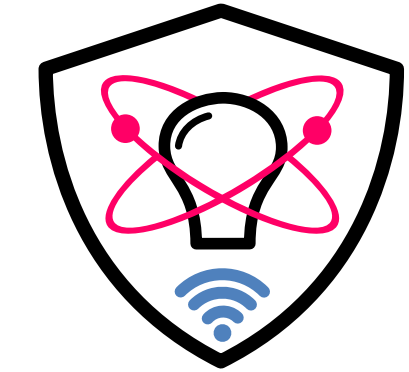


ИКТ в НОС

Влачене

Тема №17

Следене

Следене на мишката

Връзки графичен обект – мишка

- Обект следва мишката
- Обект се мести с мишката

Следене на мишката

- По-лесно се реализира
- Не изисква избиране на текущ обект
- Удобно с ортографска проекция

Видове следене

Твърда връзка

- Обектът е „закачен“ за мишката
- Движи се точно като нея
- Движи се само когато тя се движи

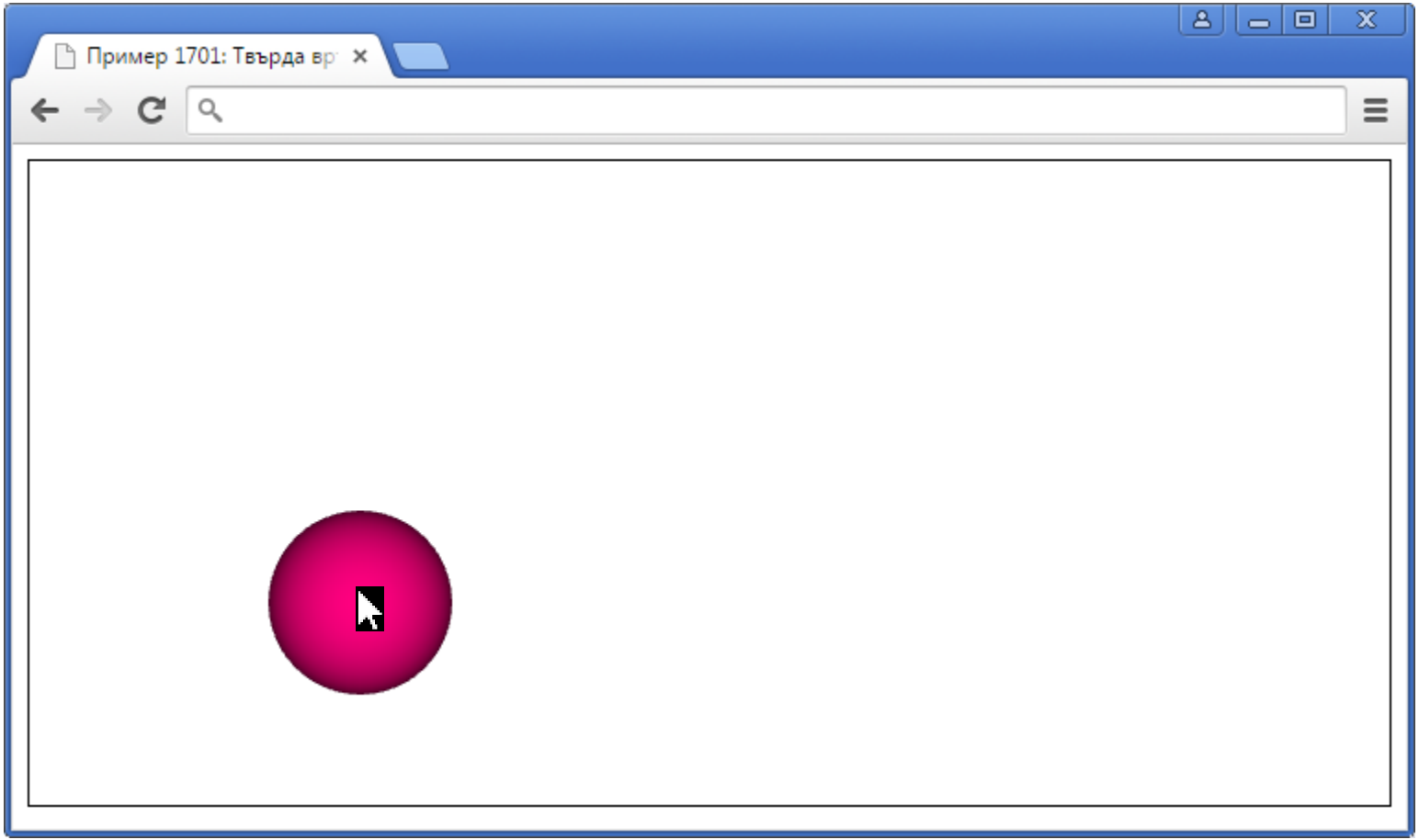
Мека връзка

- Обектът е закачен като с ластик
- Движи се почти като мишката
- Движи се и след като тя спре да се движи

Реализация на твърда връзка

- Координатите на мишката, след преобразуване до графични координати, определят центъра на обекта

```
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];
}
```

