View point

Lesson №16

View point

View point

View point

- Every scene includes a view point
- Always existing

Using

- Defining how a scene is shown
- Moving inside a 3D world
- Sliding the view (e.g. zoomin in or out)

View point structure

It is not just a 3D point

- Defines user's viewing location
- Defines direction of looking
- Defines view orientation

Mathematical background

- The view point is a projective matrix
- Its determinant must be non-zero


EUROPE

View point in Suica

Function lookAt

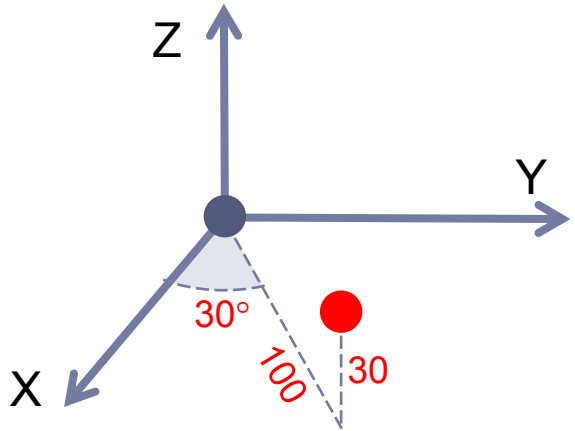
- Defines the view point
- Function lookAt (*position*, *target*, *up*)

Also

- Function demo uses lookAt
- Projection with perspective or orthographic is independent on the view point

The position is a 3D point

- Defines the location of the eye
- By default it is [86.6, 50, 30] or 100[cos 30°,sin 30°,0.3]

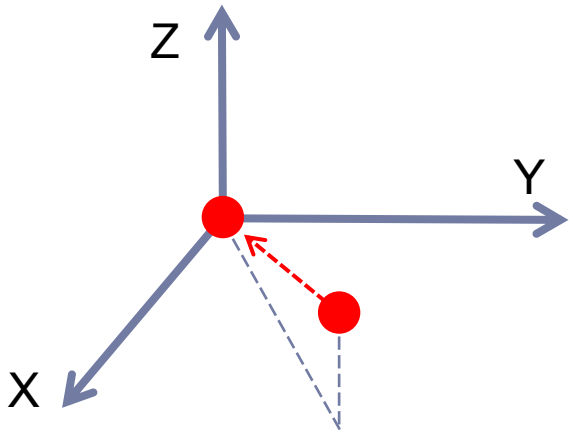


Requirements

- Sufficiently far from the scene
- But not too far from it

The target is a 3D point

- Defines the point being looked at
- It is positioned in the center of the canvas
- By default it is [0,0,0]

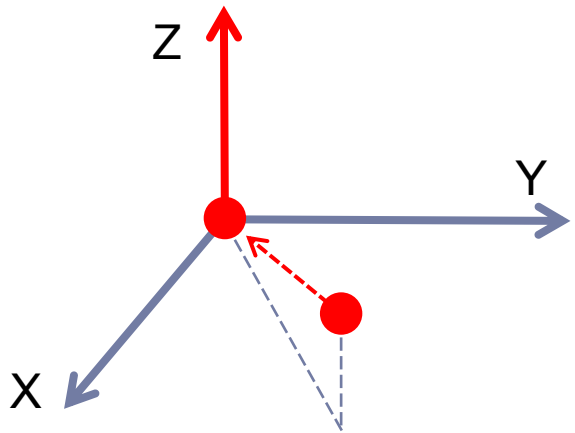


Requirements

- To be distinct from the position

The up is a vector

- Define the orientation (rotation) of the scene
- Points the “up” direction
- By default it is [0,0,1]

