ИКТ в НОС

Влачене с ограничения

Тема №18

Ограничения

Влачене с ограничения

Предната лекция

- Свободно влачене
- Местоположението зависи само от мишката

В реалност

- Движението на обекта е ограничено
- Разнообразни причини за ограничаване
- Неизбежно разминаване на мишката и обекта

Причини за ограниченията

Концептуални (породени от дизайна)

- Движение на точка по окръжност
- Движение на точка по повърхността на сфера
- Движение на точка по ръба на куб

Физически (от външни причини)

- Програмата не работи за всички стойности
- Трудно се поддържа плавност на движението
- Ограничения в размера на екрана

Идея за реализация

Движението е параметрично

- Винаги. Най-малкото, параметри са координатите на курсора на мишката

Техники

- Ограничаване на параметрите (по-рядко)
- Ограничаване на резултатите (по-често)
- Намиране на най-удобните резултати

Какво да очакваме

- Не влачим обекти, само ги преместваме
- Курсорът има значение при хващането
- При движение курсорът се разминава с обекта

Цели

- Преместването на обект да е най-интуитивно
- И по-възможност най-лесно за реализация

Влачене по права линия

Влачене по права и отсечка

С линейна комбинация

- С мишката контролираме параметър
- От него с линейна комбинация получаваме точка от правата или отсечката

С най-близка точка

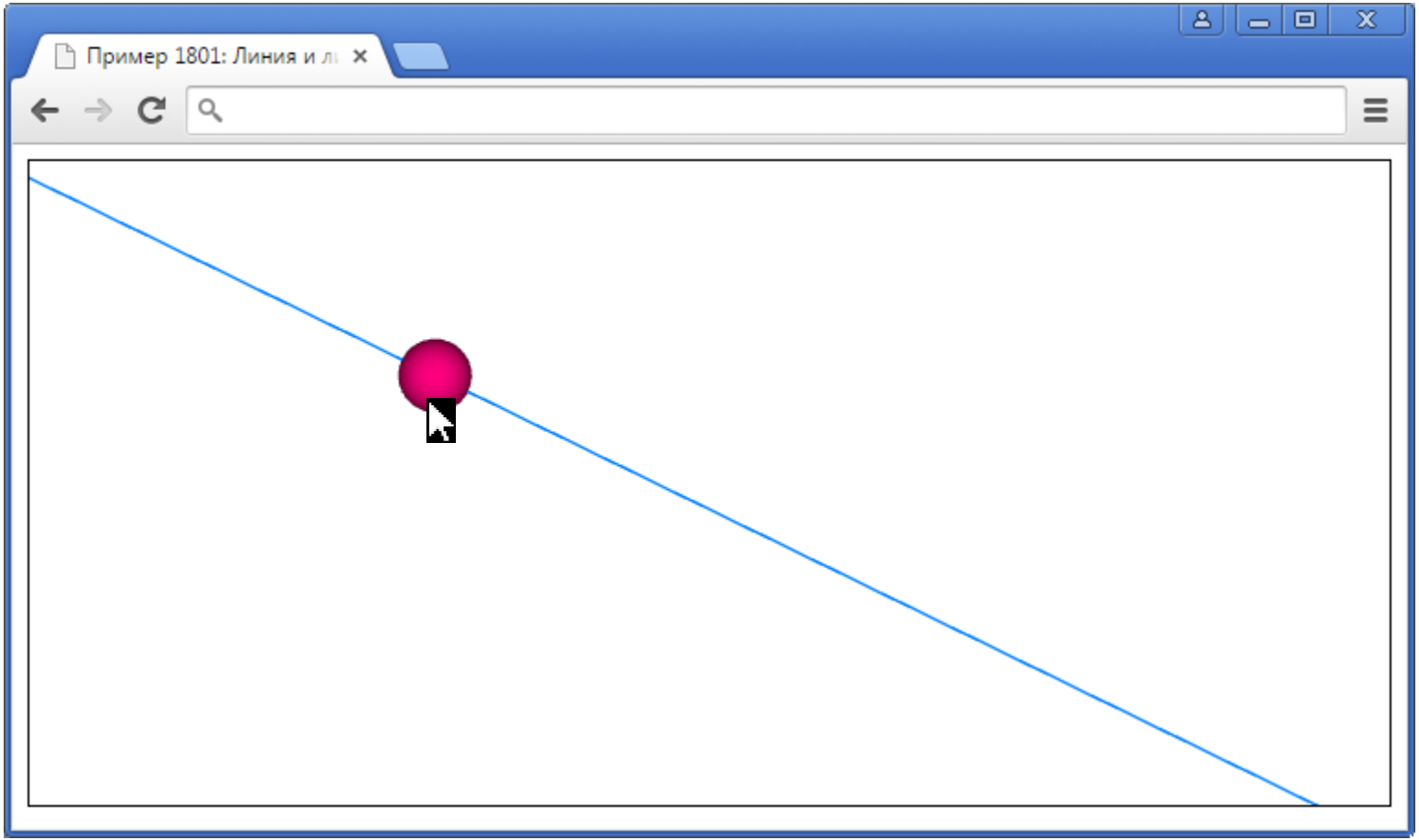
- Повече и по-сложни сметки
- Влаченето изглежда по-естествено

