Animation

Lesson №13

Principle of animation

Animation

Etymology

- From Latin *anima – giving life*
- Derivative words: animator, animalist, anime, reanimation

Two-way “cheating”

- The animator “cheats on” the spectator
- The spectator “cheats on” the animator

Seeing

Biological process

- People look with the eyes, but see with the brain
- Eyes and neurons have limited capacity

Image in the brain

- Persists for around 1/15 seconds
- Less than 15 images/sec – separate images
- More than 15 images/sec – continuous motion

Implementation

High level

- Creating frames
- Showing them one by one

Low level

- Changing properties

  motion – change of center

  rotation – change of orientation

  expanding – change of size

  turning pale – change of colour

Algorithm

Idea for generating animation

- Step 1: generate a frame
- Step 2: deliver a frame
- Step 3: go to step 1

Type of deliveries

- In real time (on screen, streaming)
- Not in real time (video file)

Pseudocode

- Traditional animation loop

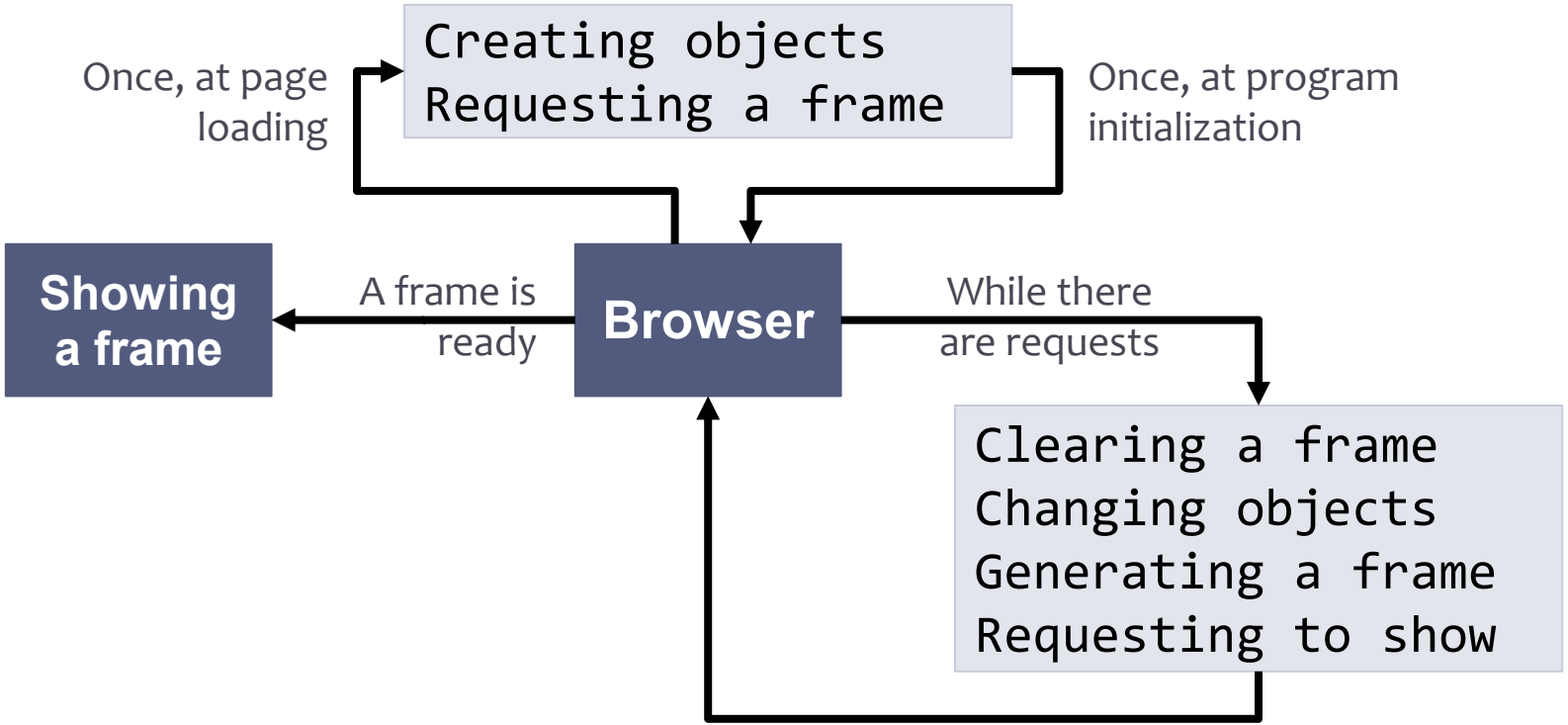
```
creating objects
loop for each frame
{
  clearing frame
  changing objects
  showing frame
}
```

Preference

- Object creation is outside the animation loop

Browser animation

- The program requests to show a frame
- The browser decides when will this happen



Animation loop in Suica

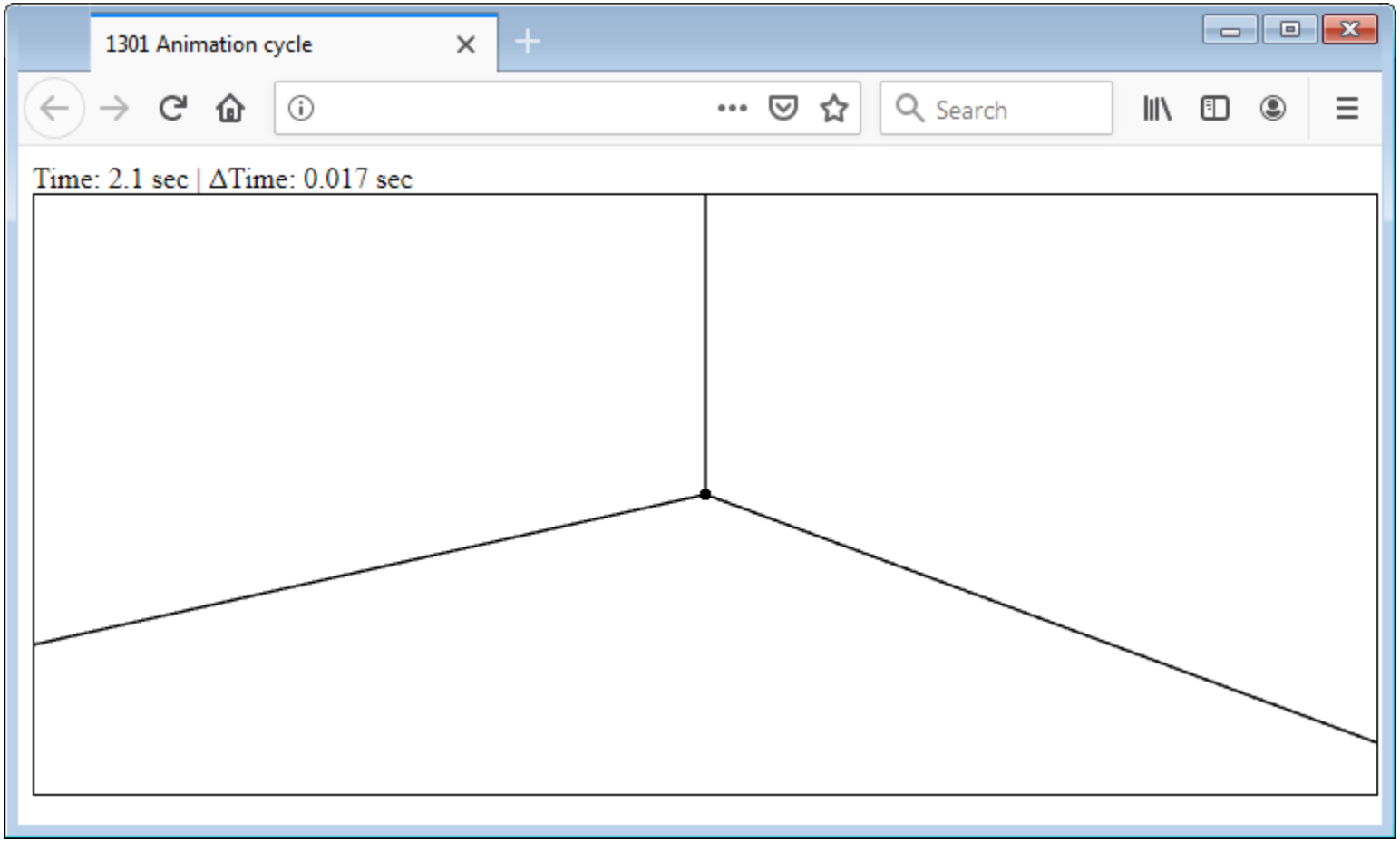
- Remember the Suica instance in a variable
- The instance has property nextFrame
- It contains the function for animation
- It is activated automatically at browser's decision

```
function main()
{
   p = new Suica();
   p.nextFrame = loop;
}

function loop()
{...}
```