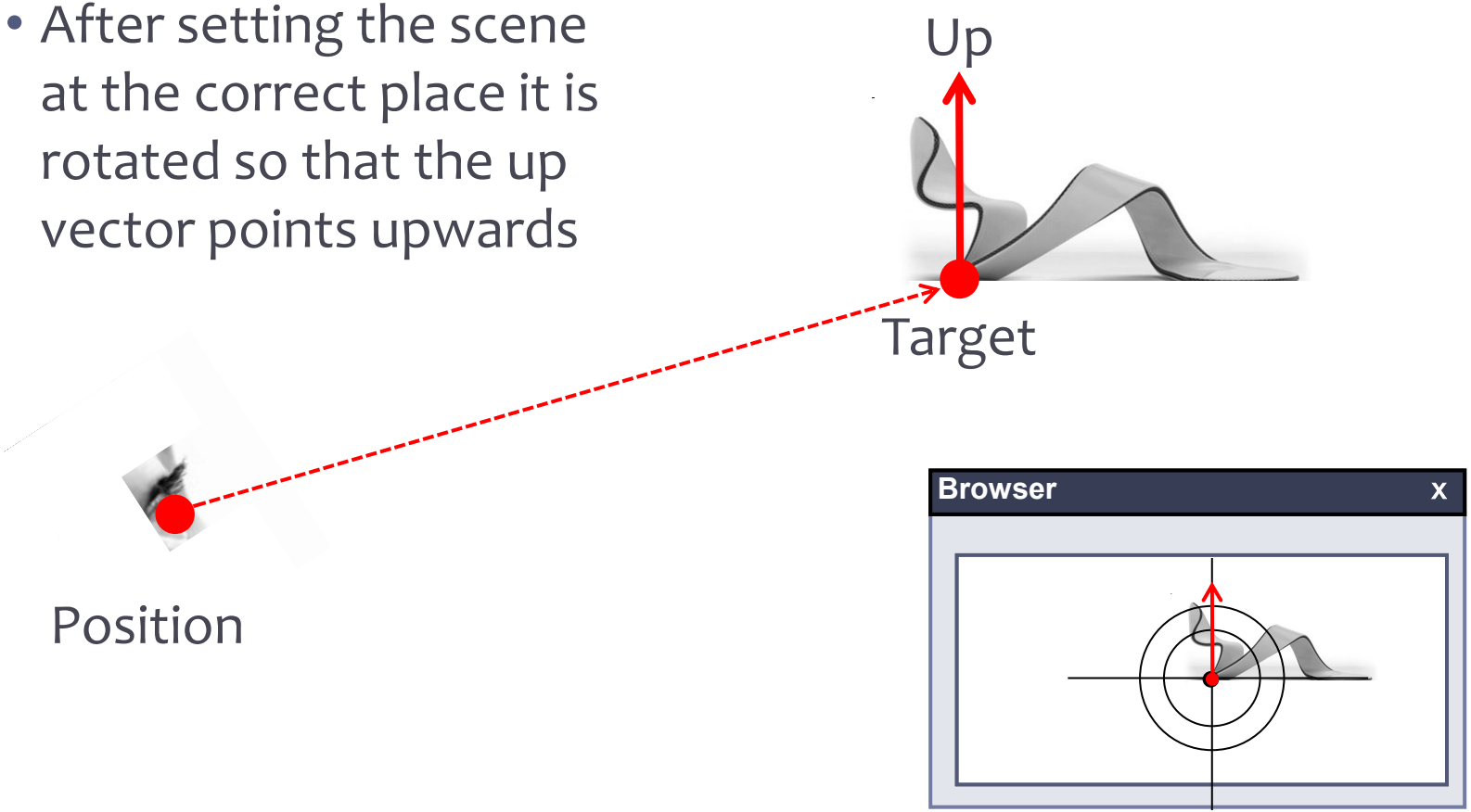


Requirements

- To be seen (from the position) as a non-zero vector

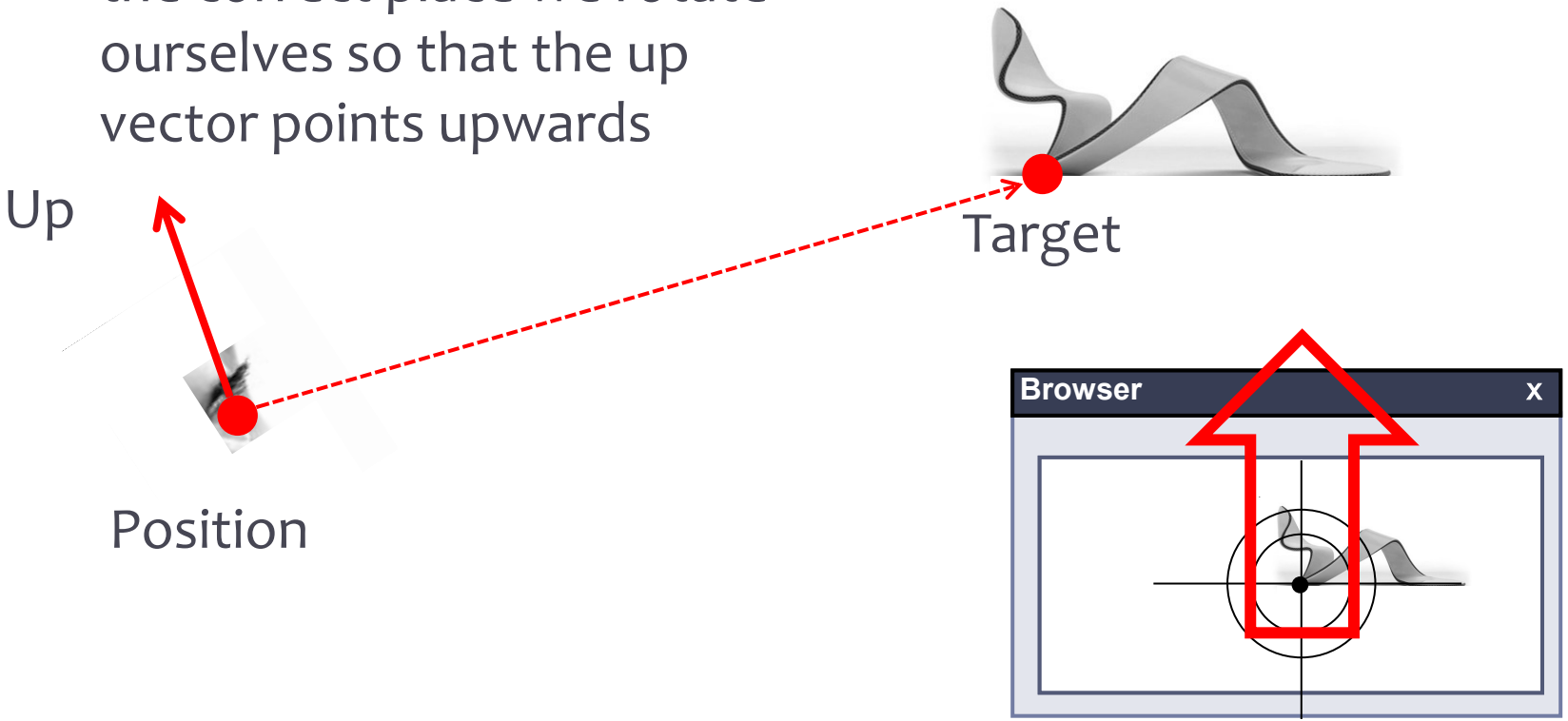
Interpretation №1

- After setting the scene at the correct place it is rotated so that the up vector points upwards



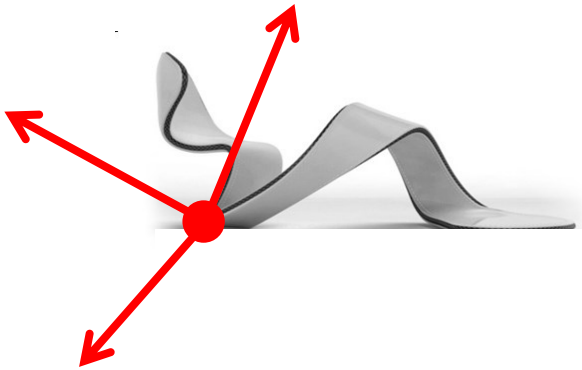
Interpretation №2

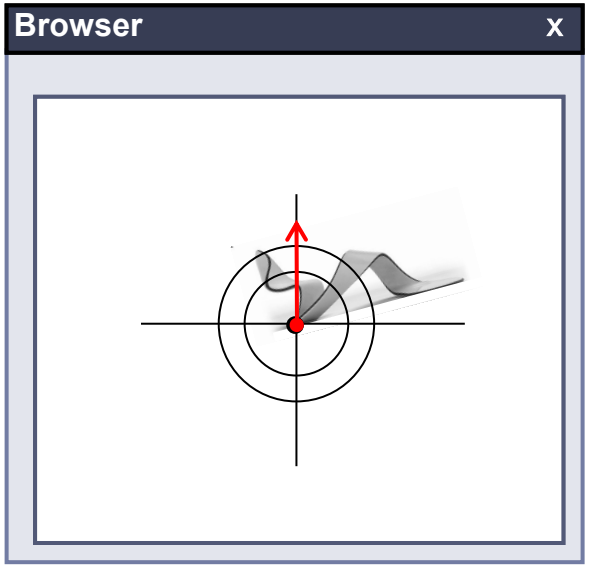
- After setting the scene at the correct place we rotate ourselves so that the up vector points upwards

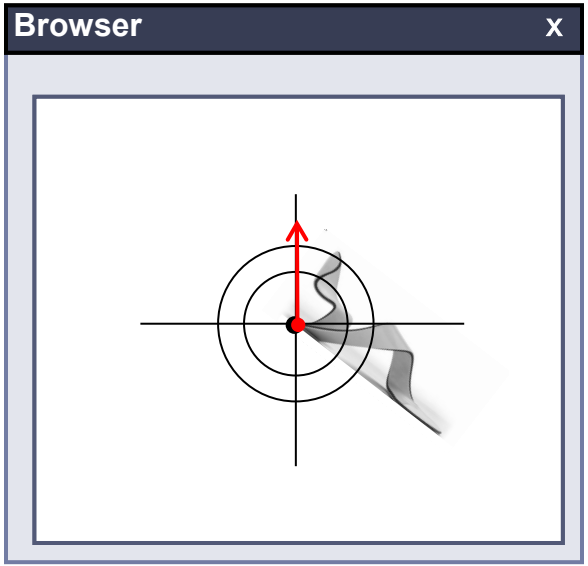


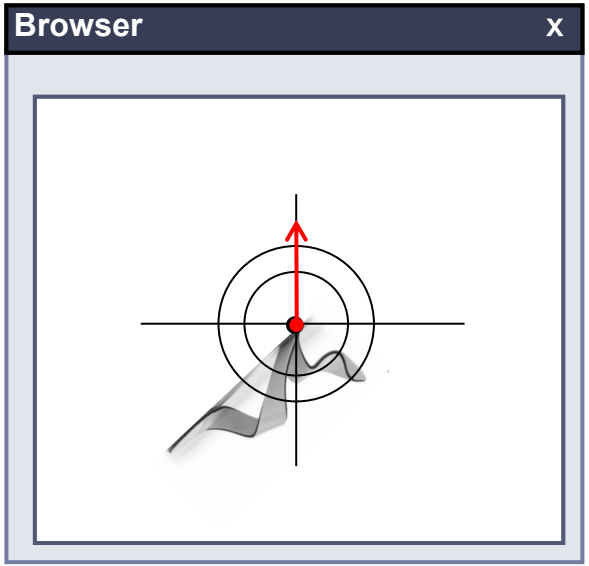
Role of up vector

- The same position
- The same target
- Different up vectors

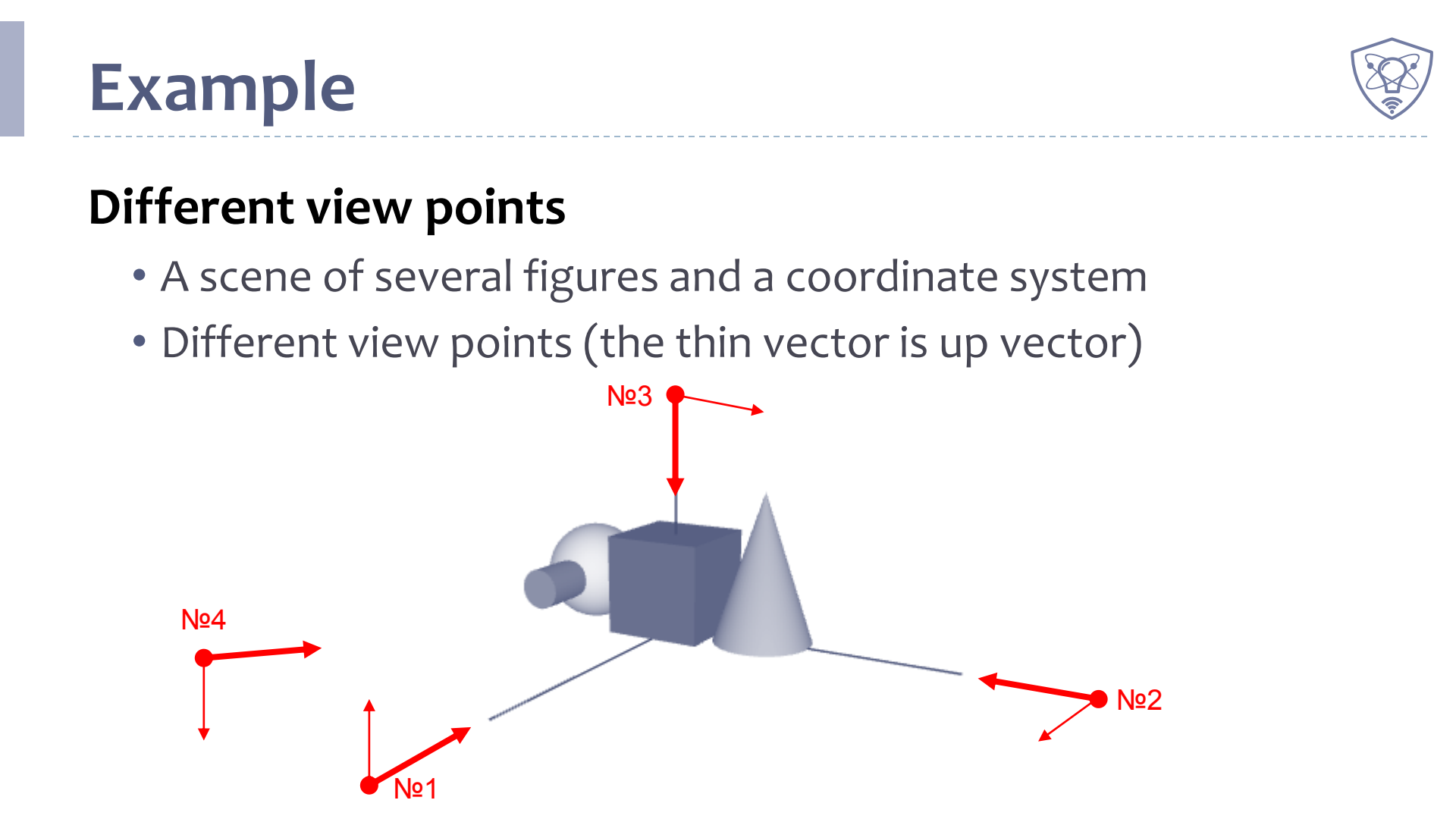








Example



Different view points

- A scene of several figures and a coordinate system
- Different view points (the thin vector is up vector)