Влачене с линейна комбинация

Влачене по линия

- Обект е с център върху линията през двете точки p1 и p2
- Уловеното движение променя коефициента на линейната комбинация k за намиране на новия център на обекта

```
if (obj)
{
   k -= (event.clientX-x)/500;
   s.center[0] = p1[0]*k+(1-k)*p2[0];
   s.center[1] = p1[1]*k+(1-k)*p2[1];
}
x = event.clientX;
```

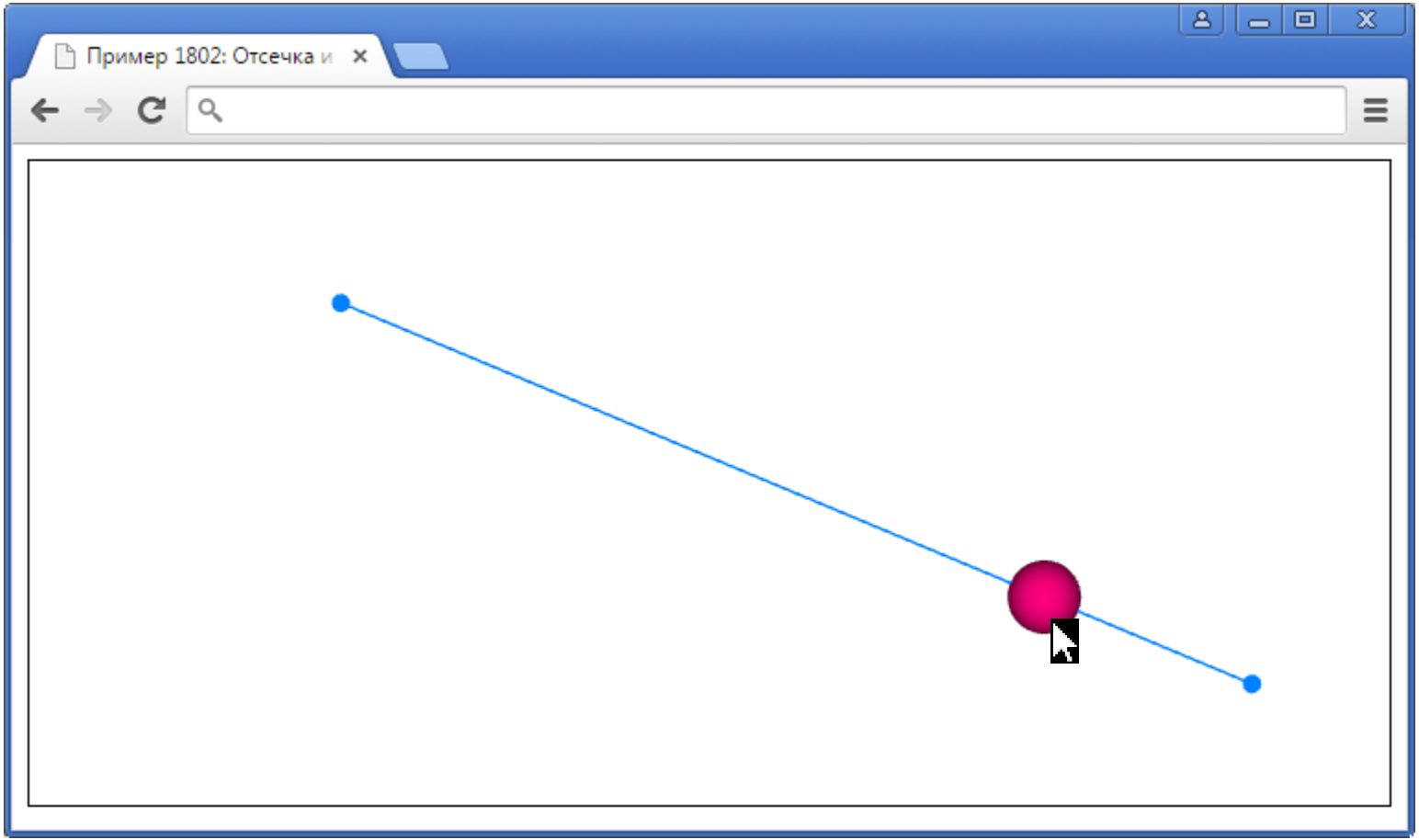


ПРОБА

