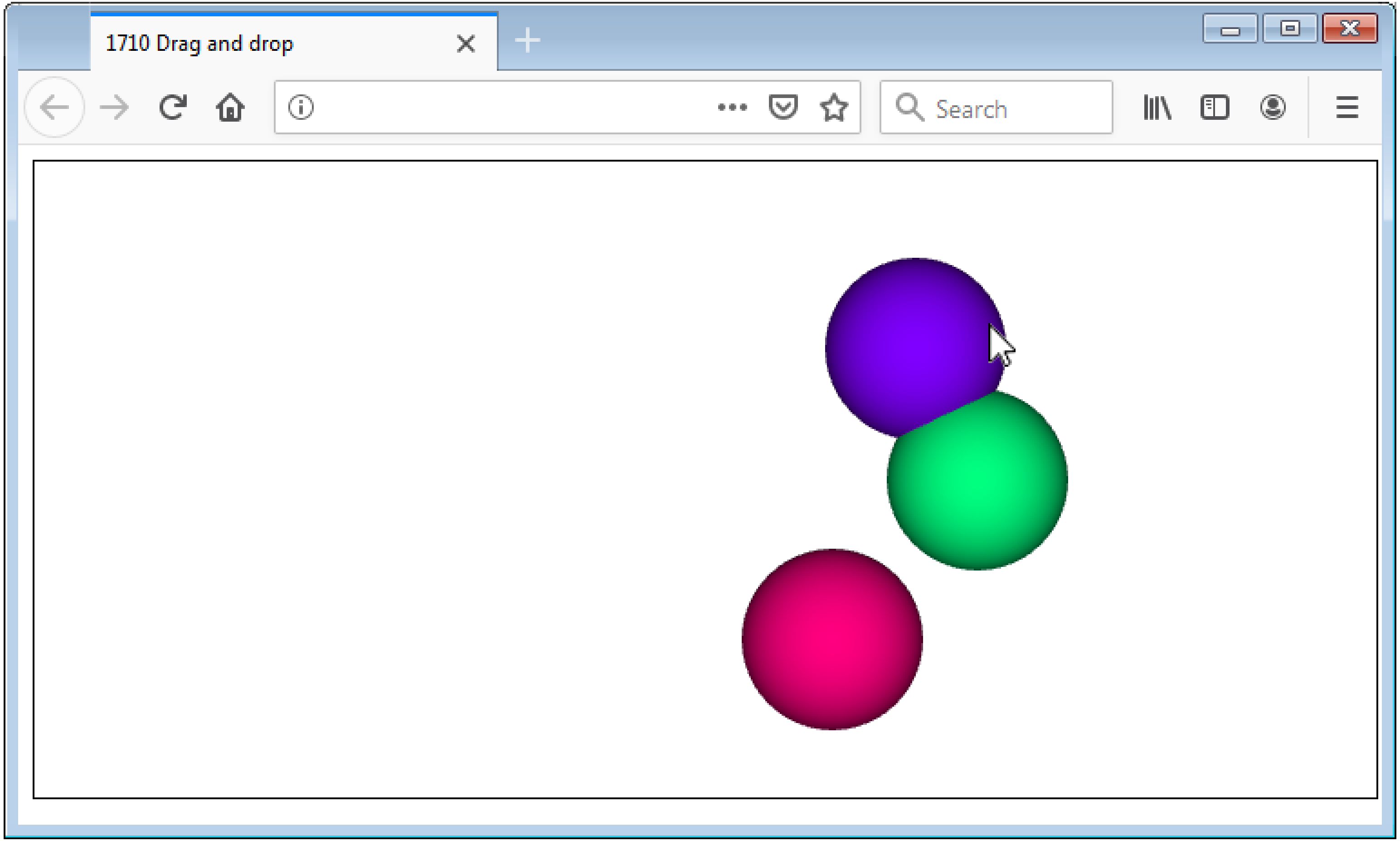
- Finding object in obj when a button is pressed
- Remembering coordinates x and y – no conversion to local coordinates, working with relative motion only

```
function mouseDown(event)
{
   x = event.clientX;
   y = event.clientY;
   obj = p.objectAtPoint(x,y);
}
```