Loop example

- Showing the elapsed time since the start of Suica – recorder in the variable Suica.time
- Showing the elapsed time since the previous frame Suica.dTime
- Using toFixed to round fractional numbers

```
function loop()
{
   document.getElementById('t').innerHTML = 'Time: '+
      Suica.time.toFixed(1)+' sec | ΔTime: '+
      Suica.dTime.toFixed(3)+' sec';
}
```



TRY IT

Linear motion

Linear motion

Different interpretations

- Geometrical – motion along a straight line
- Parametric – motion described with a single parameter
- Physical – motion depending linearly on time

Which one is used

- It depends on the context

For now

- Motion on a straight line with constant speed defined by a single parameter

Implementaiton

- With velocity vector
- With target point
- With equation of trajectory

Velocity vector

Linear motion with a vector

- The vector determines the direction and the speed
- If the vector is fixed, the motion is on a straight line

Mathematical mode

- P – center of object
- v – velocity vector
- Incremental motion: $P \leftarrow P+v$

Animation in real time

- Speed is number of space units per second
- Measuring elapsed time:
  - Animation speed should not depend on computer cpeed
  - If computer is slow the animation makes larger steps
- Browsers try to equalize the speed (usually 30 or 60 fps)
- Equation: $P_{t+\Delta t} = P_t + v.\Delta t$

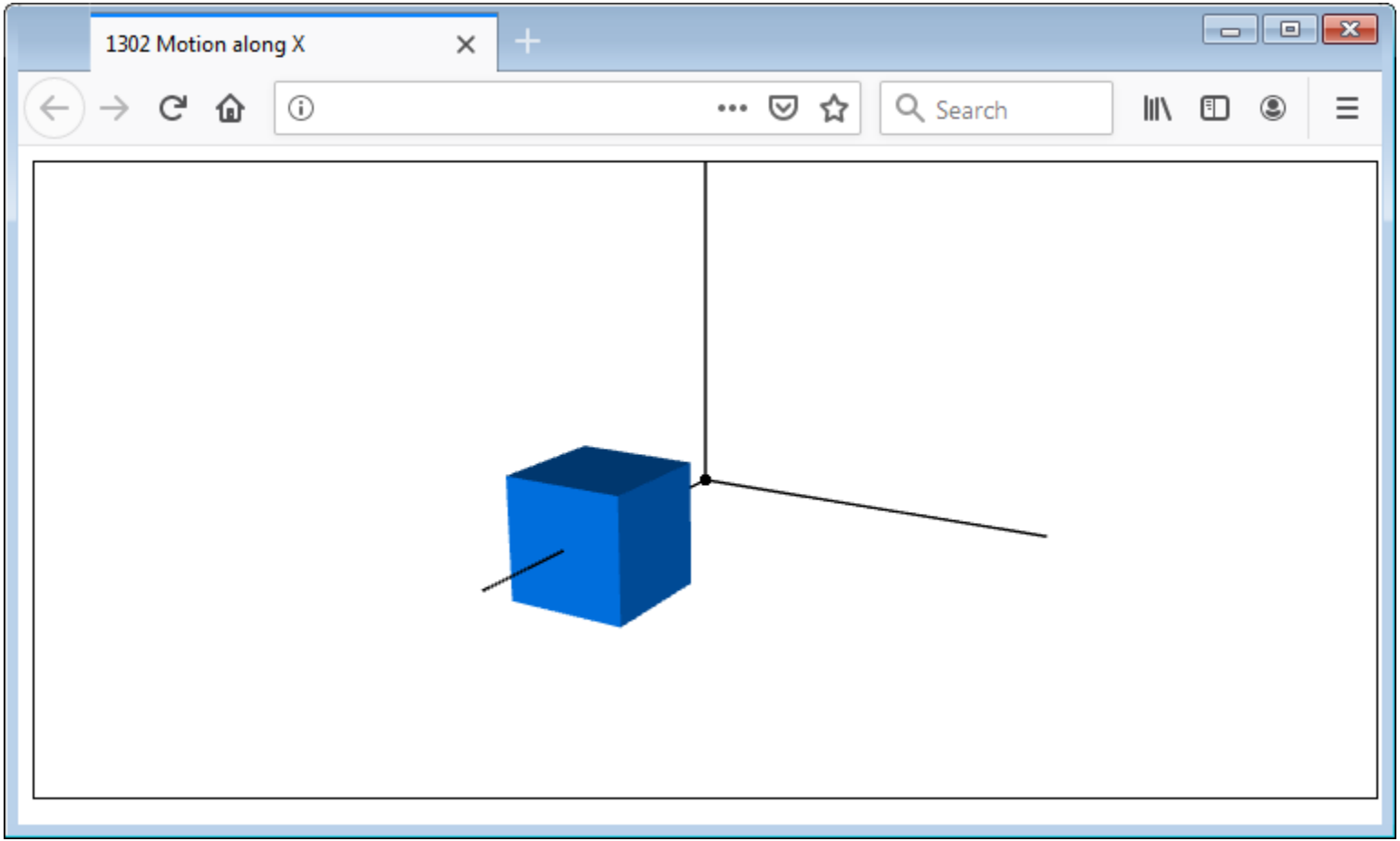
Animation in not real time

- Speed is space units per frame
- Measuring frames
  - Animation speed depends on computer speed
  - If computer is slow the animation is also slow
- Equation: $P_{f+1} = P_f + v$