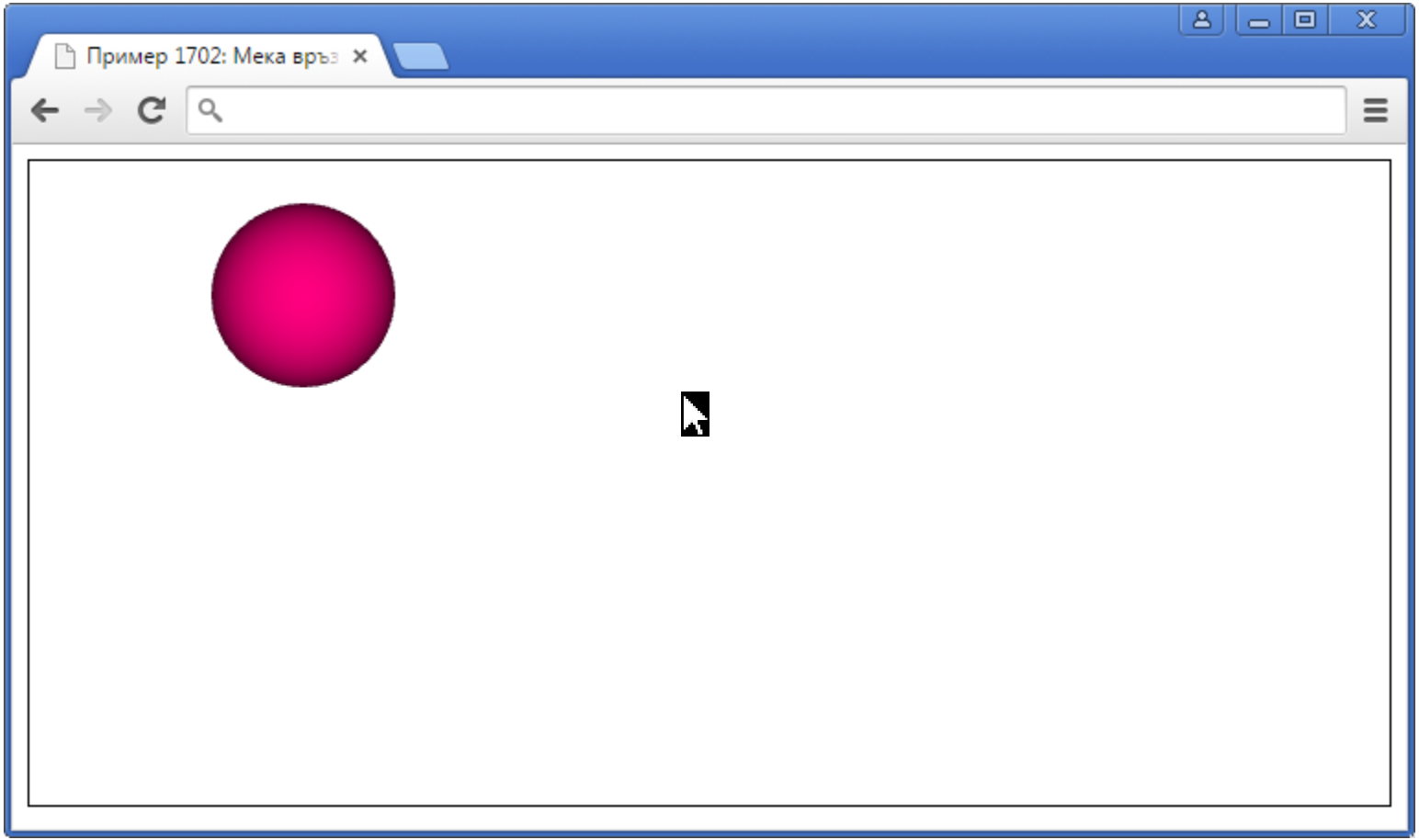


ПРОБА

Реализация на мека връзка

- В mouseMove се запомнят графичните координати в глобалните променливи x и y
- В цикъла за кадри animate се премества обекта с линейна комбинация към запомнените x и 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;
}
```

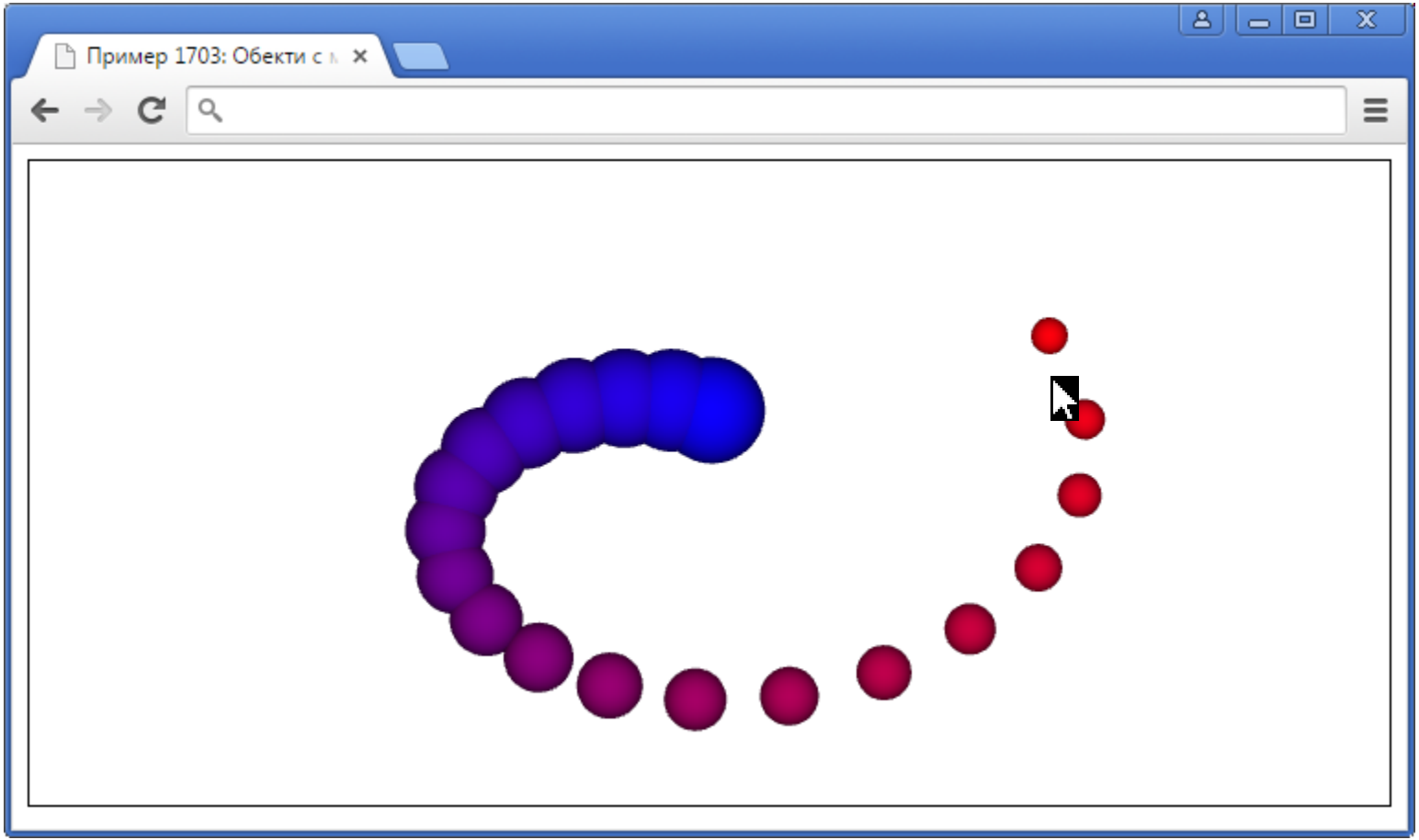


ПРОБА

Верига от меки връзки

- Няколко обекта, всеки свързан меко към предходния
- Аналогична реализация с линейна комбинация

```
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]...
   }
}
```



ПРОБА

Избор на обект

Избиране с мишка

Използване на избиране с мишката

- Избор на графичен обект, който да се манипулира
- Частен случай – влачене на обекти

Проблем

- Гледната точка може да е каквато и да е
- Обектите може да не са с правилни форми
- Обектите може да са съставни и/или сложни

Изчислен избор

Идея

- Изчисляваме зоната на всеки обект
- Проверяваме дали курсорът на мишката е в зоната

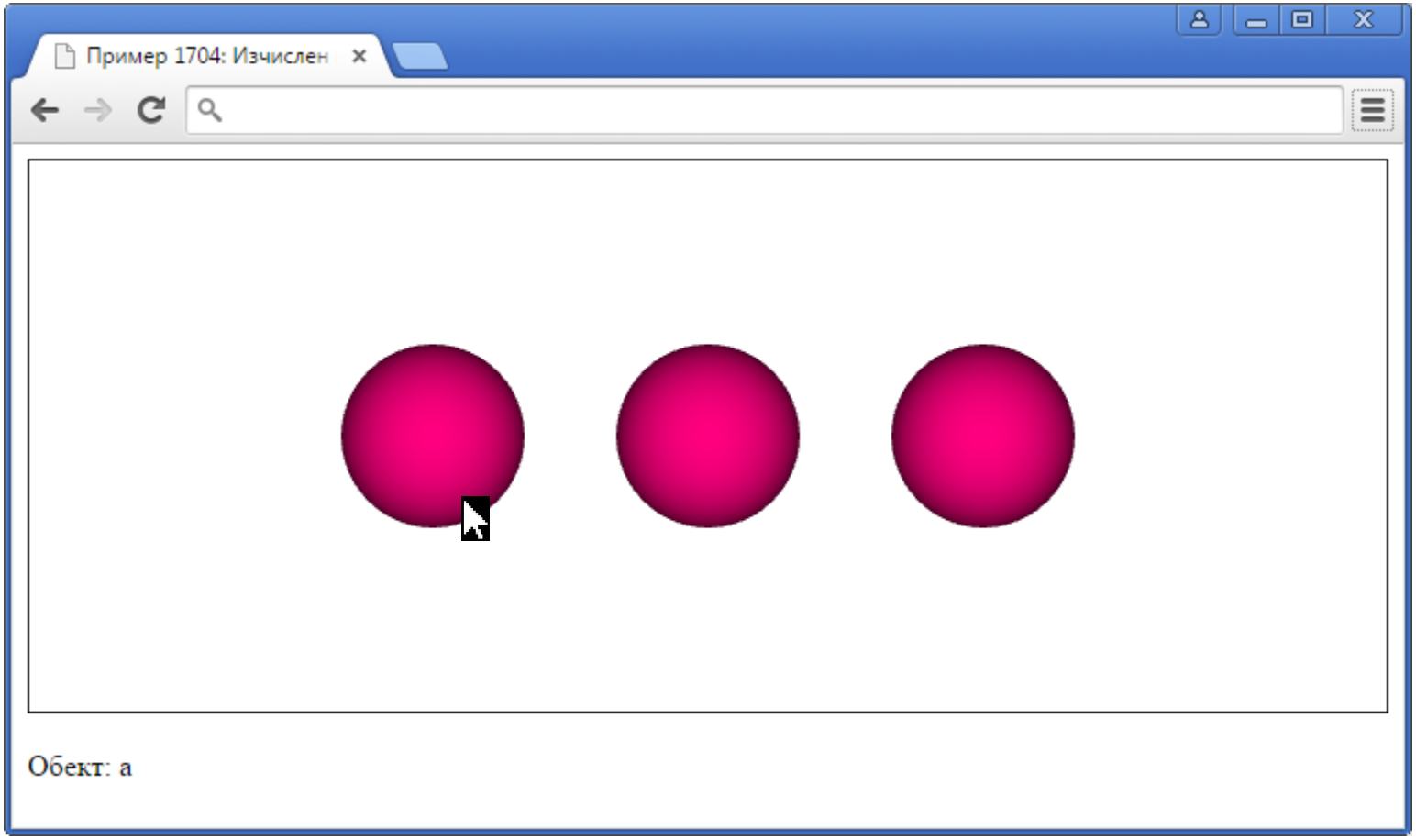
Приложимост

- Само за случаи, когато изчислението е лесно
- Подходяща проекция и гледна точка
- Подходяща форма, позиция и ориентация на обектите

Пример

- Три сфери, ортографска проекция, поглед откъм Z
- Изчисляваме разстоянието между курсора и всеки от центровете на сферите
- Показваме името на избраната сфера