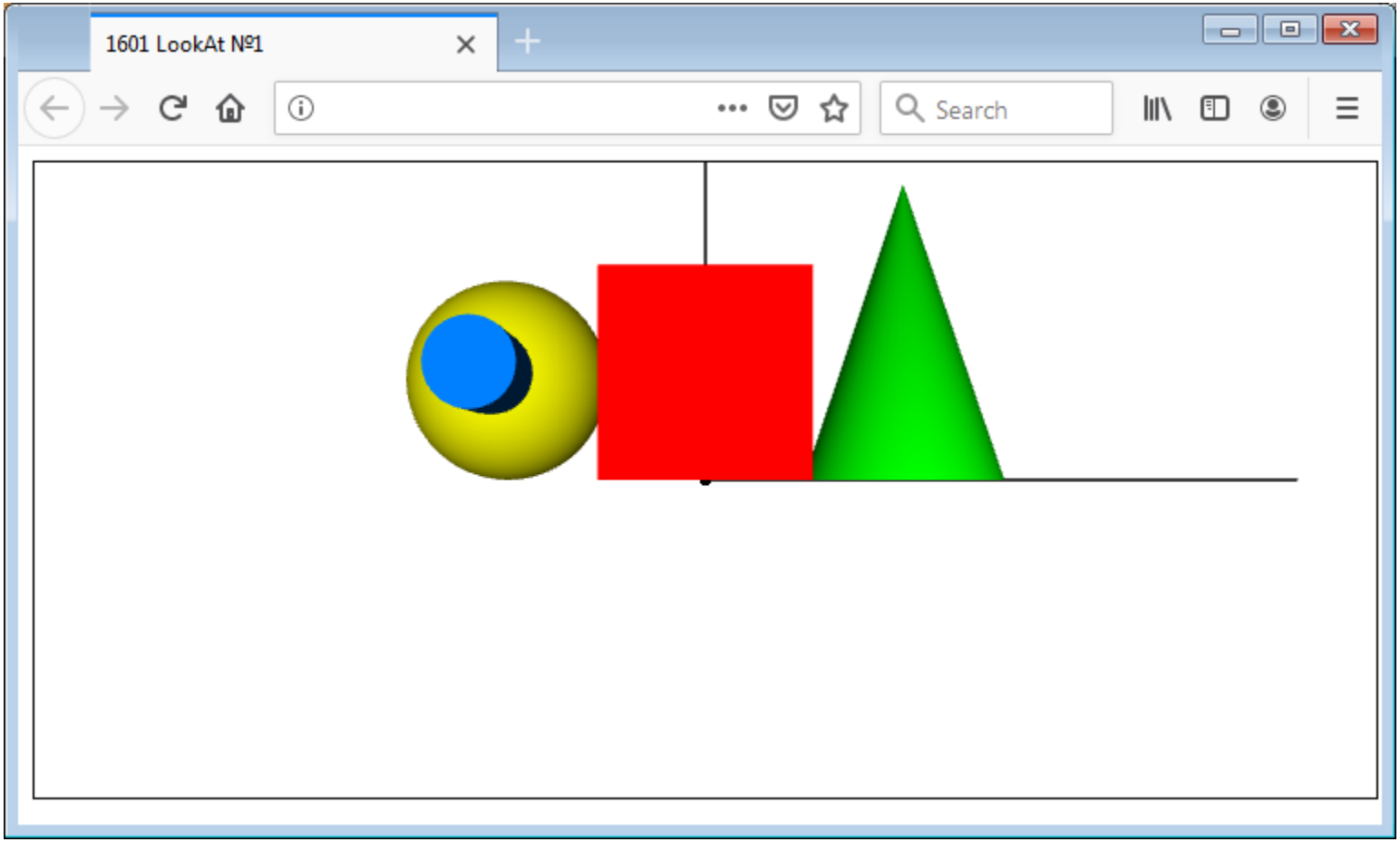
Implementation

- The origins of the thick vectors define positions
- The target in all four cases is (0,0,0)
- Thin vectors define the up direction

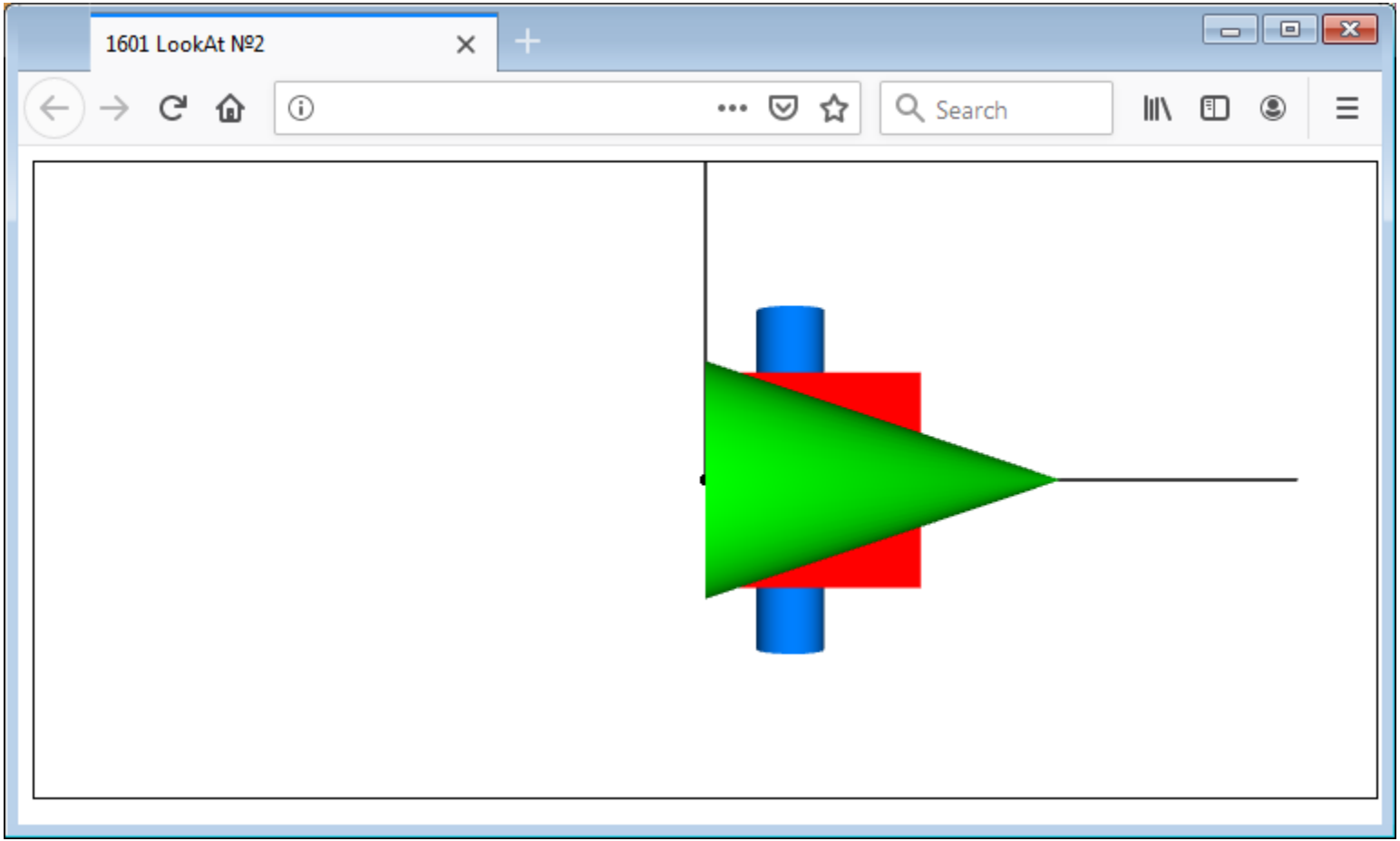
Requirements

- Positions and targets are distinct points
- Up vectors are not and are not seen as zero-vectors