Motion along X

- Cube is on the X axis and moves towards +X
- Speed is 5 units per second

```
function main()
{
   p = new Suica();
   oxyz();
   a = cube([0,0,0],10);
   p.nextFrame = moveX;
}
function moveX()
{
   a.center[0] += 5*Suica.dTime;
}
```

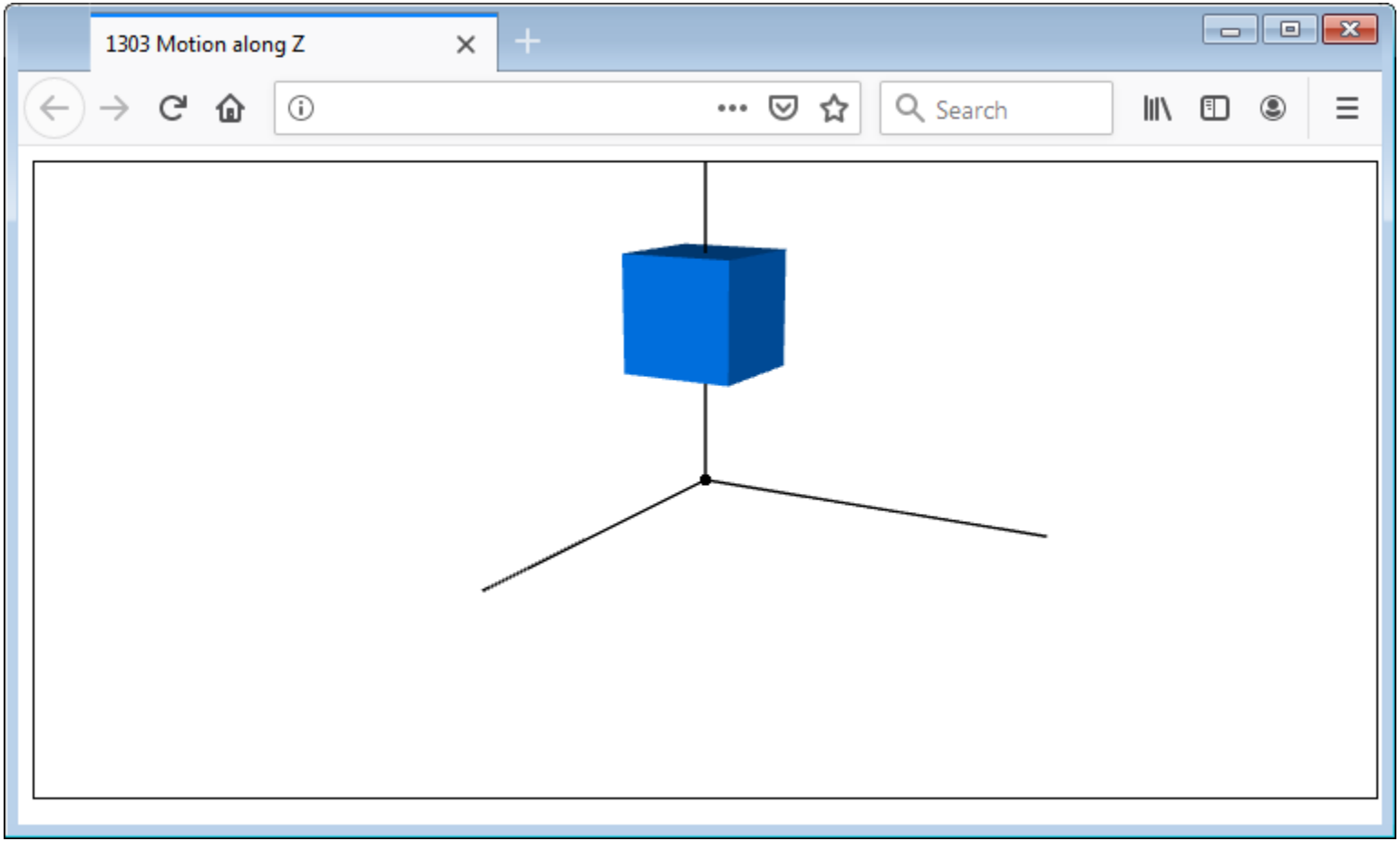


TRY IT

Motion along Z

- Faster motion (15 units per second)
- Reaching a given height restart the motion from (0,0,0)

```
function moveZ()
{
   if (a.center[2]>30)
      a.center[2] = 0;
   else
      a.center[2] += 15*Suica.dTime;
}
```