```
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';
```

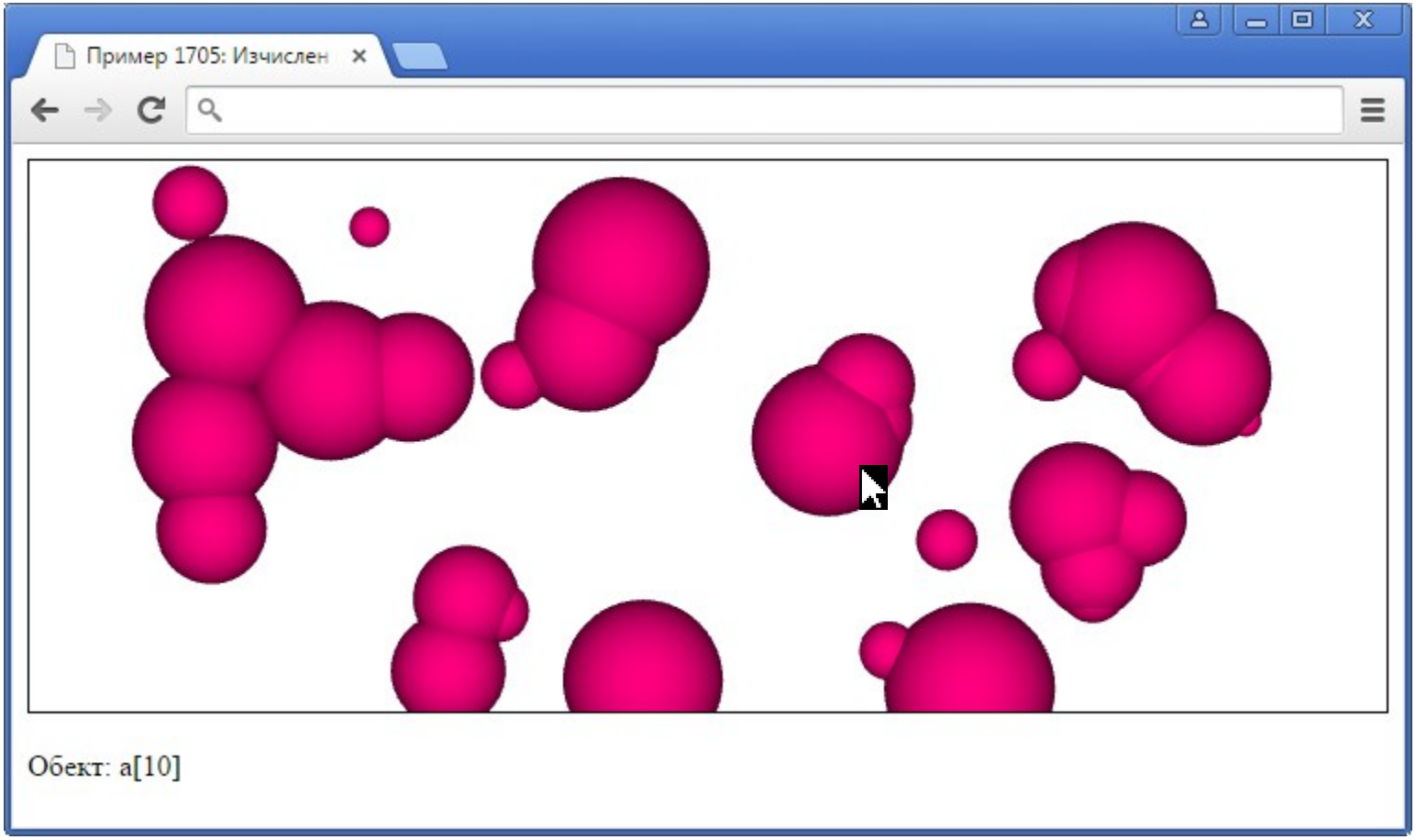


ПРОБА

Променен пример

- Много сфери, с различни радиуси
- Същата идея – пресмятане на разстоянието до центъра и сравняване с радиуса

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



ПРОБА

Произволна форма

Избор на обект с произволна форма

- Метод на обект Suica: objectAtPoint (x, y)
- Връща обекта на даден пиксел или null, ако няма такъв
- Координатите x и y съответстват на clientX и clientY на събитията с мишка

Важно

- Методът проверява само обекти, за които свойството interactive е true

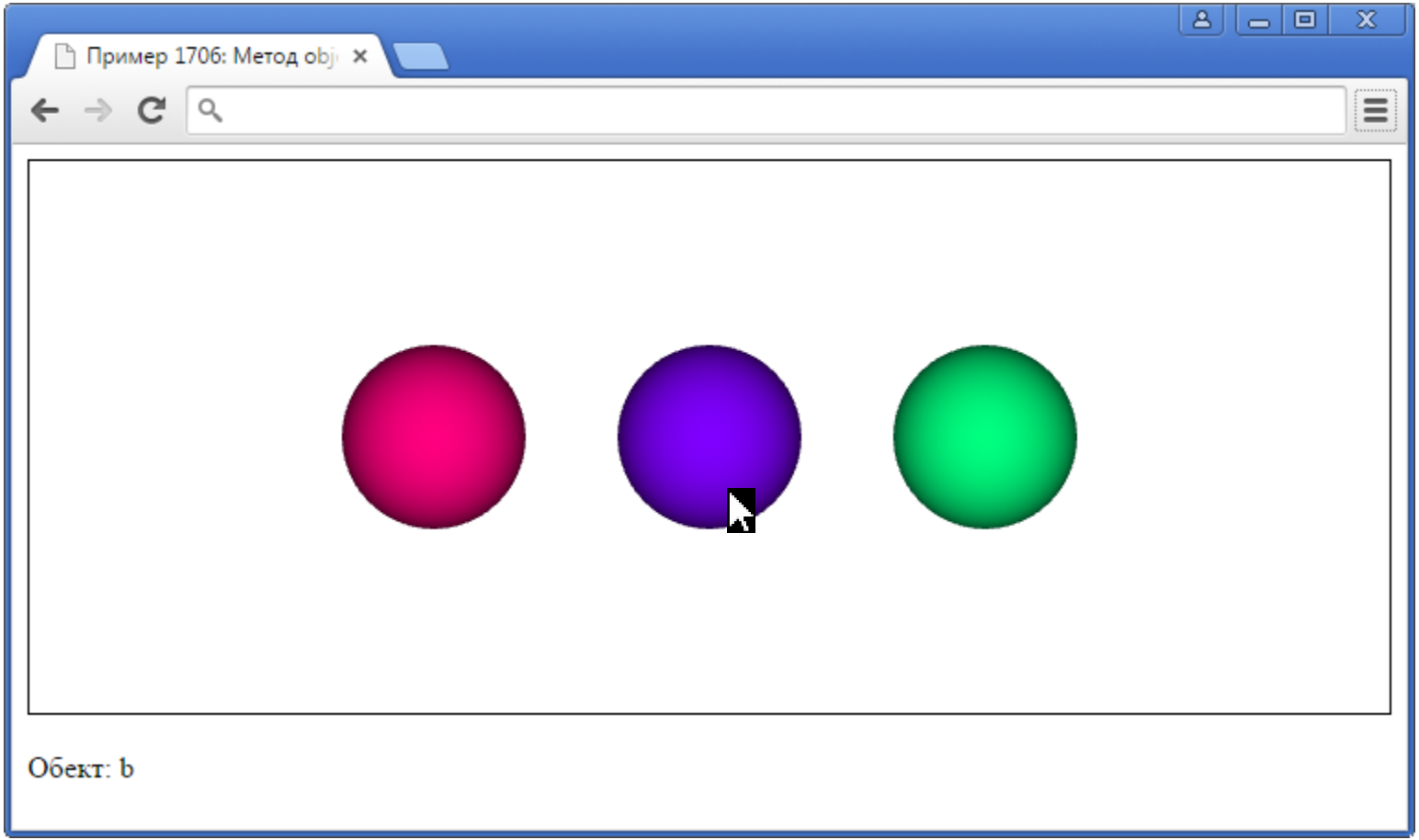
Особеност

- Резултат null се получава и когато има обект, но той не може да бъде идентифициран еднозначно
- Най-често се случва, когато цветът на пиксел е получен от няколко източника:
  - Пиксел по контур на обект включва и цвят от фона
  - Пиксел на границата на два обекта включва цветове и от двата

Пример

- Три сфери с включена с interactive интерактивност
- Избраният обект се намира чрез objectAtPoint и с координати, взети директно от обекта на събитието 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;
   ...}
```