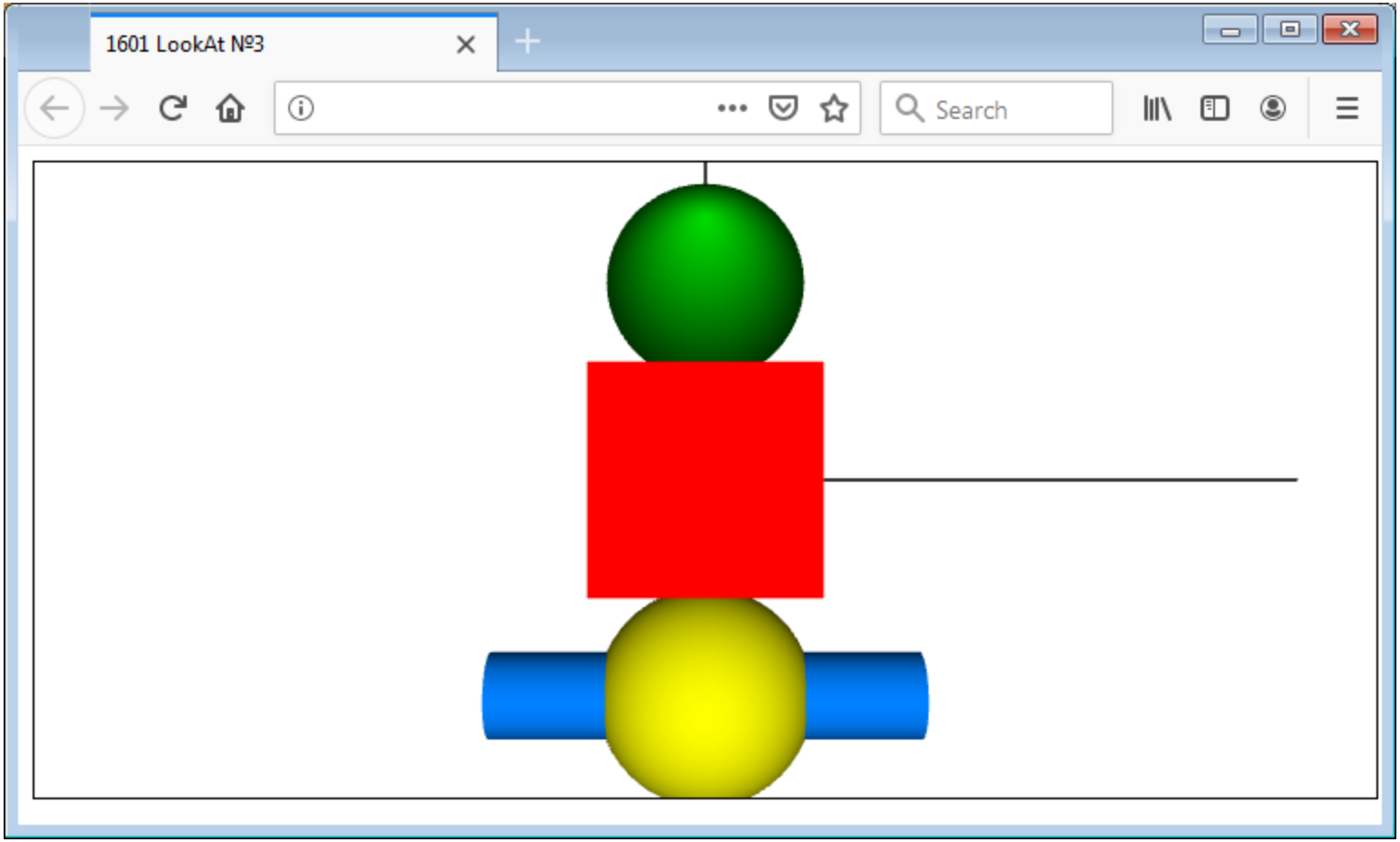
```
lookAt( [60,0,0], [0,0,0], [0,0,1] );
lookAt( [0,60,0], [0,0,0], [1,0,0] );
lookAt( [0,0,60], [0,0,0], [0,1,0] );
lookAt( [50,50,0], [0,0,0], [0,0,-1] );
```



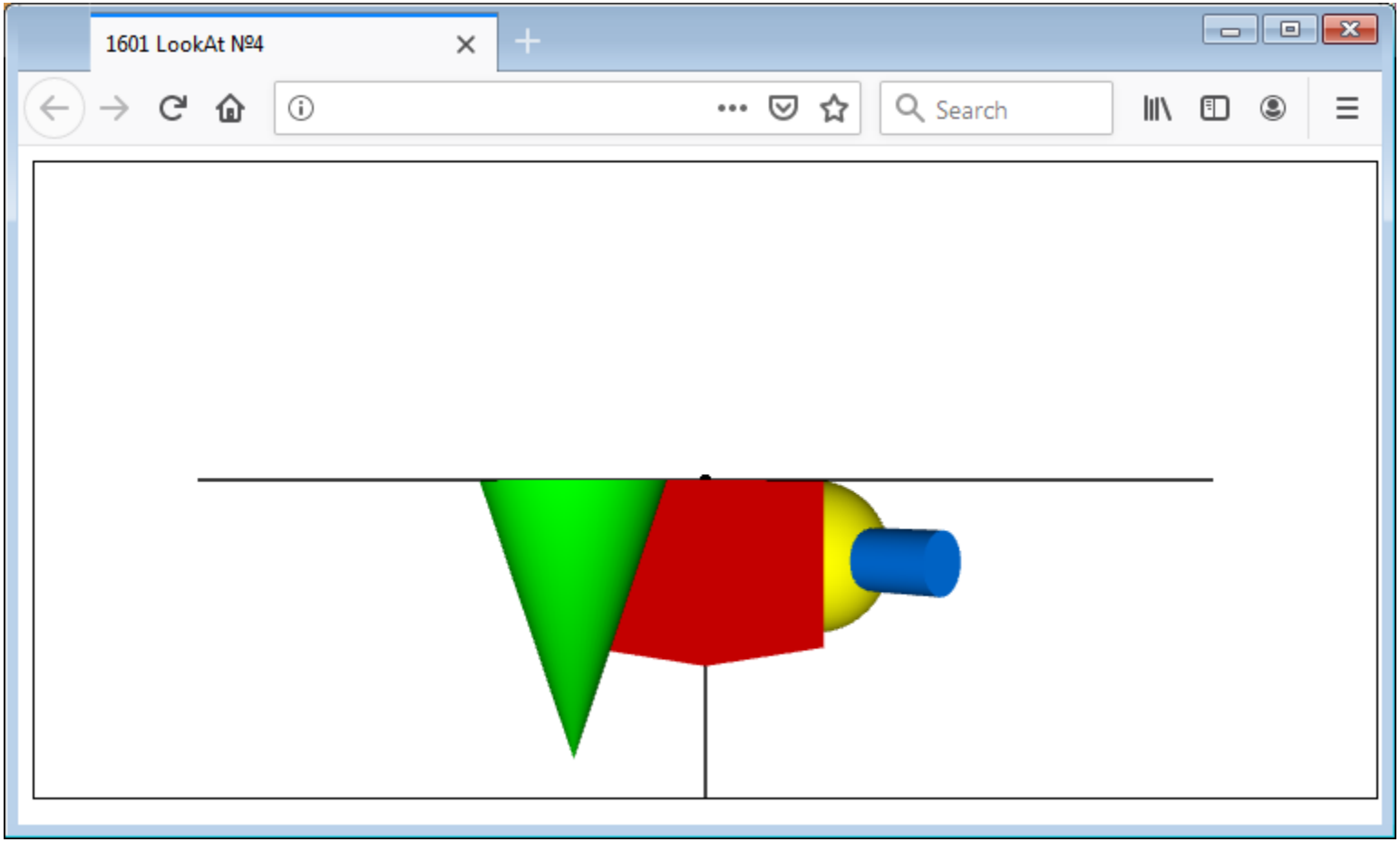
TRY IT



TRY IT



TRY IT



TRY IT

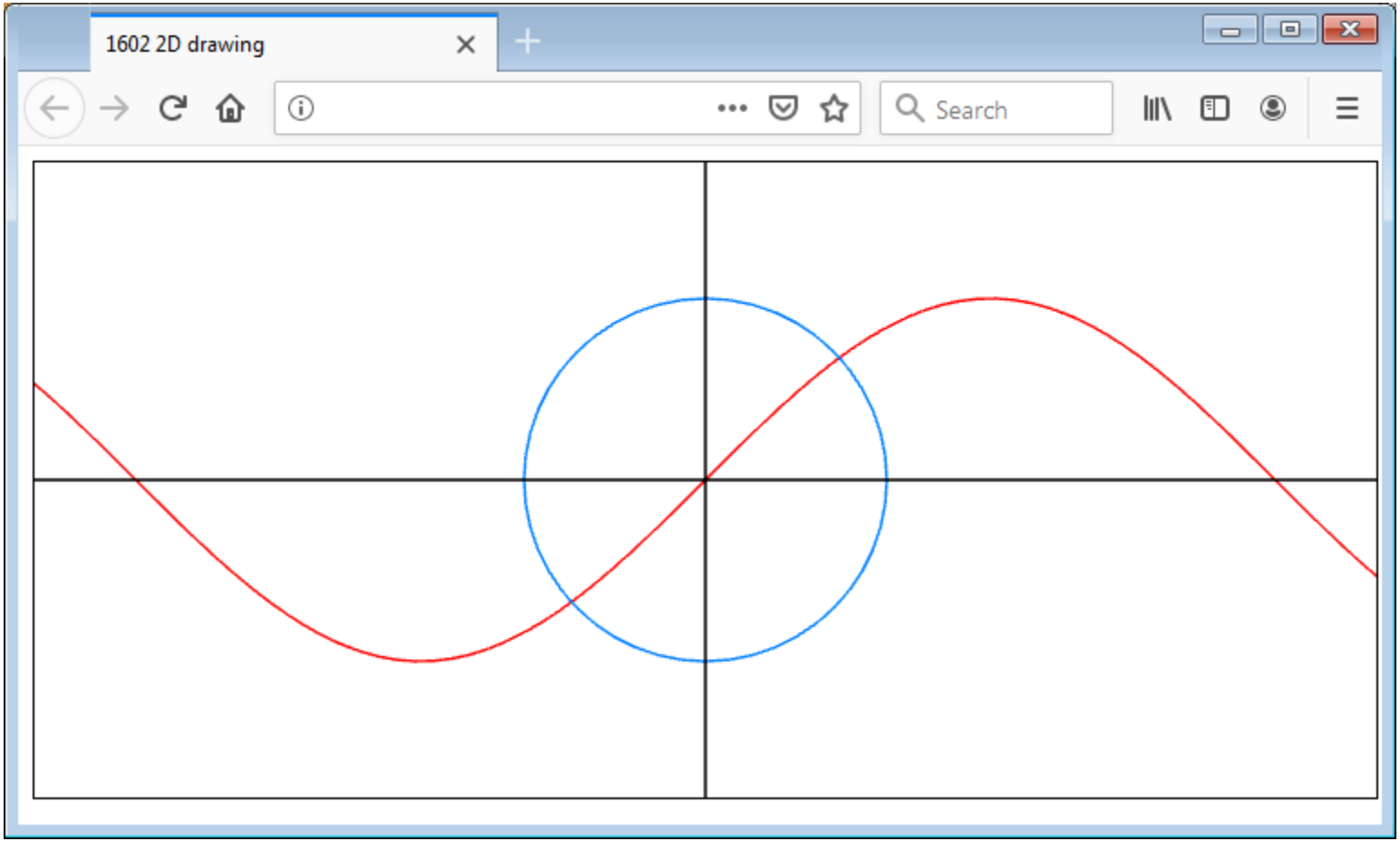
Traditional 2D drawing

Sine curve as in textbooks

- The center is (0,0), X points to the right, Y points upwards
- View position along Z, target is (0,0) and Y is up vector

```
orthographic(-100,100);
lookAt( [0,0,10], [0,0,0], [0,1,0] );

o = [0,0,0];
line(o,[1,0,0]).custom({color:[0,0,0]});
line(o,[0,1,0]).custom({color:[0,0,0]});
...
```