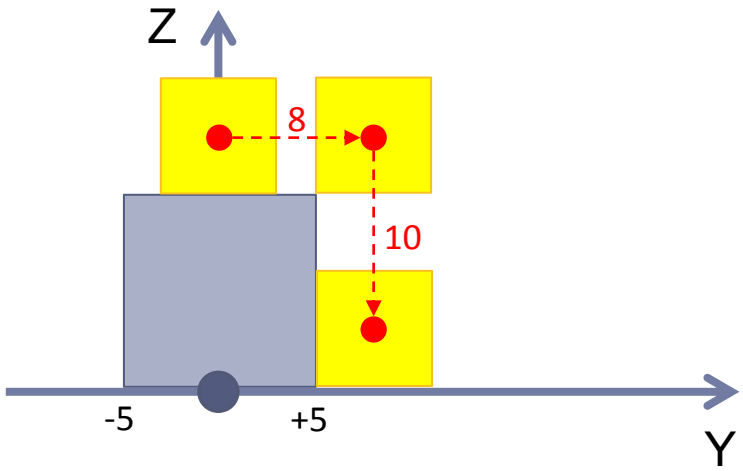


TRY IT

Sequential motion

Two motions

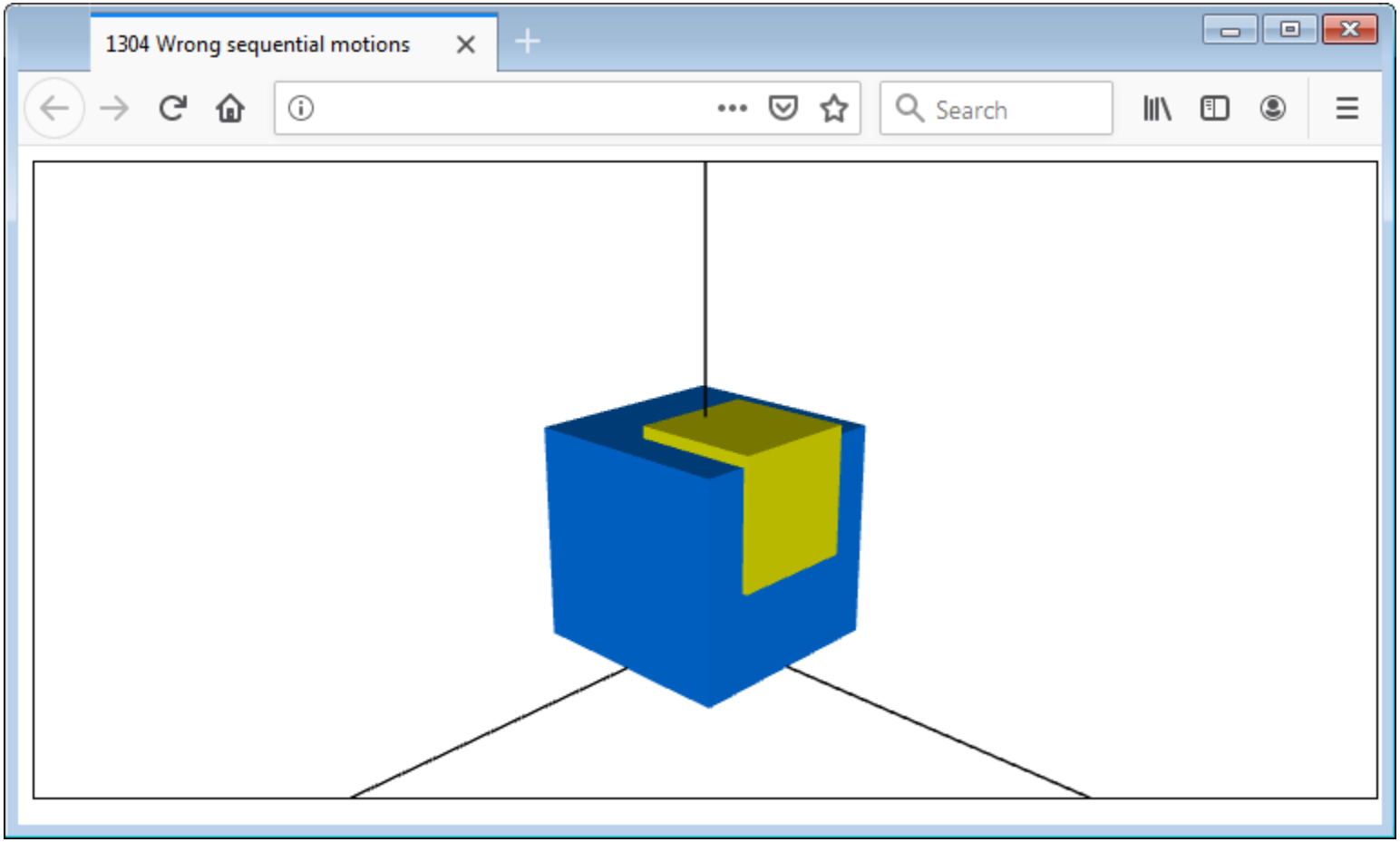
- Yellow cube 6x6x6 on top of blue cube 10x10x10
- The yellow cube slides for 2 seconds and falls for 1 seconds
- Velocity vectors are (0,4,0) and (0,0,-10)



Solution №1

- Both vector are correctly used
- Added return to the initial position
- However, the motion is wrong

```
function move()
{
   a.center[1] += 4*Suica.dTime;
   a.center[2] += -10*Suica.dTime;

   if (a.center[2]<-10)
      a.center = [0,0,10];
}
```

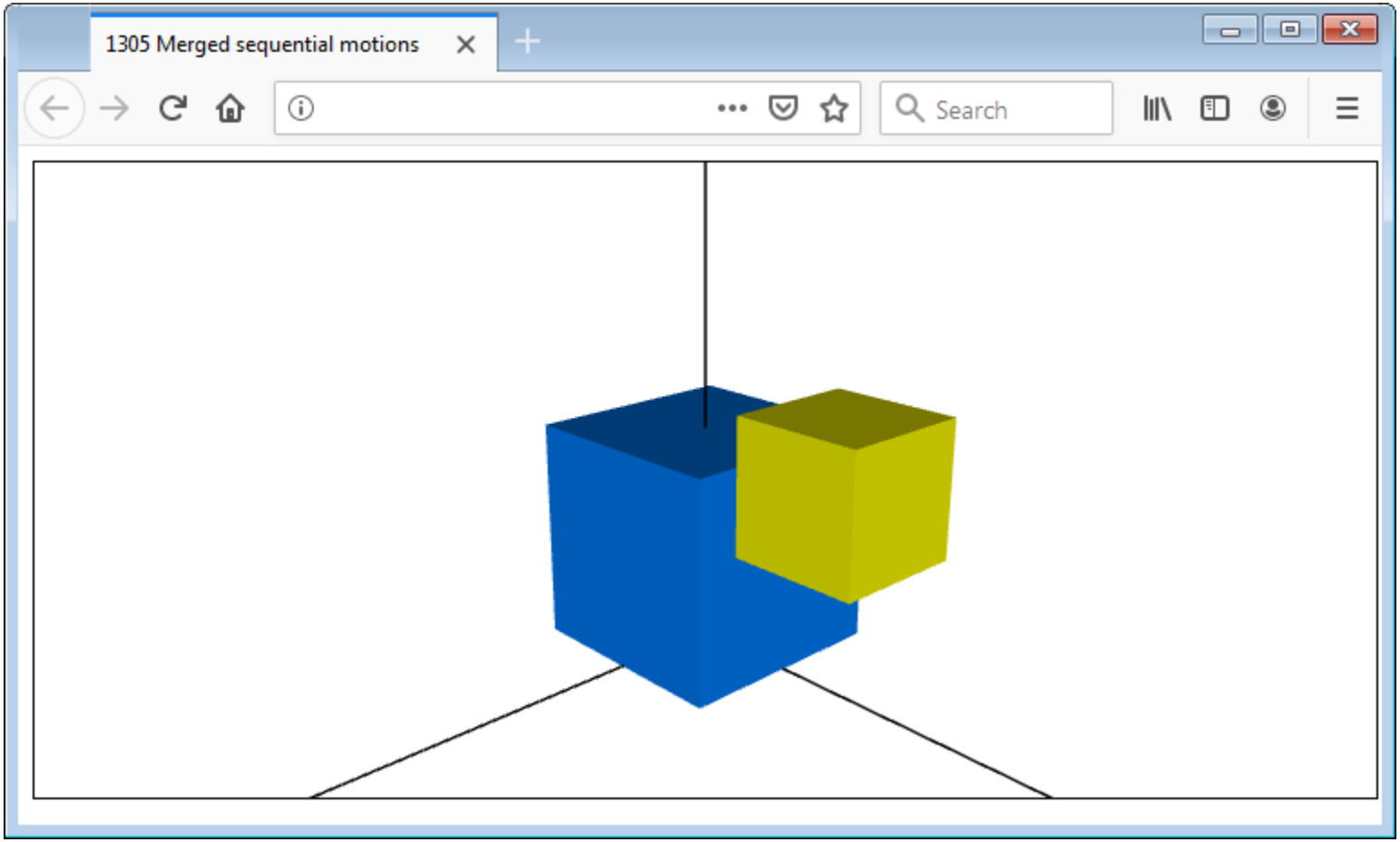


TRY IT

Solution №2

- One loop, split depending on coordinates
- Sliding when y<8, falling when y≥8 and z>0, otherwise no motion

```
function move()
{
   if (a.center[1]<8)
      a.center[1] += 4*Suica.dTime;
   else if (a.center[2]>0)
   {
      a.center[1] = 8;
      a.center[2] += -10*Suica.dTime;
   }
}
```