ПРОБА

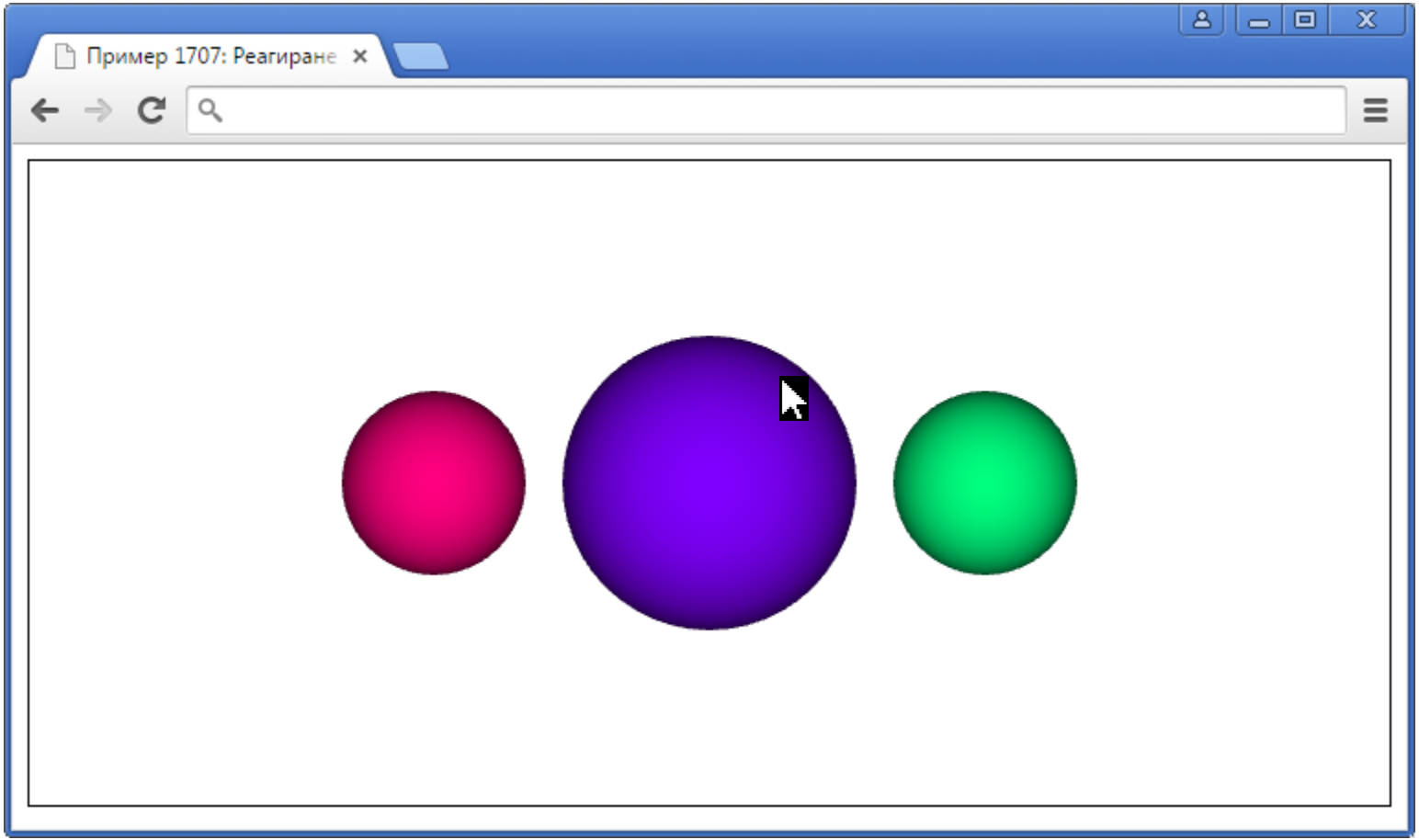
Реакция на избран обект

- Текущо избраната сфера lastObj е с по-голям радиус
- При избор на нова сфера newObj, старата си възвръща оригиналния размер, а новата става голяма

```
var lastObj;

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

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



ПРОБА

Пример

Пръстен от колони

- Правилни паралелепипеди са подредени в кръг
- Всички са с еднаква височина
- При преминаване с мишката над колона, тя става малка
- Всички колони растат плавно до първоначалния размер
- През цялото време сцената се върти

Реализация

- Колоните са паралелепипеди с отместен център origin
- Завъртането със spin ги разполага по невидима окръжност
- Всички обекти са интерактивни, за да могат да бъдат намерени от 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,
         }));
```

- Въртенето на сцената е с lookAt (времето е забавено 4 пъти)