TRY IT

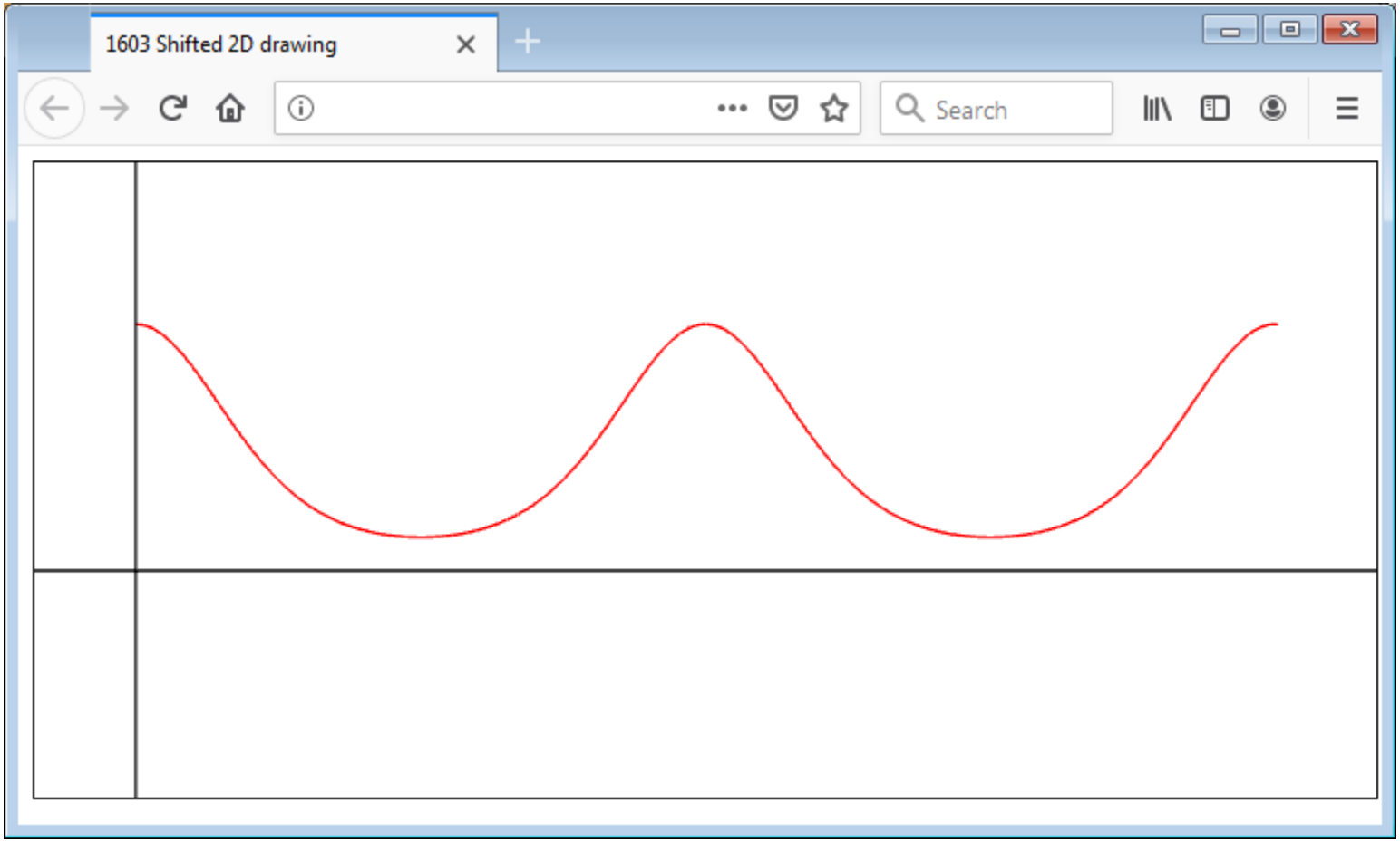
Plot $f(x)=e^{\cos(x)}$ for $x \in [0,2\pi]$

- Position and target are shifted together, thus the viewing direction is orthogonal to the XY plane

```
lookAt([100*Math.PI,50,10],[100*Math.PI,50,0],[0,1,0]);

function f(x) { return Math.exp(Math.cos(x)); }

dX = 0.1;
for (var x=0; x<4*Math.PI; x+=dX)
{
   p = [50*x,50*f(x),0];
   q = [50*(x+dX),50*f(x+dX),0];
   segment(p,q).custom({color:[1,0,0]});
}
```



TRY IT

View point animation

View point motion

View point as graphical element

- Can change
- Can create illusion of motion
  - Movement of the scene
  - Movement of the viewer within the scene

Example

- Change of target is perceived as self rotation

Following a 2D drawing

A plot of $sin(x)$

- X axis has grid marks
- View point follows the point of drawing

Problem

- Creating many points slows down the animation

Solution

- Working with only n points
- Instead of creating new points – reusing old off-screen points

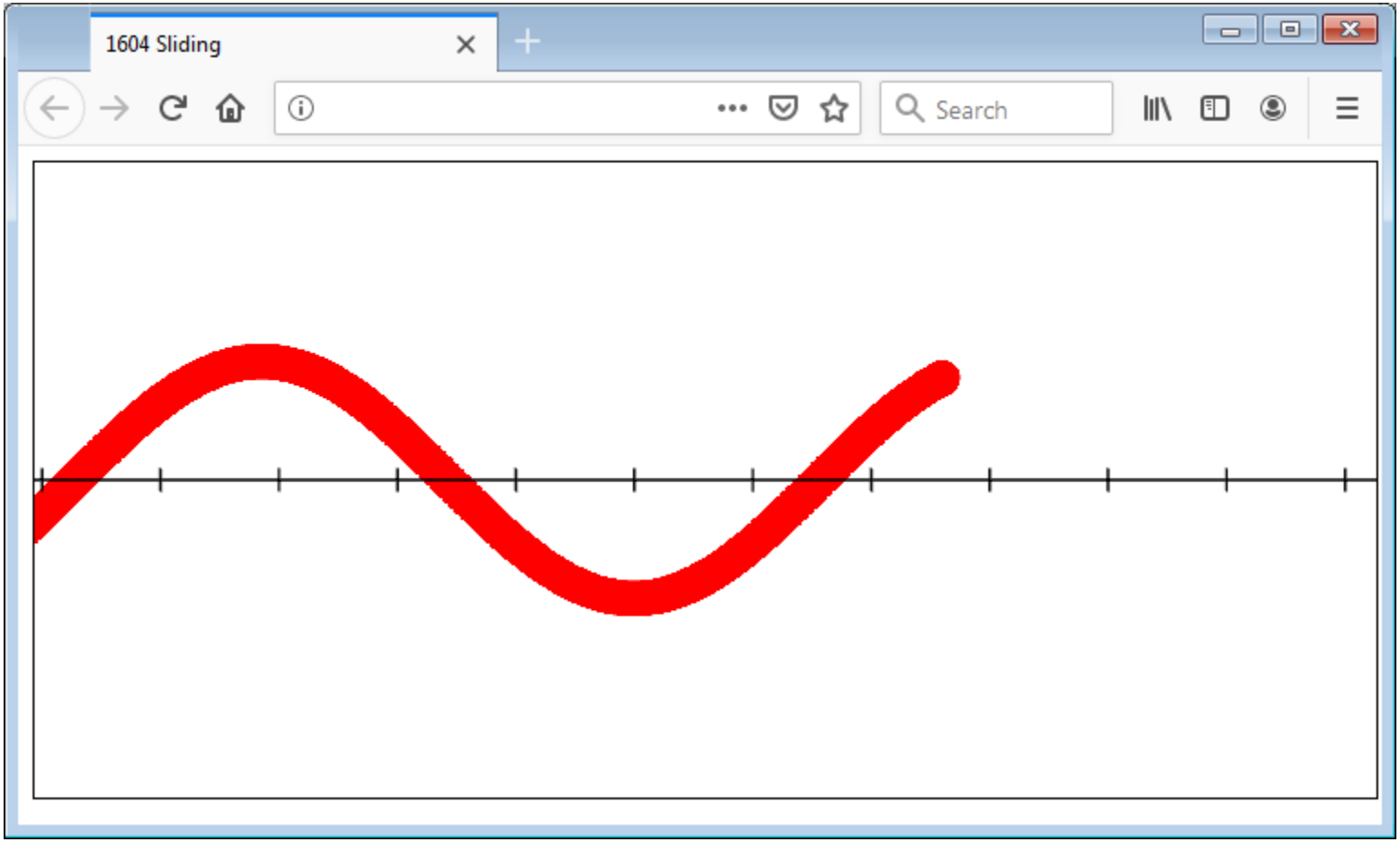
Implementation

- An array of n points

```
n = 160;
q = [];
for (var i=0; i<n; i++) q[i] = point([0,0,0])...
```