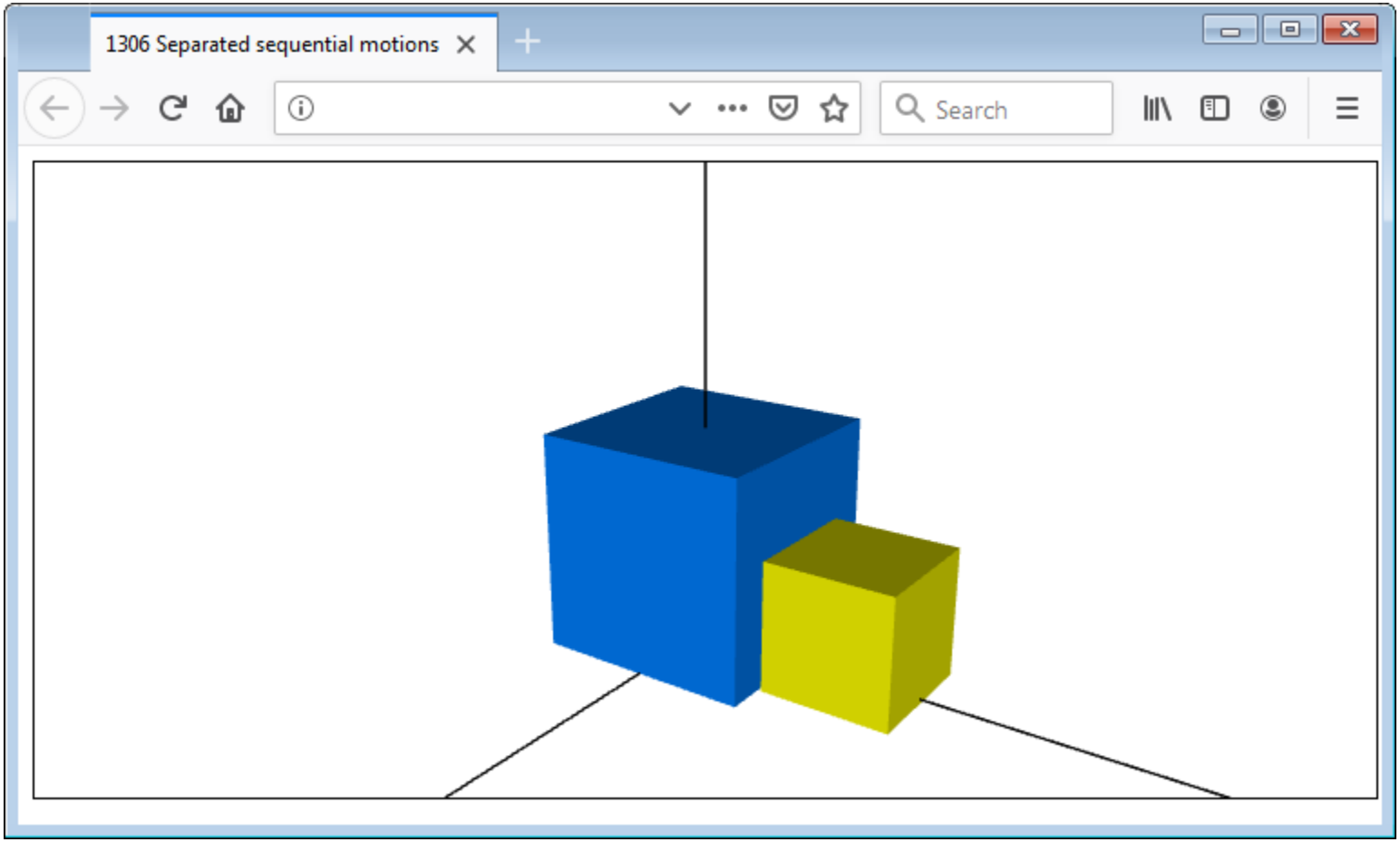


TRY IT

Solution №3

- Separate loops for different animation phases
- Start with a loop for sliding – slide
- Continue with a loop for falling – fall
- Finally removing animation loops

```
function main()
{  ... p.nextFrame = slide; }

function slide()
{  ... if (a.center[1]>=8) p.nextFrame = fall; }

function fall()
{  ... if (a.center[2]<0) p.nextFrame = undefined;}
```



TRY IT

From point to point

From point to point

Motion from point to point

- Most often motion
- Primitive form of motion on a trajectory

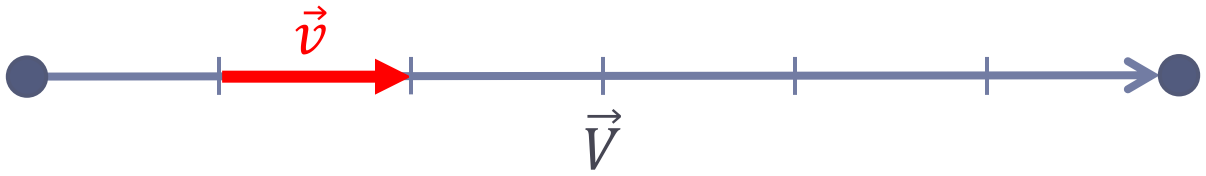
Implementation

- With velocity vector
- With linear combination

Velocity vector

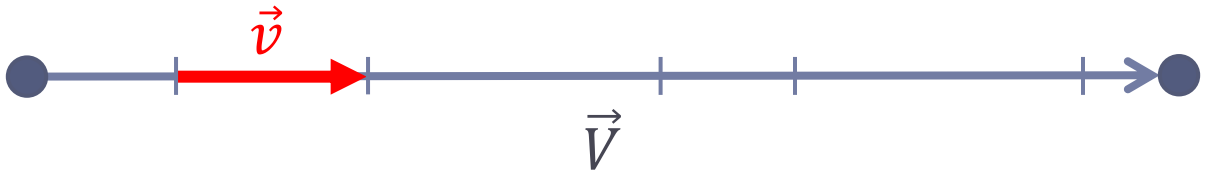
Calculating velocity vector

- Considering the segment as a vector $\vec{V}$
- Defining the desired number of steps (frames) n
- Velocity vector $\vec{v} = \frac{1}{n}\vec{V}$
- Note: $\vec{v}$ is calculated once and the motion only uses the frames, not the elapsed time