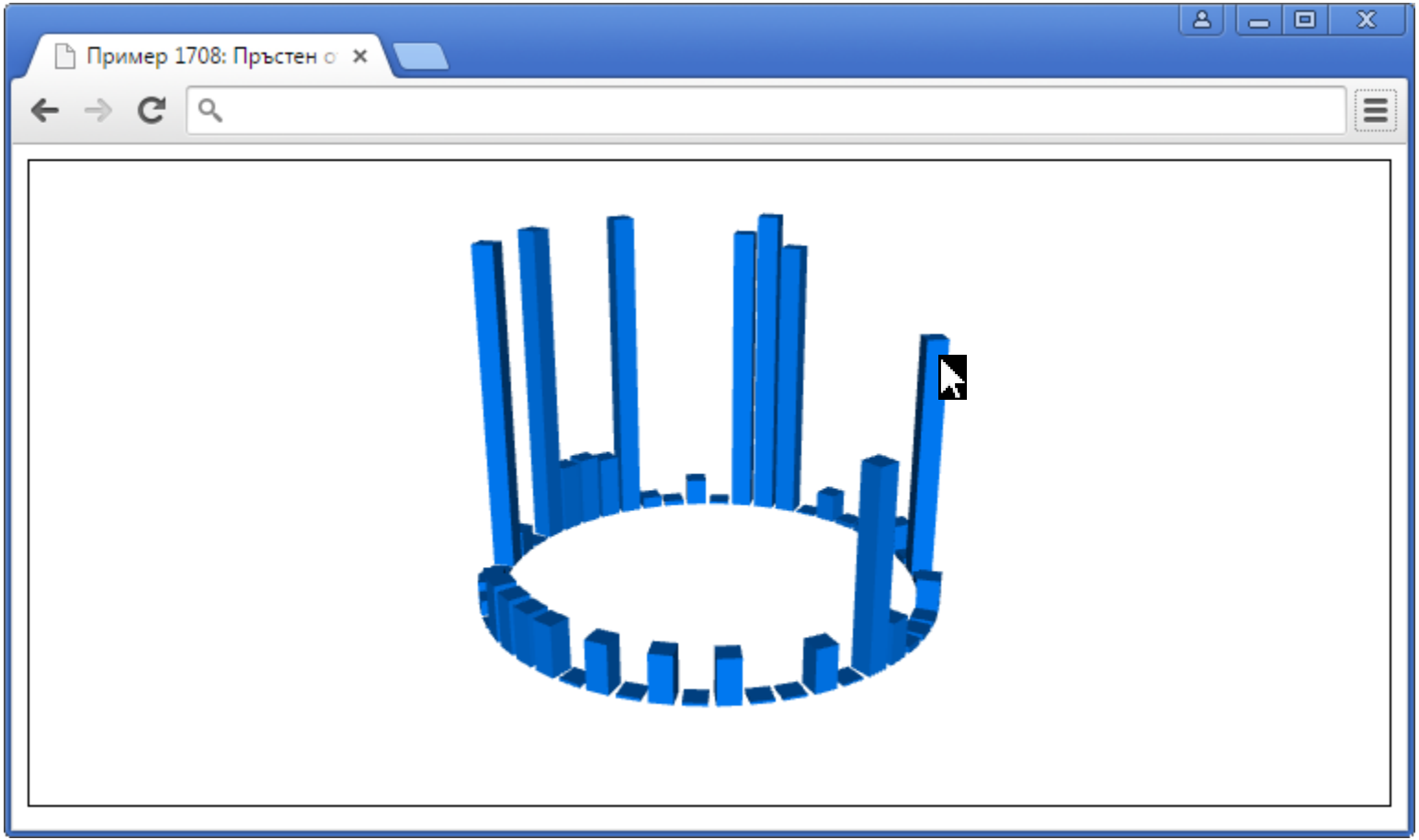
```
var t = Suica.time/4;
lookAt ([50*Math.cos(t),50*Math.sin(t),20],...);
```

- Плавното нарастване на колоните е с 2% на кадър и се прилага докато височината е по-малка от 15

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

- При движение на мишката търсим обекта obj под нея
- Ако има такъв – смаляваме му височината

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



ПРОБА

Влачене

Етапи

Етапи на влаченето

- Натискане на бутон – избиране на обект
- Движение на мишката – промяна на обекта
- Пускане на бутона – пускане на обекта

Комбиниране при натискане на бутон

- Ако е хванат обект, започва неговото влачене
- Ако няма хванат обект, започва да се влачи гледната точка (напр. да се върти сцената)

Наивно влачене

Събития

- Улавяме трите събития mousedown, mouseup и mousemove
- При натискане на бутон намираме обекта в obj