- Counter i, defining which point is being generated
- Looking 2 units behind the generated point
- Value i%n defines the point index in the array

```
x = i/20;
lookAt( [x-2,0,10], [x-2,0,0], [0,1,0] );
q[i%n].center = [x,Math.sin(x),0];
i++;
```



TRY IT

Spiral

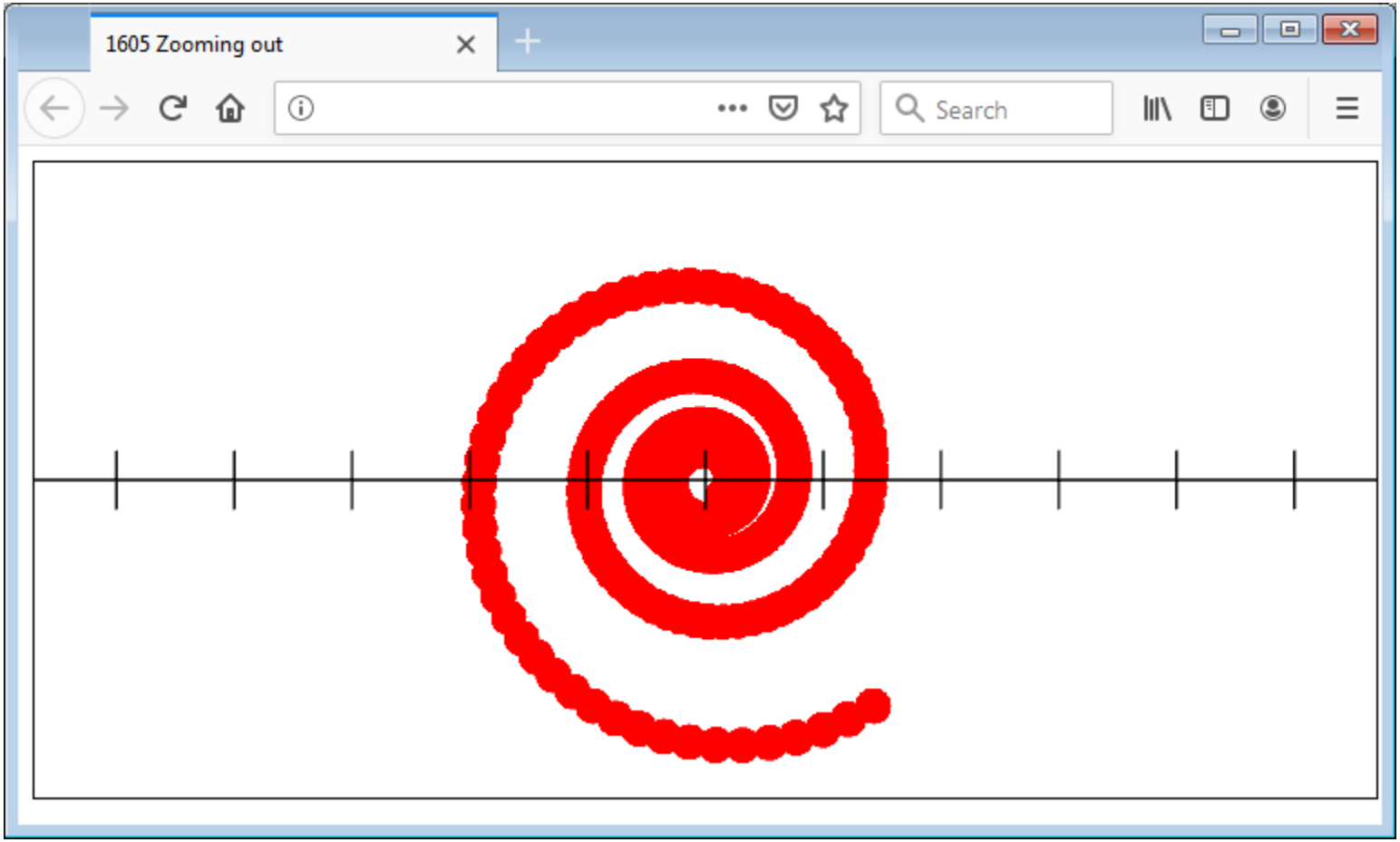
- Circular motion with variable radius r and angle x
- Radius increases gradually
- View point position goes further away (it depends on r)

Question

- Why the plot disappears at some point?

```
r = r*1.01;
lookAt( [0,0,1+4*r], [0,0,0], [0,1,0] );

x = i/10;
q[i%n].center = [r*Math.cos(x),r*Math.sin(x),0];
i++;
```



TRY IT

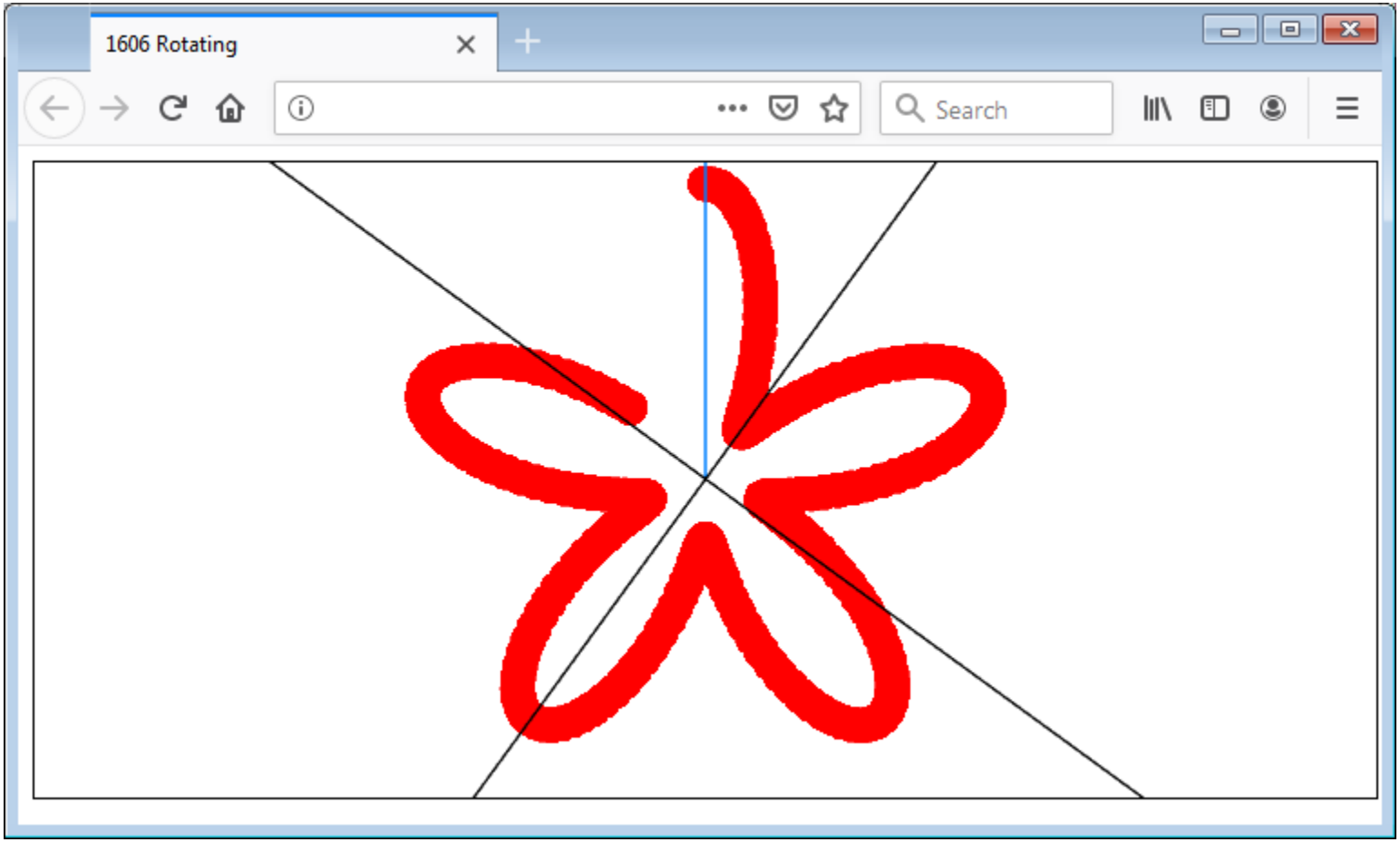
Plot of $r=3+2sin(5x)$

- The scene is rotating by rotating the up vector
- The drawing position is always upwards
- It is marked with the ray l, which second point rotates as the scene rotates

```
x = i/40;
r = 3+2*Math.sin(5*x);
l.to = [Math.cos(x),Math.sin(x),0];

lookAt([0,0,20],[0,0,0],[Math.cos(x),Math.sin(x),0]);

q[i%n].center = [r*Math.cos(x),r*Math.sin(x),0];
i++;
```



TRY IT

Animation in 3D

Scene rotation

Double interpretation

- Different motions with the same visual representation
- They have different performance impact