- During motion only the center is updated, in respect to the cursor’s offset
- Subtracting in Y because screen Y and graphical Y have opposite directions
- Remembering the last x and y

```
function mouseMove(event)
{
   obj.center[0] += event.clientX-x;
   obj.center[1] -= event.clientY-y;

   x = event.clientX;
   y = event.clientY;
}
```



TRY IT

Dragging a scene

2D interactivity

Goal

- Having a 2D scene
- Sliding the scene interactively
- Scaling the scene interactively

Interface

- Sliding with the left mouse button
- Scaling with vertical motion and the right mouse button

Implementation

- Using mousedown and mousemove, without mouseup
- Removing the context menu with contextmenu
- When a button is pressed just store coordinates x and y

```
...addEventListener('mousedown',mouseDown,false);
...addEventListener('mousemove',mouseMove,false);
...addEventListener('contextmenu',
       function(e){e.preventDefault();},false);

function mouseDown(event)
{
  x = event.clientX;
  y = event.clientY;
}
```

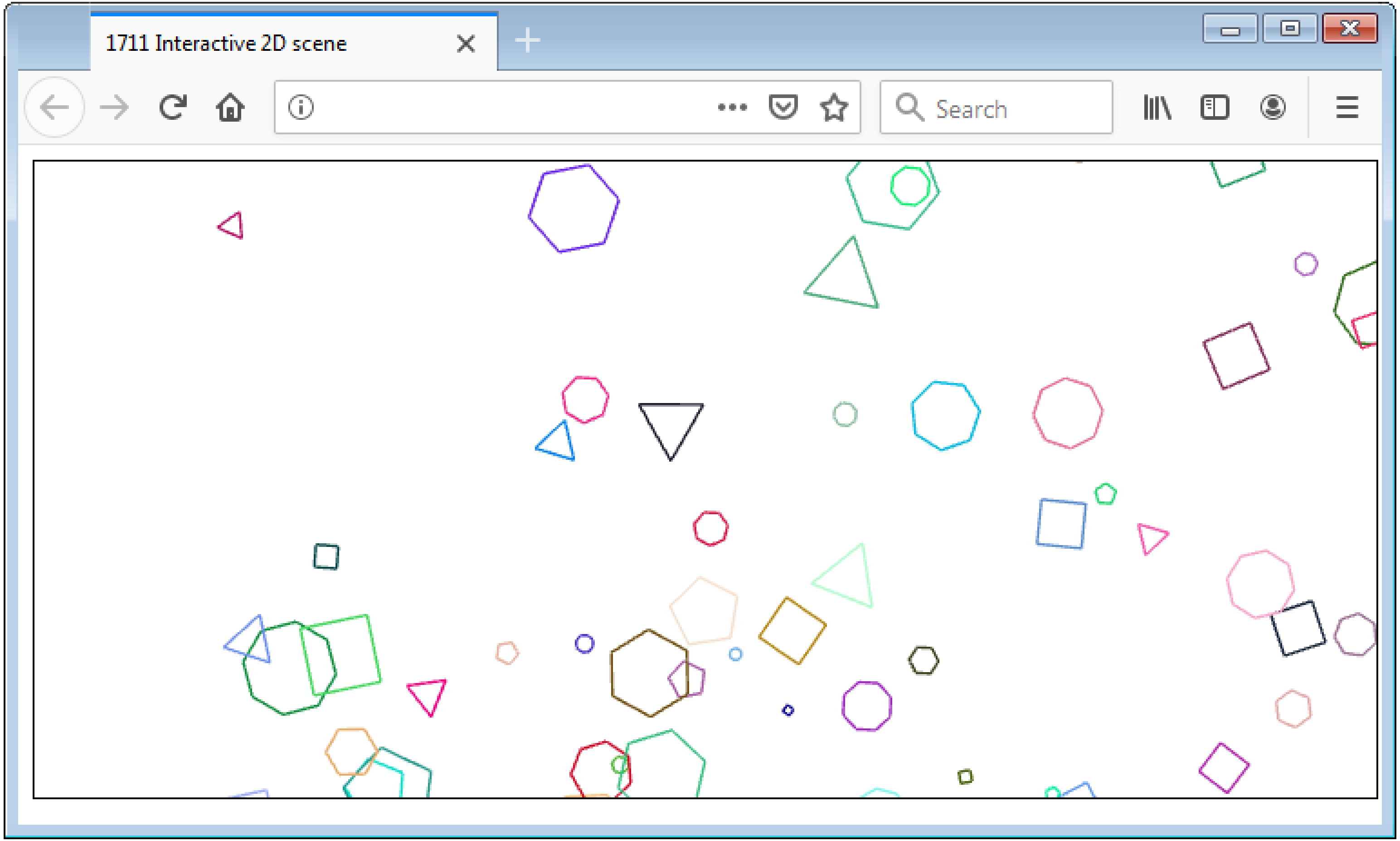
Mouse motion

- The view point is defined by lookX and lookY
- Scaling is implemented as distancing based on lookS

```
function mouseMove(event)
{
   ...
   lookAt ([lookX,lookY,lookS*650],
           [lookX,lookY,0], [0,1,0]);
   x = event.clientX;
   y = event.clientY;
}
```

- Pressing the left button transfers offset to lookX and lookY
- Offset is scaled by lookS
- Pressing the right button changes the scale factor lookS

```
if (event.buttons==1)
{
   lookX -= lookS*(event.clientX-x);
   lookY += lookS*(event.clientY-y);
}
if (event.buttons==2)
{
   lookS *= Math.pow(1.01,event.clientY-y);
}
```



TRY IT

3D interactivity

Goal

- Having a 3D scene
- Rotating the scene interactively
- Scaling the scene interactively

Interface

- Rotating with the left mouse button
- Scaling with vertical motion and the right mouse button

Implementation

- View point on a sphere with radius lookD
- Angular coordinates in lookA and lookB
- Target fixed at (0,0,0), up vector fixed to (0,0,1)

```
lookAt ( [lookD*cos(lookA)*cos(lookB),
          lookD*sin(lookA)*cos(lookB),
          lookD*sin(lookB)], [0,0,0], [0,0,1]);
```