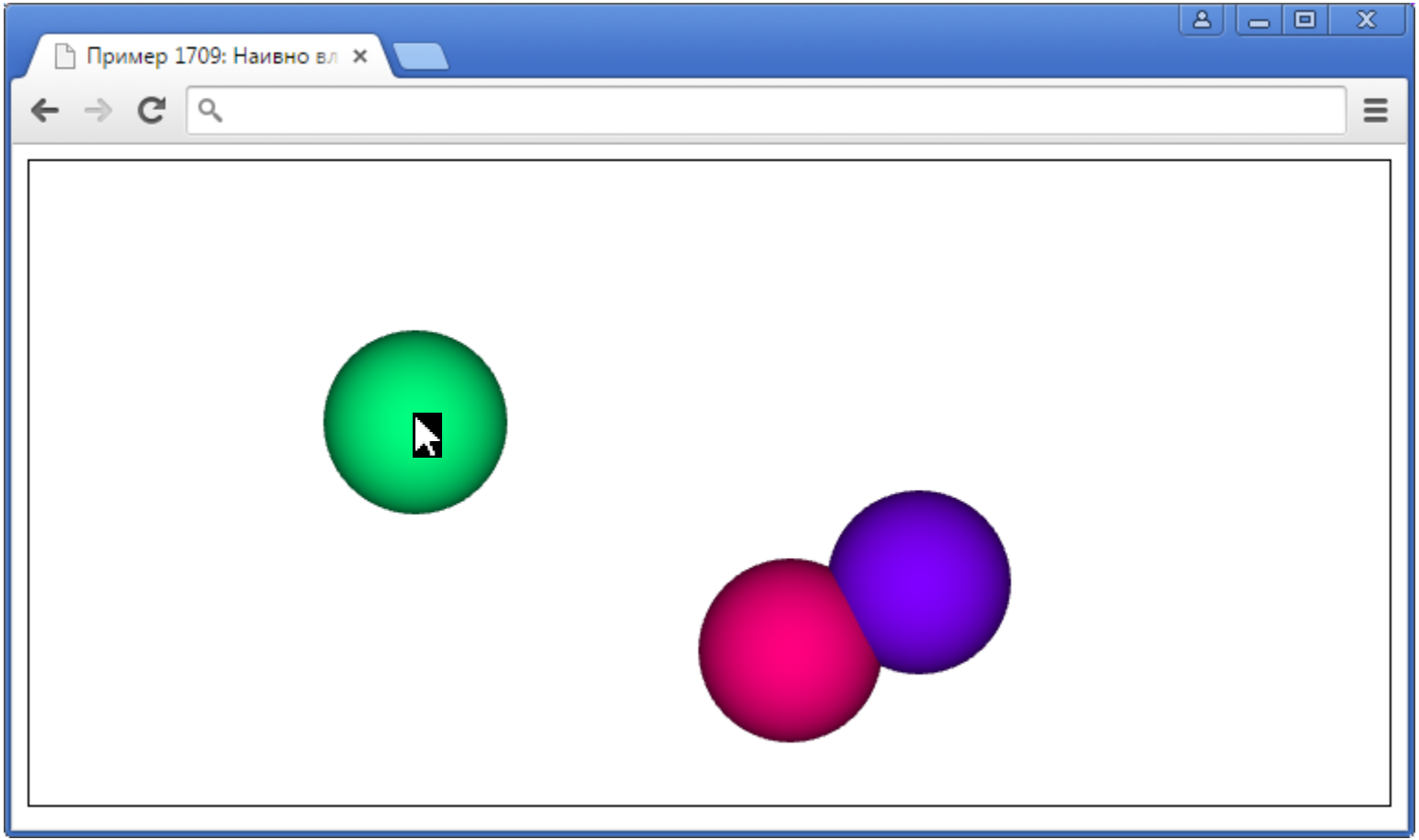
```
p.gl.canvas.addEventListener('mousedown',...);
p.gl.canvas.addEventListener('mouseup',...);
p.gl.canvas.addEventListener('mousemove',...);

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

Събития

- При пускане на бутон забравяме, че има избран обект
- При движение, ако има избран обект, сменяме центъра му

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

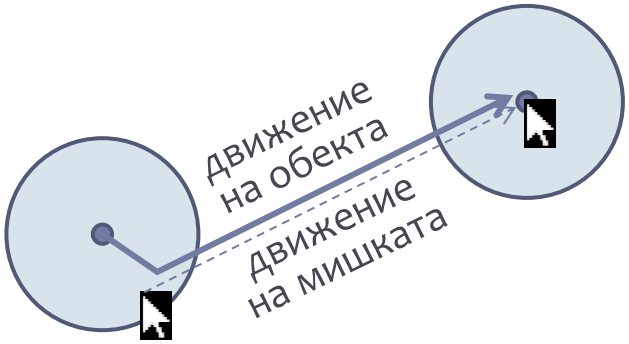


ПРОБА

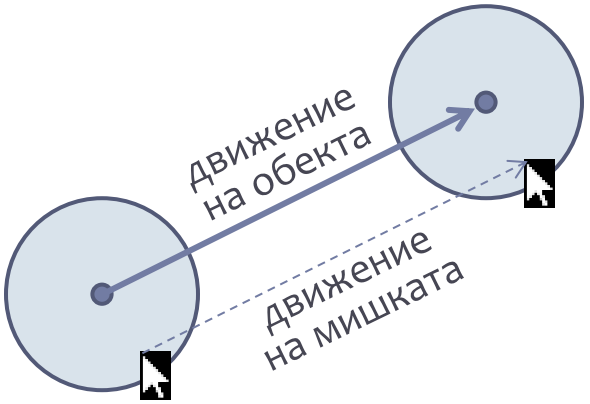
Дефект

Влаченето не е „естествено“

- Обектът винаги се „центрира“ независимо къде е хванат



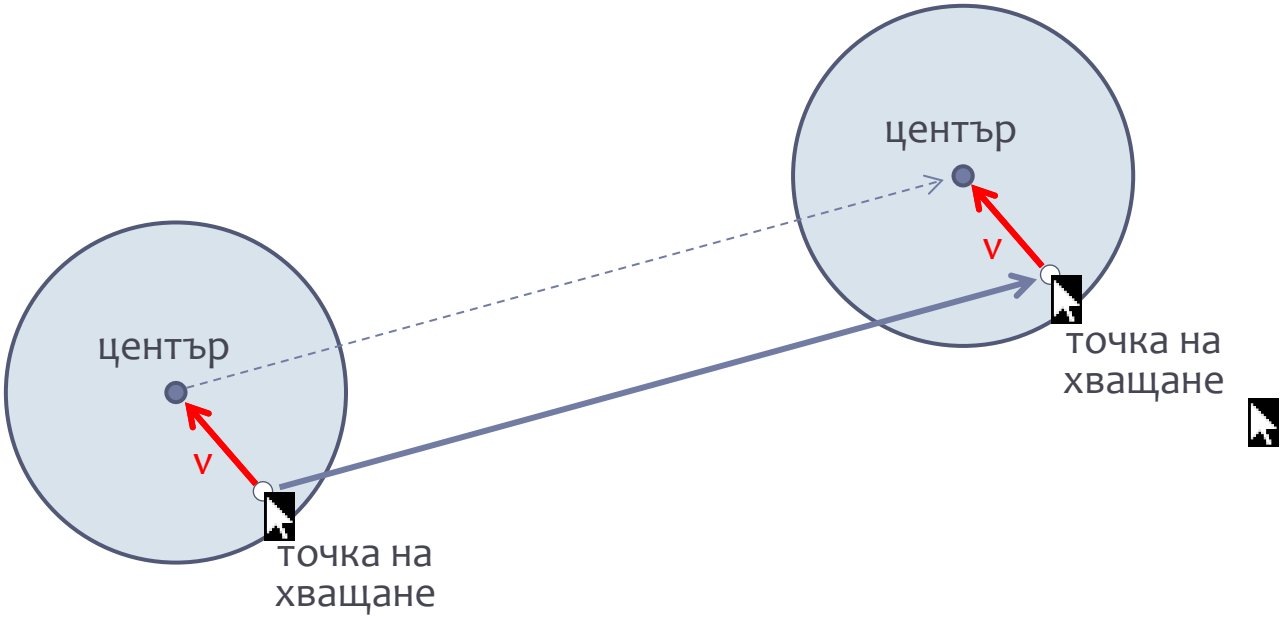
Наивно влачене



Желано влачене

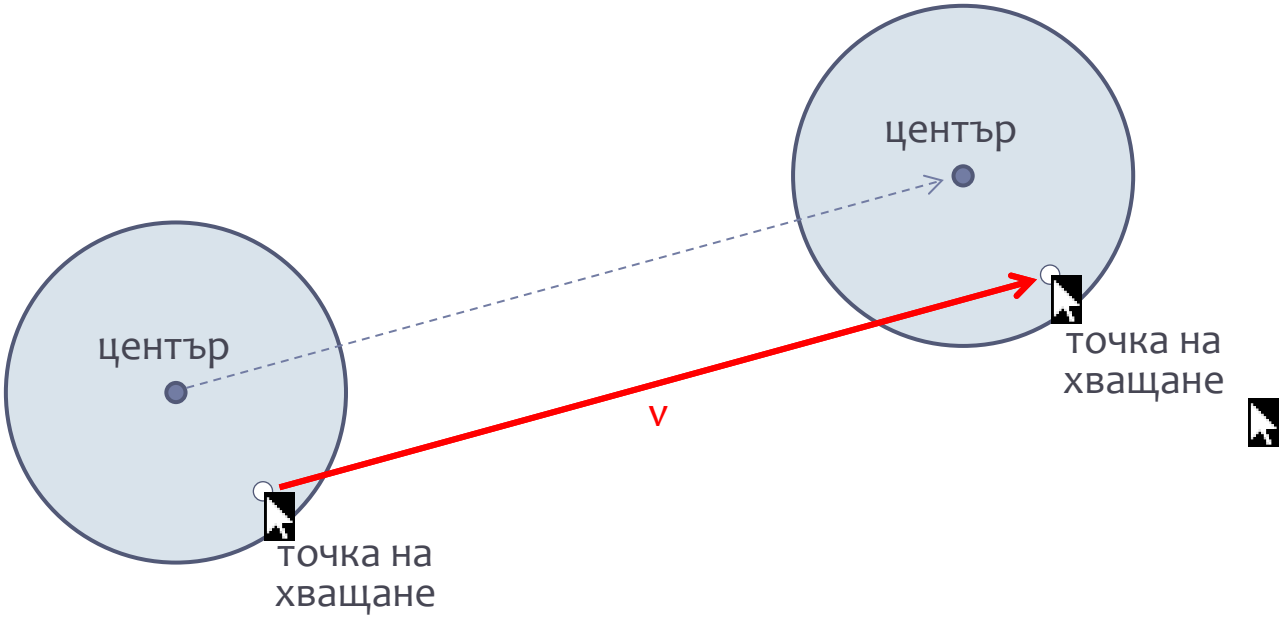
Решение №1

- Знаем вектора v от точката на хващане до центъра
- При движение, отместваме центъра спрямо позицията на мишката с този вектор



Решение №2

- Знаем вектора v от точката на хващане до текущата позиция
- При движение, отместваме центъра с този вектор



Сравнение на двете решения

Решение №1	Решение №2
Векторът v се изчислява еднократно в началото на влачението	Векторът v се изчислява на всяка стъпка от влачението
Не е нужно да помним последните координати	Трябва да помним последните координати