Examples

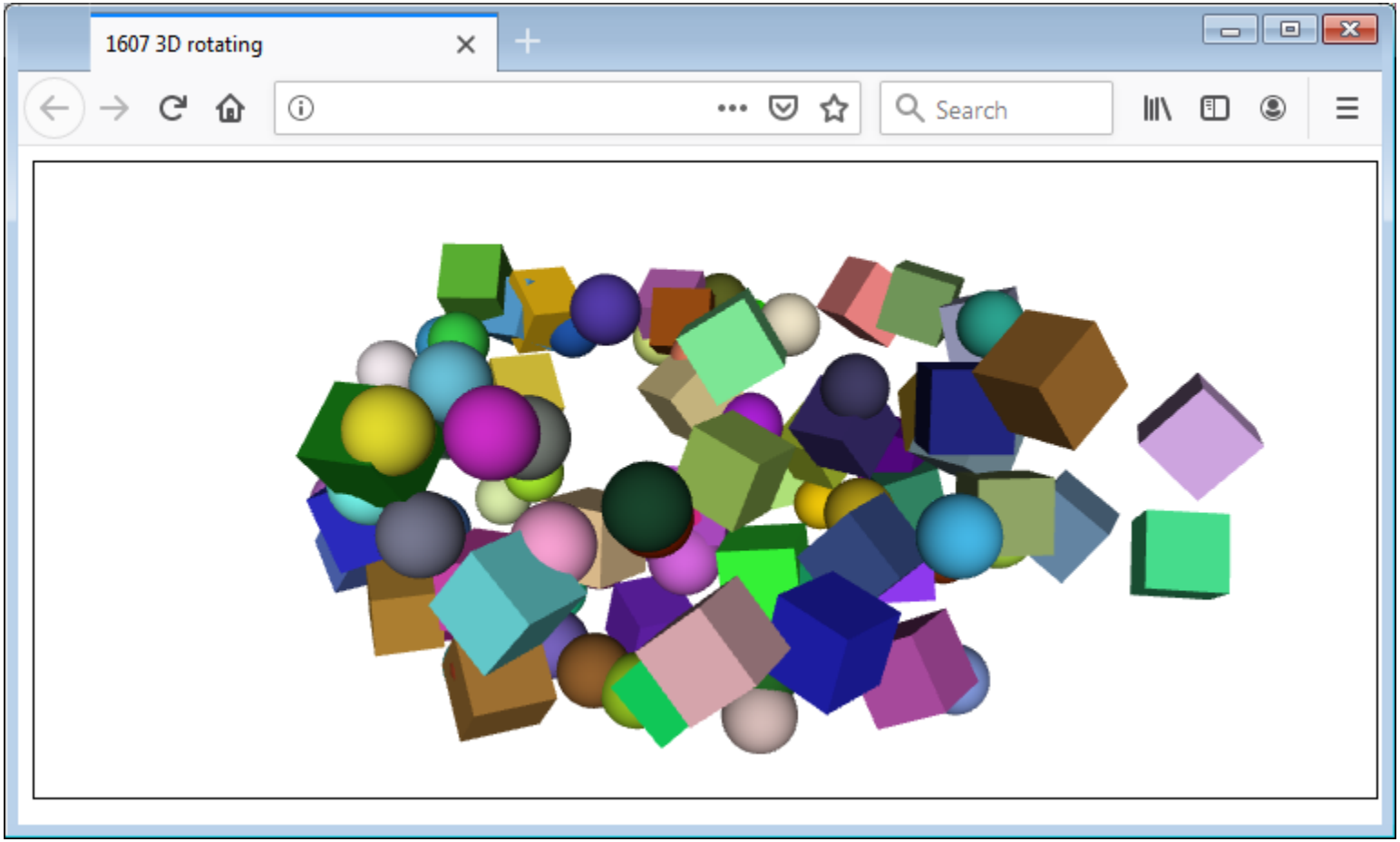
- A scene with objects rotating around the Z axis
- A view point orbiting a fixed scene with objects

Implementation of scene rotation

Scene rotation via orbiting

- Simulation of the function demo
- Scene contains immobile objects, positioned randomly
- View point position make a circular motion, target is fixed

```
function rotateViewpoint()
{
   t = Suica.time;
   lookAt( [100*Math.cos(t),100*Math.sin(t),30],
                                    [0,0,0], [0,0,1] );
}
```

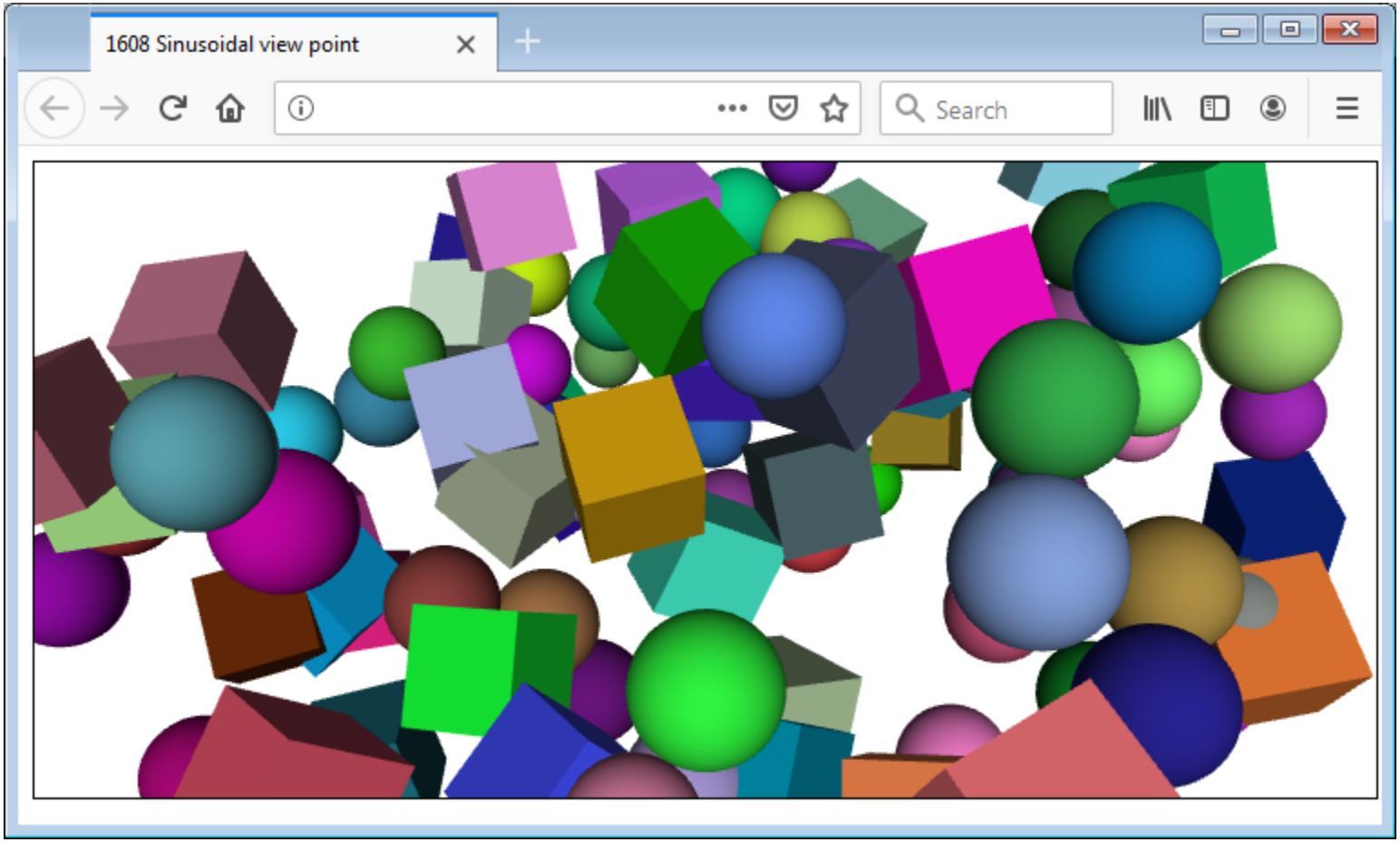


TRY IT

Sine rotation

- View point orbiting the scene
- Going near and further while orbiting
- Slightly waggling by changing the up vector
  (direction is variable, but is predominantly vertical)

```
function rotateViewpoint()
{
   t = Suica.time;
   d = 100+40*sin(2*t);
   lookAt( [d*cos(t),d*sin(t),30],
           [0,0,0],
           [sin(2*t),cos(3*t),2] );
}
```



TRY IT

Examples

Chasing an object

Scene

- Square matrix of building
- Object moving randomly in the matrix
- View point chasing the object

Idea

- The object is the target of the view point
- Position and targets will be modified by a linear combination (this produces smoother motion)

A matrix of buildings

- Cuboids at odd coordinates
- Random heights
- Random bluish colours

```
for (var x=-12; x<11; x+=2)
for (var y=-12; y<11; y+=2)
{
   var c = random(0.5,1);
   cuboid([x+1,y+1,0],[0.8,0.8,random(0.4,2)]).custom({
      origin: [0,0,-0.5],
      color:  [0,c/2,c]
   });
}
```

Object motion

- The object ball is a sphere
- Its direction of motion dir is an angle
- Parameter k defines the number of steps to travel from one intersection to another intersection (distance 2 units)

```
dir=0;
k=20;
function chase()
{
   ball.center[0] += 2/k*Math.cos(dir);
   ball.center[1] += 2/k*Math.sin(dir);
}
```

Turning left or right

- Turning is done on every k steps (number of frames is store in frame)
- Turning adds $-\pi/2$, 0 or $\pi/2$ to the direction angle – this makes turning left, going forwards or turning right

```
if (frame%k==0)
{
   dir += Math.PI/2*(Math.round(random(-1.5,1.5)));
   ball.center[0] = Math.round(ball.center[0]);
   ball.center[1] = Math.round(ball.center[1]);
}
```

Motion restriction

- The object does not leave the matrix
- Building are position within coordinates ±10
- When at the border, clearing frame forcing a new direction