Considering time

- Considering the segment as a vector $\vec{V}$
- Defining the desired duration T of the motion
- Velocity vector is $\vec{v} = \frac{\Delta t}{T}\vec{V}$
- Note: $\vec{v}$ is calculated for every frame, because Δt changes continuously



Example

- Two spheres a and b on random locations
- Third sphere c moves from the first to the second sphere
- Motion lasts t=3 seconds

Implementation

- Sphere c copies the center of a with sameAs

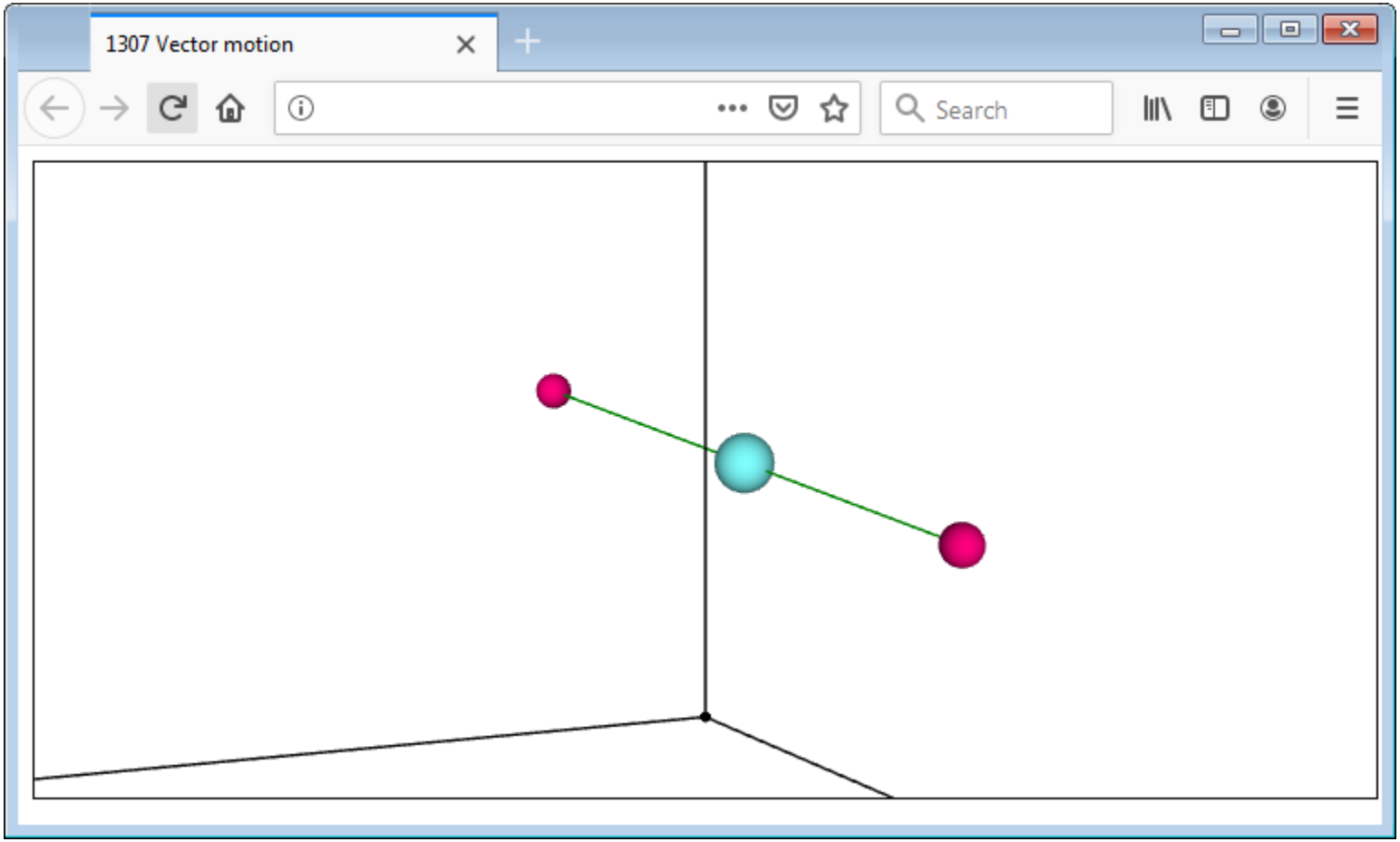
```
a = sphere([random(-15,15),random(-15,15),...);
b = sphere([random(-15,15),random(-15,15),...);
c = sphere(sameAs(a.center),3).custom({...});

v = vectorPoints(b.center,a.center);
t = 3;
```

Main loop

- Until time t has not come, make motion
- The center of c is modified at every frame:

$$\begin{cases} x_c \leftarrow x_c + v_x \dfrac{\Delta t}{T} \\ y_c \leftarrow y_c + v_y \dfrac{\Delta t}{T} \\ z_c \leftarrow z_c + v_z \dfrac{\Delta t}{T} \end{cases}$$

```
function loop()
{
   if (Suica.time<=t)
      for (var i=0; i<3; i++)
         c.center[i] += v[i]*Suica.dTime/t;
}
```



TRY IT

Motion along a ring

- A ring of segments
- Objects moves along the segments
- At the end of a segments it turns smoothly to the next segment

Idea

- The ring is made of connected segments
- Motion is from the beginning to the end of a segment, then start with the next segment and so on
- Object orientation is bound to object main axis

Ring implementation

- There n spheres in a circle, but randomly raised or lowered
- There is a segment between every two neighbour spheres

```
for (var i=0; i<n; i++)
{
   a = 2*Math.PI*i/n;
   b.push(sphere([10*Math.cos(a),10*Math.sin(a),
                                  random(-5,5)],0.15));
}
for (var i=0; i<n; i++)
{
   var j = (i+1)%n;
   segment(b[i].center,b[j].center).custom({...});
}
```