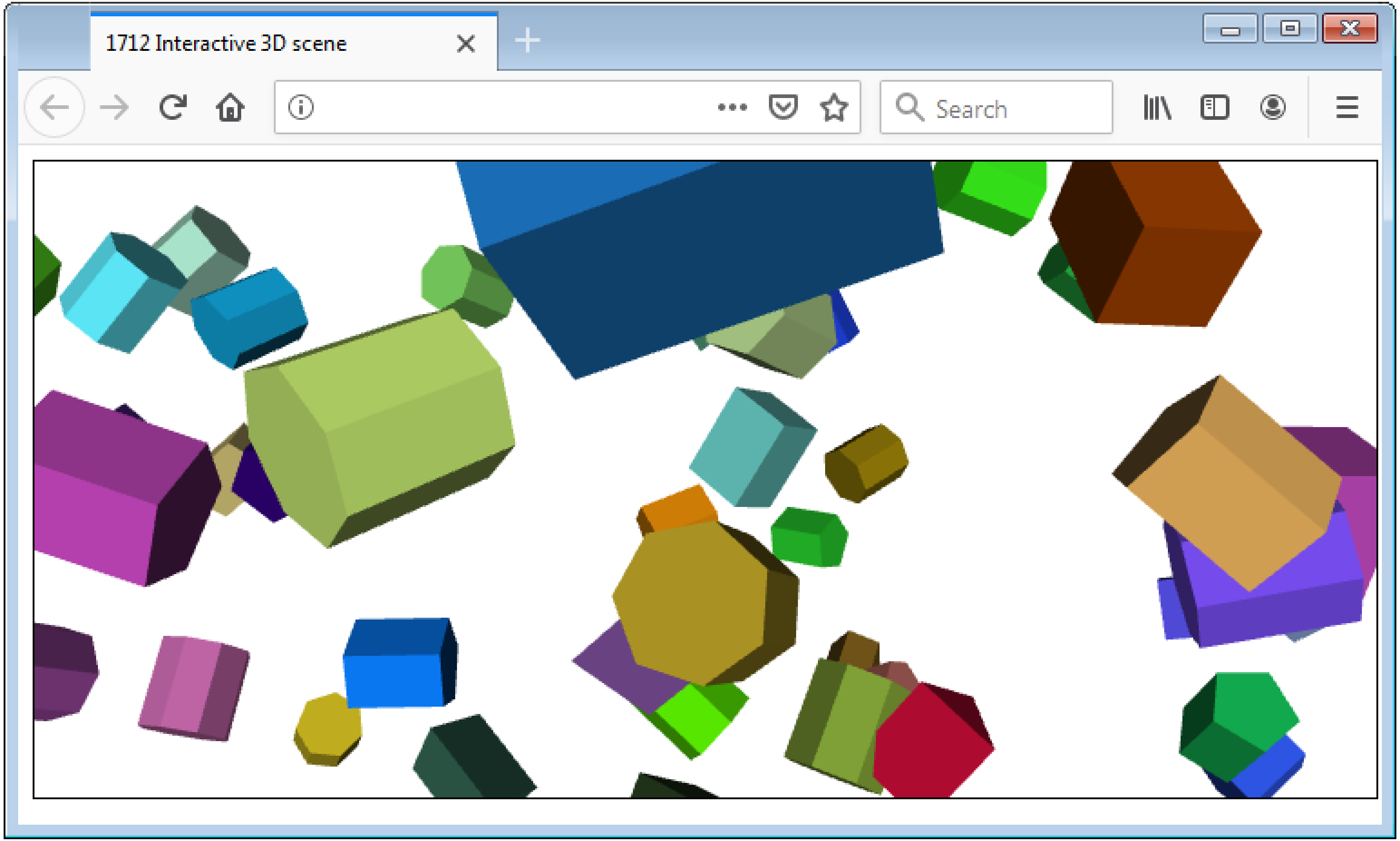
- Distance is raised on power (why?)
- Added restriction on distance

```
lookD *= Math.pow(1.01,event.clientY-y);
if (lookD<10) lookD=10;
if (lookD>1000) lookD=1000;
```

- Angles for the spherical coordinates are bound to the horizontal and vertical mouse motion
- Value 200 means motion of 200 pixels corresponds to rotation of 1 radian
- Vertical angle is restricted to avoid looking from the top or the bottom (there the up vector is seen as zero vector)

```
lookA -= (event.clientX-x)/200;

lookB += (event.clientY-y)/200;
if (lookB>+1.5) lookB=+1.5;
if (lookB<-1.5) lookB=-1.5;
```



TRY IT

Summary

Following and selecting an object

Following the mouse

- Hard link – fixed distance
- Soft link – elastic distance

Selecting an object

- With calculating its position
- With the function objectAtPoint
- Only interactive objects are processed by objectAtPoint

Drag and drop

Dragging with the mouse

- Step 1 – grabbing (selecting) an object
- Step 2 – moving (following)
- Step 3 - dropping the object

Scene dragging

- With dragging of the view point
- Possibility for interactive rotation of the scene
- Possibility for interactive navigation in the scene

ICT in SES

The end

Comments, questions