Подходящо при влачене, зависещо само от общото отместване	Подходящо при влачене, зависещо от относителното отместване
Улеснява само традиционното влачене	Улеснява допълнителни ефекти – мащабиране, инерция и т.н.

Влачене

Реализация на решение №2

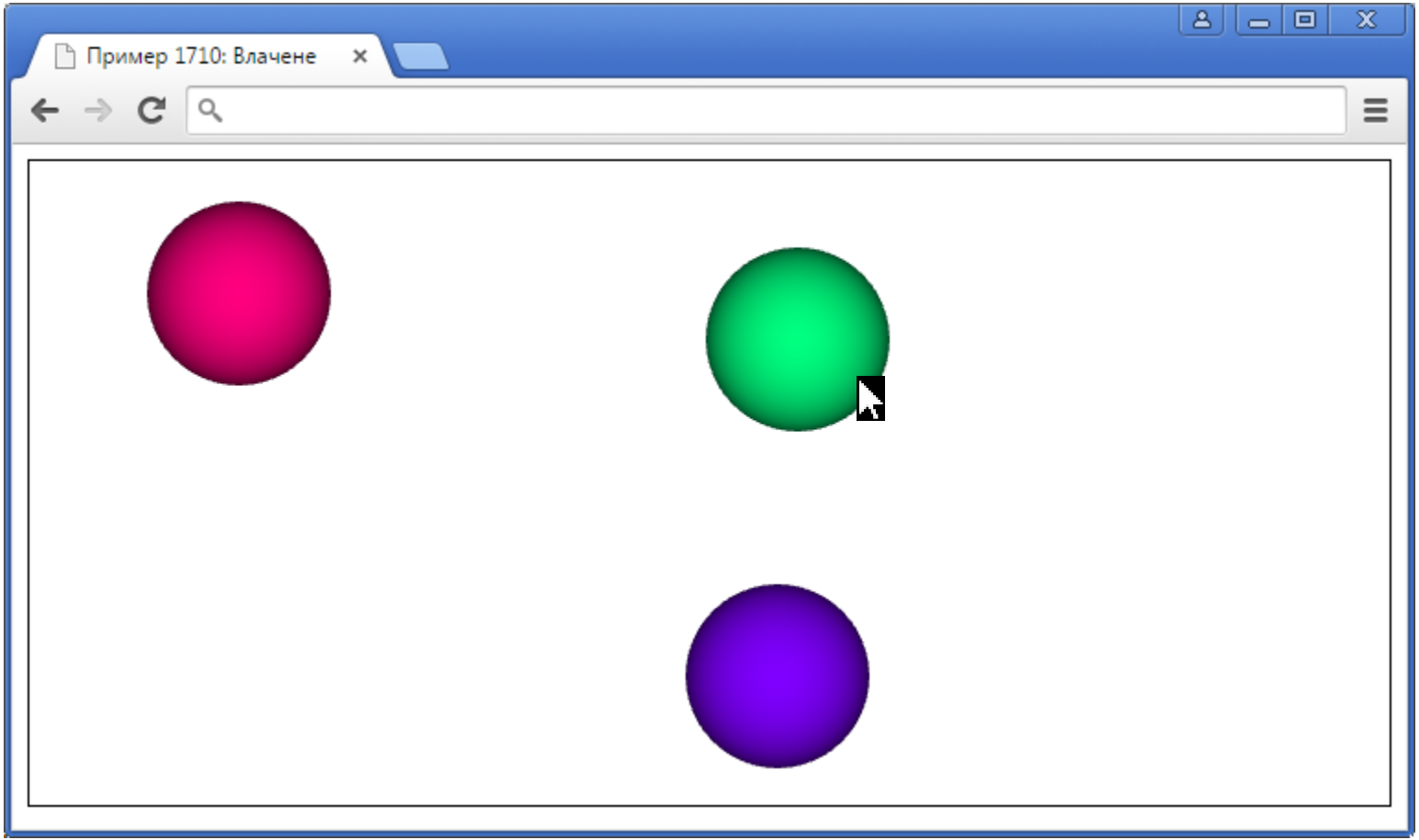
- При натискане на бутон намираме обекта obj
- Запомняме координатите x и y – не преобразуваме в локални координати, работим с относително движение

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

- При движение се променя центъра според отместването на курсора, спрямо последните запомнени координати
- По X прибавяме, а по Y изваждаме (посоките на осите са такива, че по X съвпадат, а по Y са противоположни)
- Запомняме последните x и y

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

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



ПРОБА

Влачене на сцена

Интерактивност в 2D

Примерна цел

- Имаме 2D сцена
- Искаме да плъзгаме сцената интерактивно
- Искаме да мащабираме сцената интерактивно

Примерен интерфейс

- С ляв бутон на мишката плъзгаме
- С десен бутон и вертикално движение –мащабираме

Реализация

- Ловим mousedown и mousemove, без mouseup
- Забраняваме контекстното меню през contextmenu
- При натискане на бутон само запомняме координатите x и 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;
}
```