Implementaiton of motion

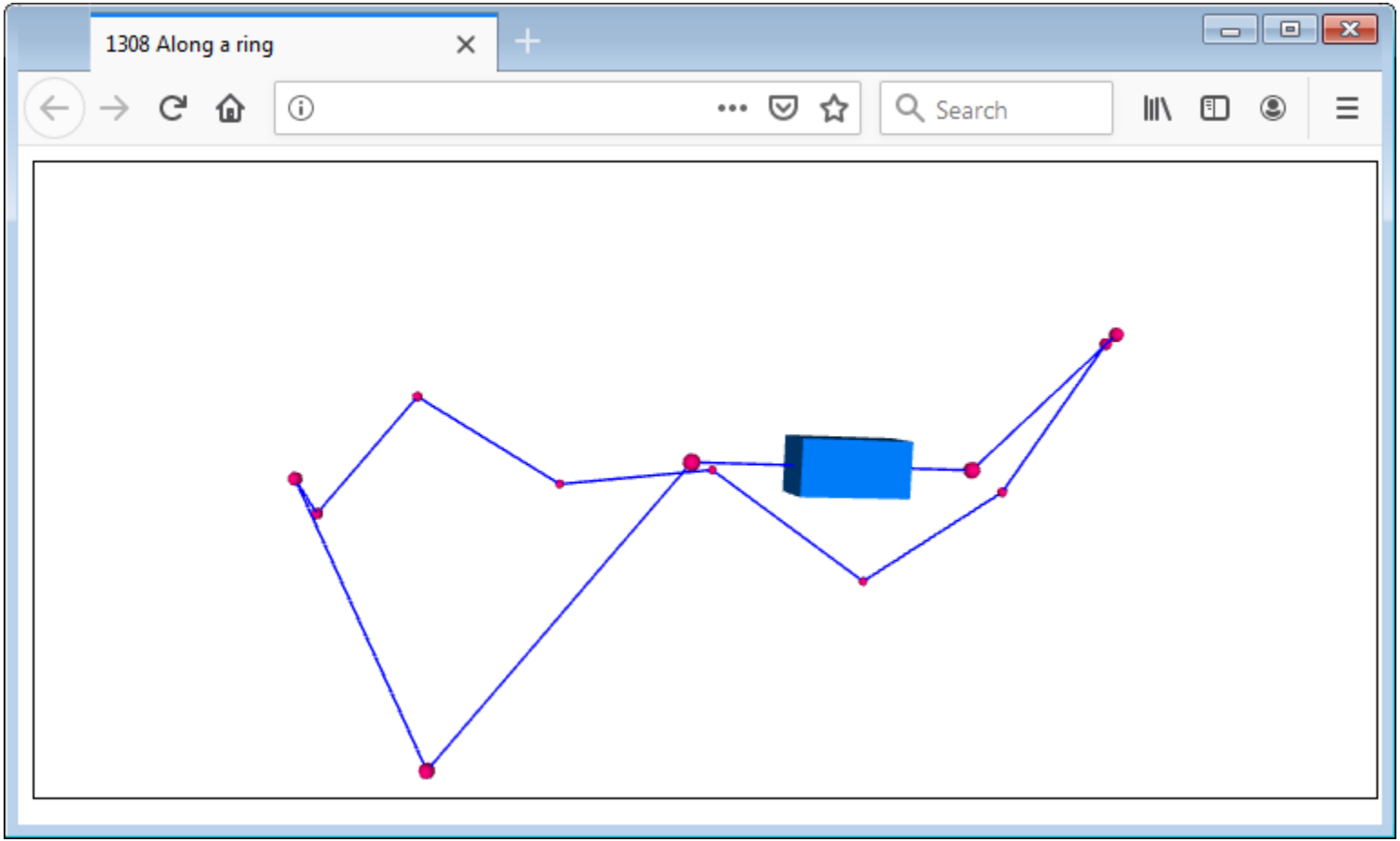
- Counting number of frame in frame
- One segment is passed for 50 frames
- Every 50th frame a new segment is started

```
frame++;
if (frame%50==1)
{
   from = to;
   to = (to+1)%n;
   v = vectorPoints(b[to].center,b[from].center);
   c.center = sameAs(b[from].center);
}
for (var i=0; i<3; i++) c.center[i] += v[i]/50;
```

Implementation of the motion

- The object is a cuboid
- Local Z axis (focus) is along the direction of motion
- Turning is with linear combination – 80% from the current direction and 20% from the new direction (defined in v)

```
c = cuboid([0,0,0],[1,1,2]);
c.focus = [0,0,1];

function loop()
{  ...
   for (var i=0; i<3; i++)
      c.focus[i] = c.focus[i]*0.8+0.2*v[i];
}
```



TRY IT

Linear combination

Motion with linear combination

- Start and end points
- Parameter $k \in [0,1]$ for coordinates
- Intermediate point $A.(1-k)+k.B$ (if motion is $A \rightarrow B$)

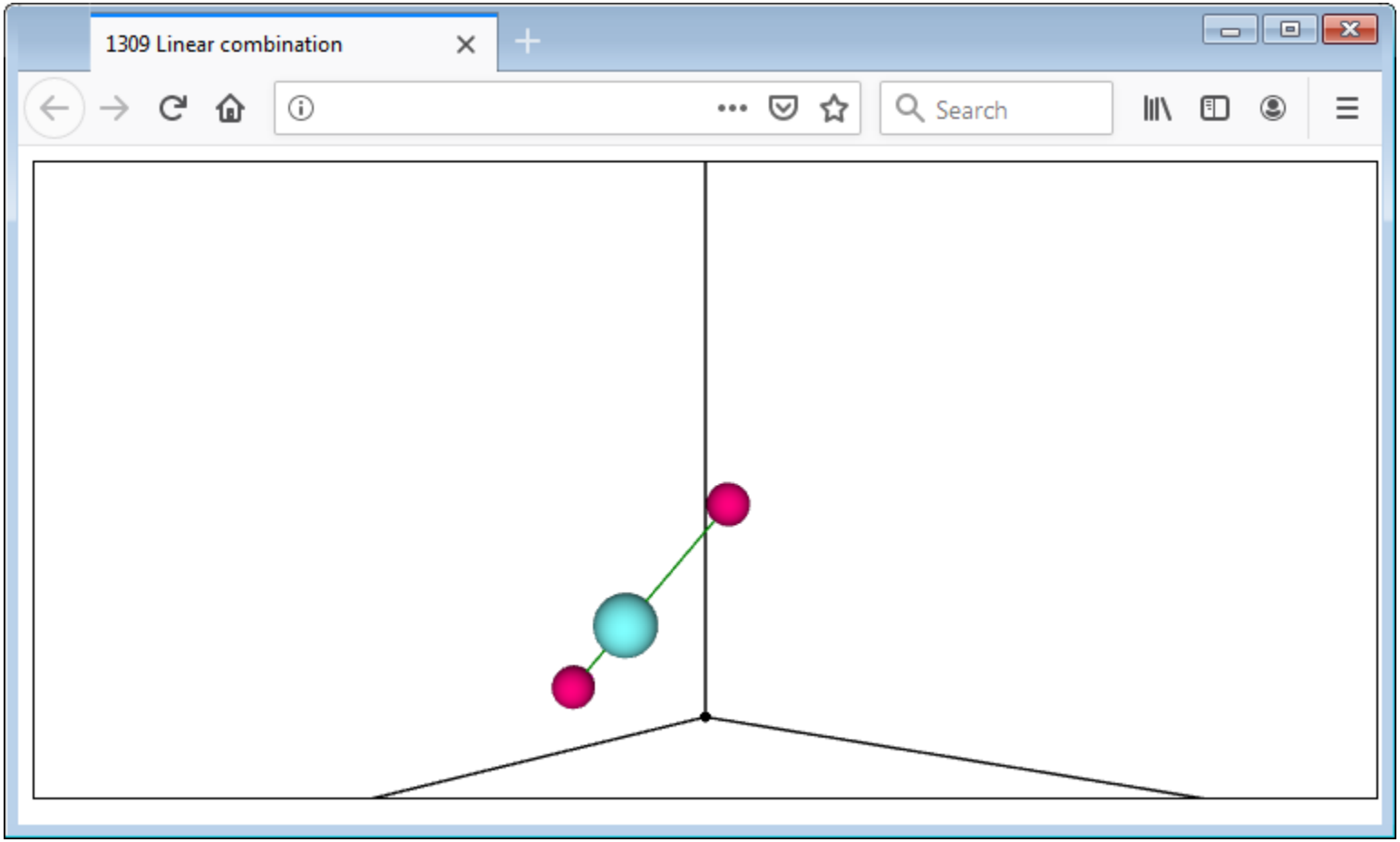
Advantages

- Better control over motion
- Regular and irregular motion
- Flexible start and end points