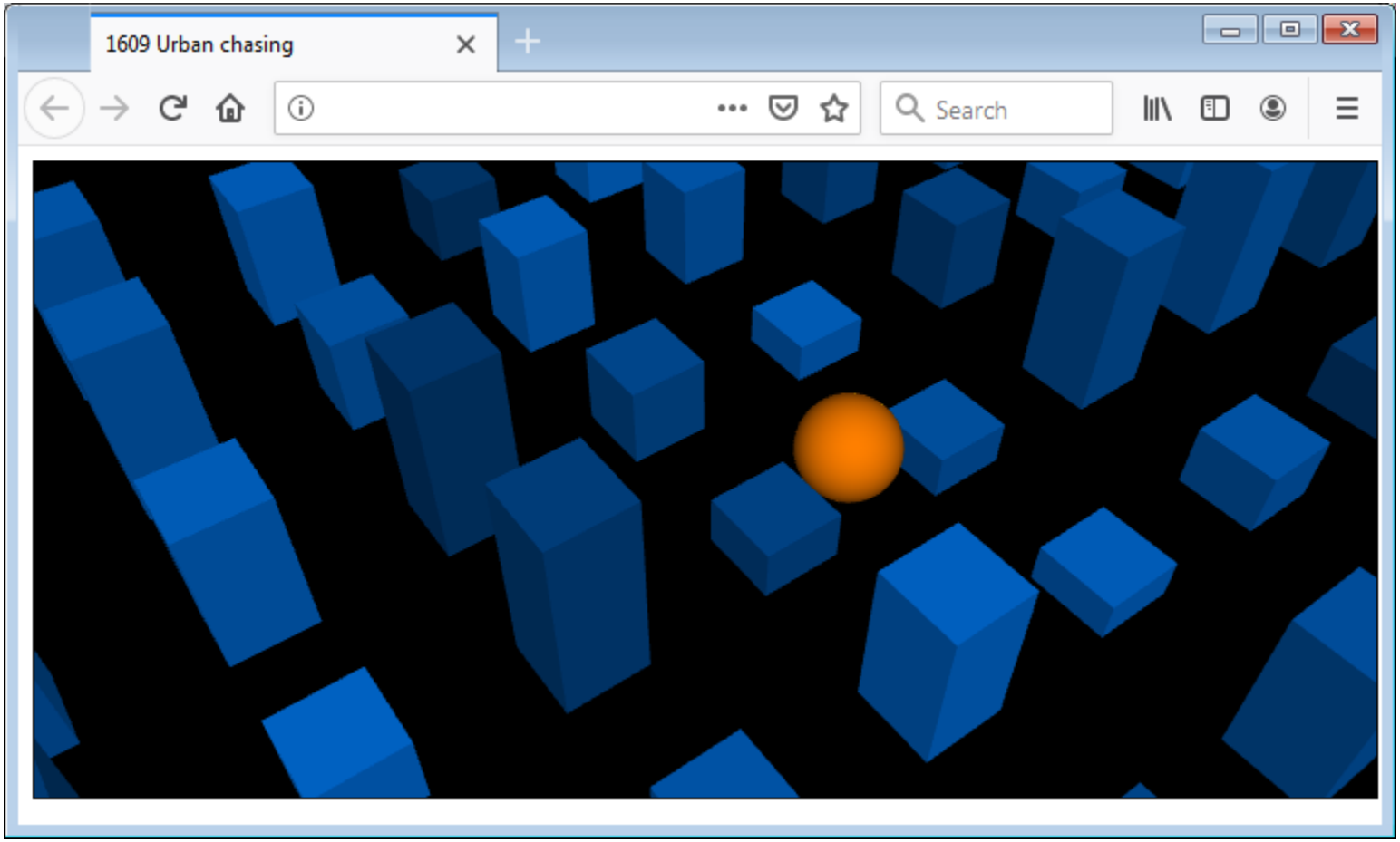
```
if (Math.abs(ball.center[0])>10 ||
    Math.abs(ball.center[1])>10 )
{
   ball.center[0] = Math.max(ball.center[0],-10);
   ball.center[0] = Math.min(ball.center[0],+10);
   ball.center[1] = Math.max(ball.center[1],-10);
   ball.center[1] = Math.min(ball.center[1],+10);
   frame = 0;
}
```

Motion of view point

- Position and target change smoothly towards the moving object
- There is a distance between them
- Coefficients in the linear combinations define how smooth is the motion of the view point

```
for (var i=0; i<3; i++)
{
   target[i] = target[i]*0.96+0.04*ball.center[i];
   pos[i] = pos[i]*0.98+0.02*ball.center[i]+0.3/(4-i);
}

lookAt(pos,target,up);
```



TRY IT

Motion in scene

Scene

- The same matrix of building
- The view point is “in” the moving object

Implementation

- Position is the same as the position of the object (actually there is no need to have an object)
- Direction of motion and position define the target

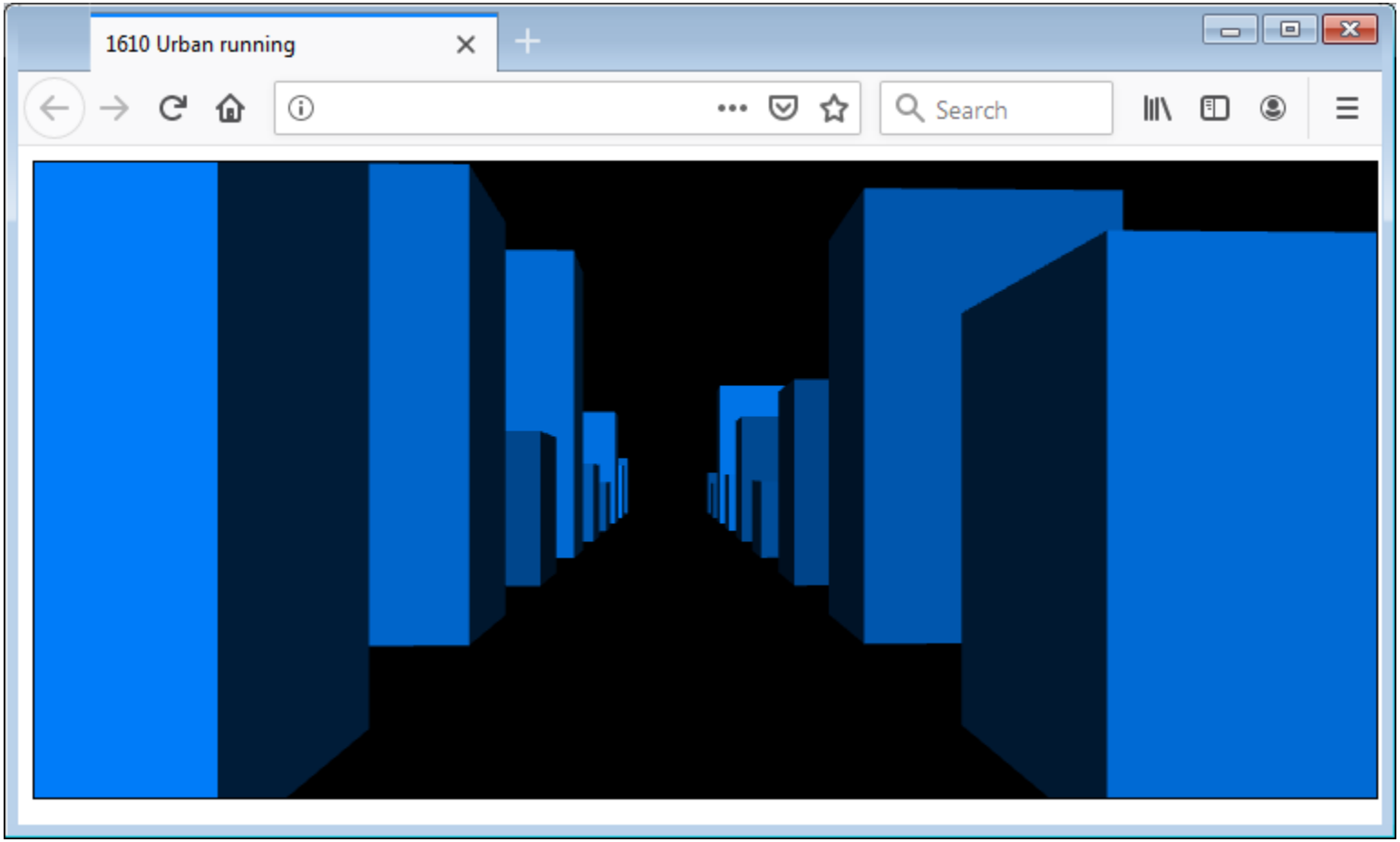
Smooth motion

- A second position pos2 and direction dir2, which follow pos and dir with a linear combination
- Target target2 is a point from a circular motion at angle dir2 around center pos2

```
for (var i=0; i<3; i++)
   pos2[i] = pos2[i]*0.95+0.05*pos[i];

dir2 = dir2*0.95+0.05*dir;
target2 = [ pos2[0]+Math.cos(dir2),
            pos2[1]+Math.sin(dir2),0.5];

lookAt(pos2,target2,up);
```



TRY IT

Summary

View point

View point

- Defined by lookAt

Parameters

- Position – 3D point of the eye
- Target – 3D point to look at, projected in the canvas center
- Up vector – 3D vector rotating the scene, it is always upwards

View point motion

Two effects

- Illusion of moving the whole scene and all its objects
- Implementation of navigation within a scene

Smoothness

- Motion of view point is sensitive to sharp changes, smoothing motion is recommended

The end

Comments, questions