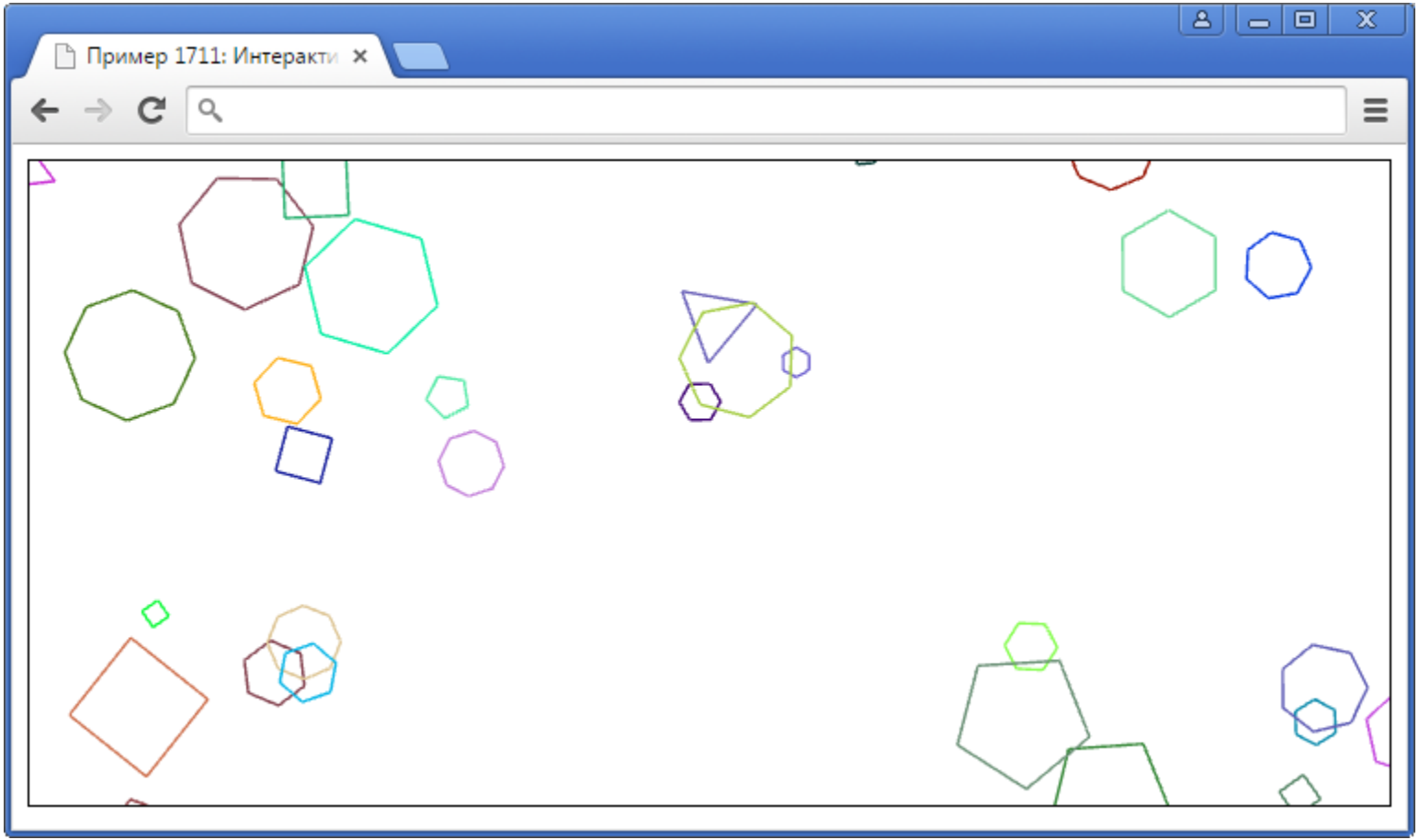
Движение на мишката

- Мястото на гледната точка се определя от lookX и lookY
- Мащабът на сцената е реализиран като отдалеченост чрез коефициента lookS

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

- При натиснат ляв бутон отместването на мишката се прехвърля към lookX и lookY
- Отместването се мащабира с lookS
- При натиснат десен бутон се променя мащабът 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);
}
```



ПРОБА

Интерактивност в 3D

Примерна цел

- Имаме 3D сцена
- Искаме да въртим сцената интерактивно
- Искаме да мащабираме сцената интерактивно

Примерен интерфейс

- С ляв бутон на мишката въртим хоризонтално и вертикално
- С десен бутон и вертикално движение –мащабираме

Реализация

- Гледната точка е по сфера с радиус lookD
- Ъгловите координати са lookA и lookB
- Целта е фиксирана на (0,0,0), а посоката нагоре на (0,0,1)

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

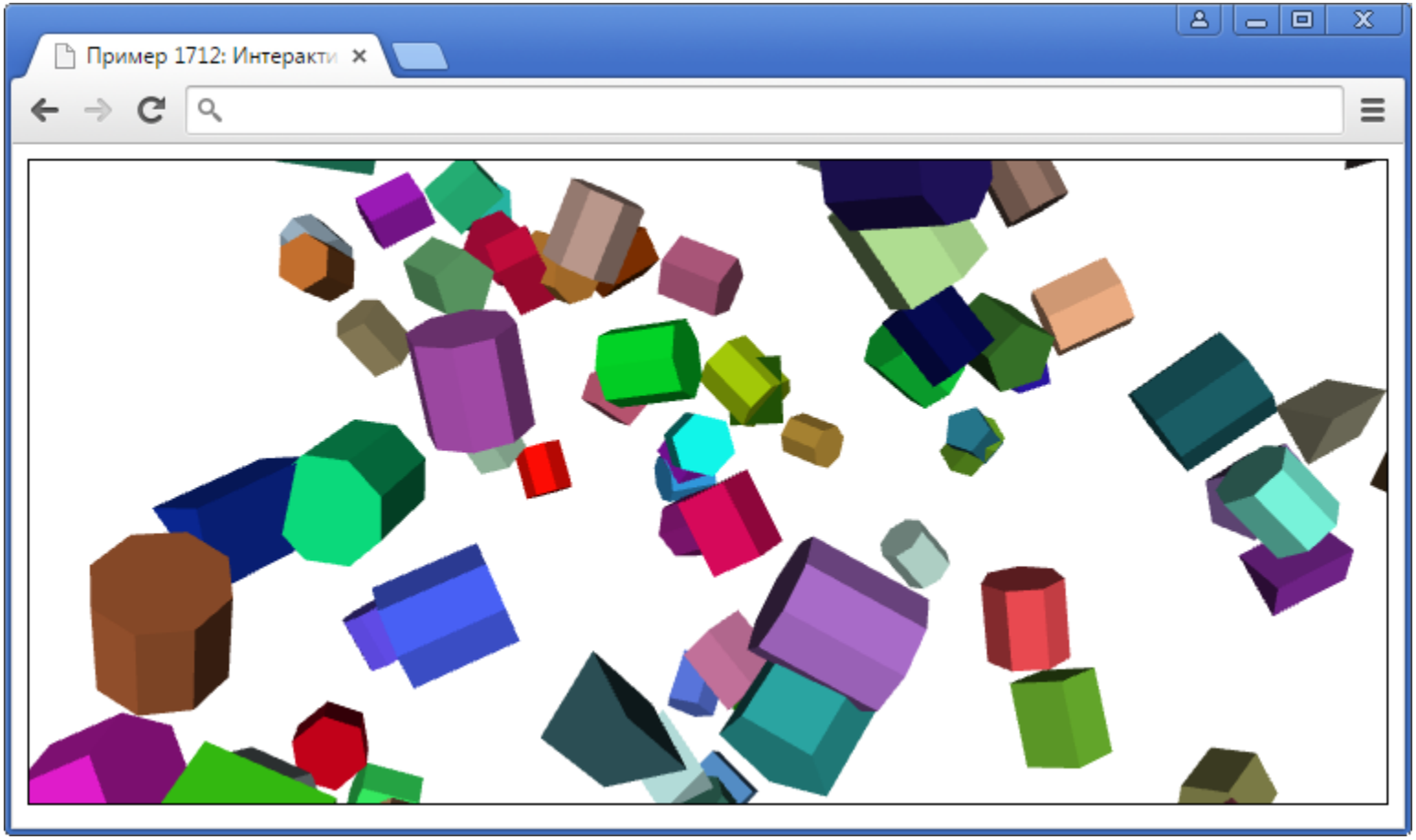
- Разстоянието е с повдигане на степен (защо?)
- Добавени са ограничения за допустимо разстояние

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

- Двата ъгъла на сферични координати са свързани към хоризонталното и вертикалното движение на мишката
- Коефициентът 200 отговаря за скàлите – преместване на 200 пиксела съответства на завъртане на 1 радиан
- Ограничението за вертикален ъгъл е да се попречи сцената да се погледне точно отгоре или точно отдолу (тогава векторът нагоре (0,0,1) се вижда като нулев вектор)

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

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



ПРОБА

Обобщение

Следене и избор на обект

Следене на мишката

- Твърда връзка – разстоянието е фиксирано
- Мека връзка – разстоянието се променя плавно

Избор на обект

- Чрез изчисляване на зоната, върху която се намира
- Чрез вградения метод objectAtPoint
- Само обекти със включено свойство interactive са видими от objectAtPoint

Влачене

Влачене с мишката

- Стъпка 1 – хващане (избор) на обект
- Стъпка 2 – движение (следване) на обект
- Стъпка 3 – пускане на обект

Влачене на сцената

- Чрез влачене на гледната точка
- Позволява интерактивно въртене
- Позволява интерактивно движение в сцена

ИКТ в НОС

Край

Коментари, въпроси