Implementation

- Parameter k is sinusoidal, transformed from [-1,1] to [0,1]
- Sphere coordinates are a linear combination of the coordinates of the start and end spheres
- Visible effect – motions at both ends is slower

```
function loop()
{
   k = 0.5+0.5*Math.sin(Suica.time);
   for (var i=0; i<3; i++)
      c.center[i] = a.center[i]*(1-k)+k*b.center[i];
}
```



TRY IT

Example

Tennis

- Two rackets move in parallel planes
- The turn slightly as they move
- There is a ball moving fast between them

Idea

- Rackets make harmonic motions
- The ball uses a linear combination of rackets' centers

Rackets

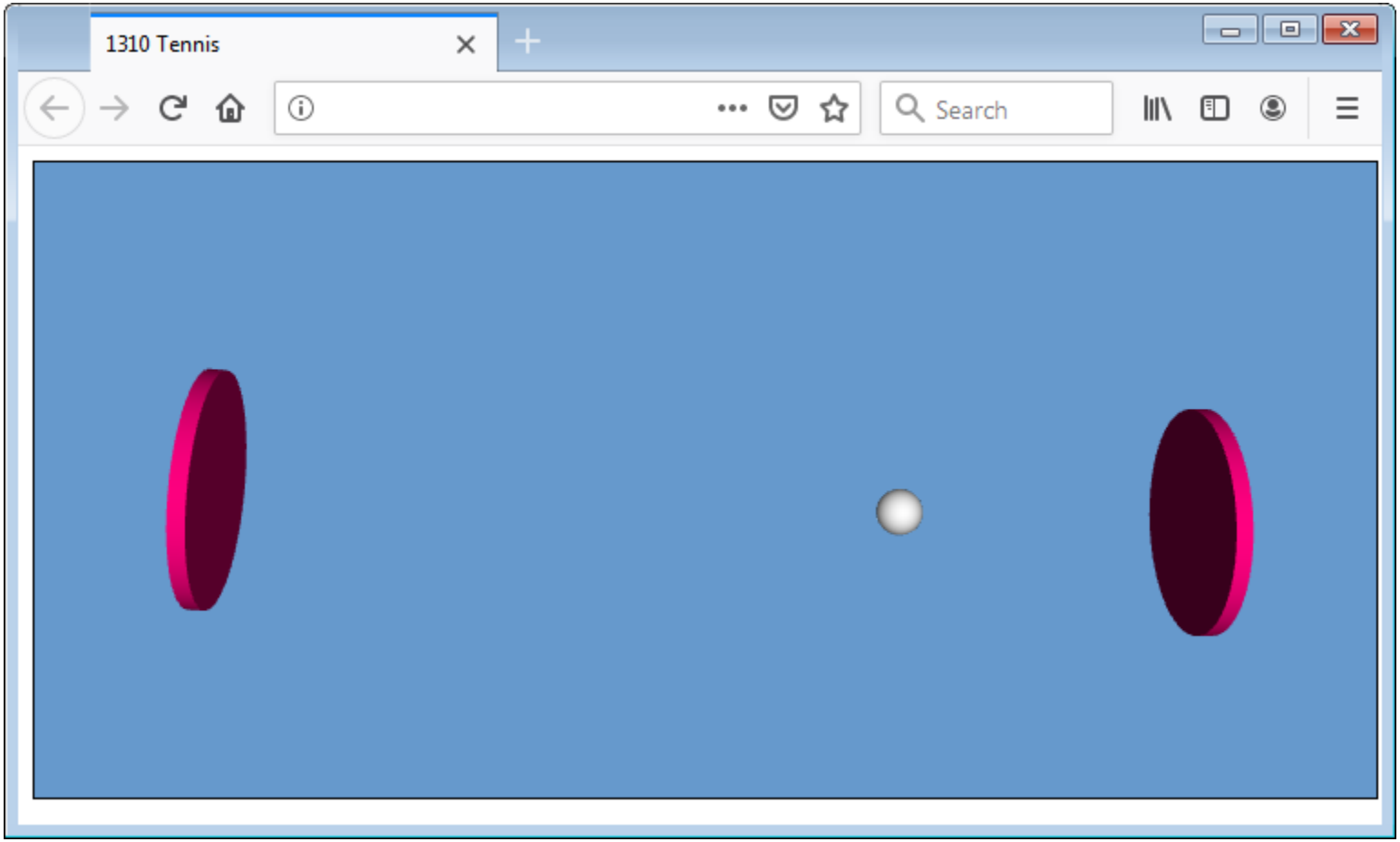
- Flat cylinders
- Motion along X and Z are sinusoidal with different coefficients in order to make motions differ from one another
- Coordinates if racket is used as the local Z axis of the other rackets – this implements their slight rotation

```
sin = Math.sin;
t = Suica.time;
a.center = [15*sin(2.2*t),-30,5*sin(4.7*t)];
b.center = [15*sin(4.3*t),+30,5*sin(2.9*t)];

a.focus = b.center;
b.focus = a.center;
```

Implementation of the ball

- It is a sphere
- Linear combination between the centers of both rackets
- Coefficient k is not [0, 1] but [0.04, 0.96], so that the ball does not penetrate the rackets
- The speed of the ball is 8 times faster than time

```
k = 0.5+0.46*sin(8*t);
for (var i=0; i<3; i++)
   c.center[i] = a.center[i]*(1-k)+k*b.center[i];
```



TRY IT

Summary

Animation

Essence

- Sufficiently fast showing of images
- Perception of motion is via illusion
- Animation with a browser
  - The application sends request to create a frame
  - The browser defines when this is possible

Animation loop

- Drawing a frame is a function pointer by nextFrame
- In Suica.time and Suica.dTime are the elapsed times since the start of Suica and since the previous frame
- Function toFixed rounds fractional numbers

Animation phases

- If phases can be split, each can be implemented as a separate animation loop
- If phases are coexisting, there is one animation loop with conditional execution of phases

Linear motion

Velocity speed

- Defines the direction and the speed of motion
- Suitable when the target is fixed and the speed is fixed

Linear combination

- Allows dynamic change of the position of end points
- Allows non-linear motion

The end

Comments, questions