Влачене по отсечка

- Аналогично на движението по линия
- Допълнително ограничение $k \in [0,1]$
- Връзката между мишката и обекта може да се разкъса при движение на мишката много извън отсечката

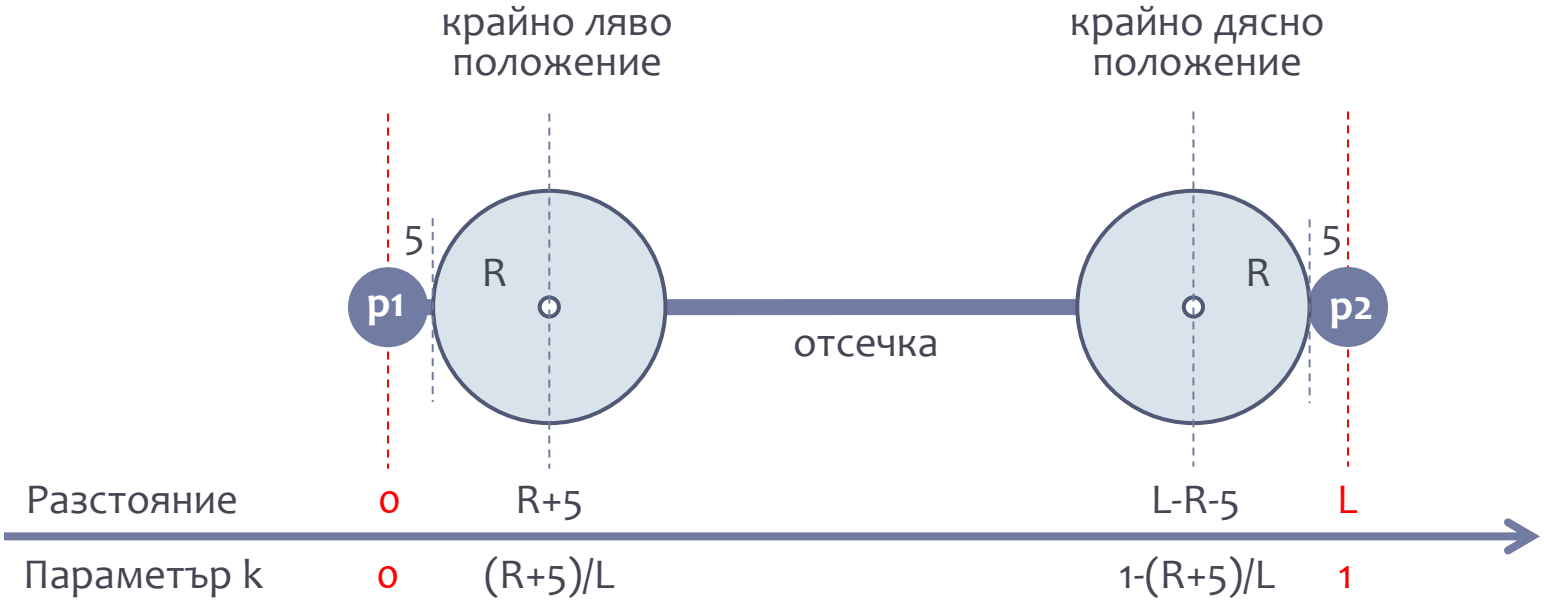
```
if (obj)
{
   k -= (event.clientX-x)/500;
   if (k<0) k=0;
   if (k>1) k=1;
   s.center[0] = p1[0]*k+(1-k)*p2[0];
   s.center[1] = p1[1]*k+(1-k)*p2[1];
}
```



ПРОБА

Усложнение

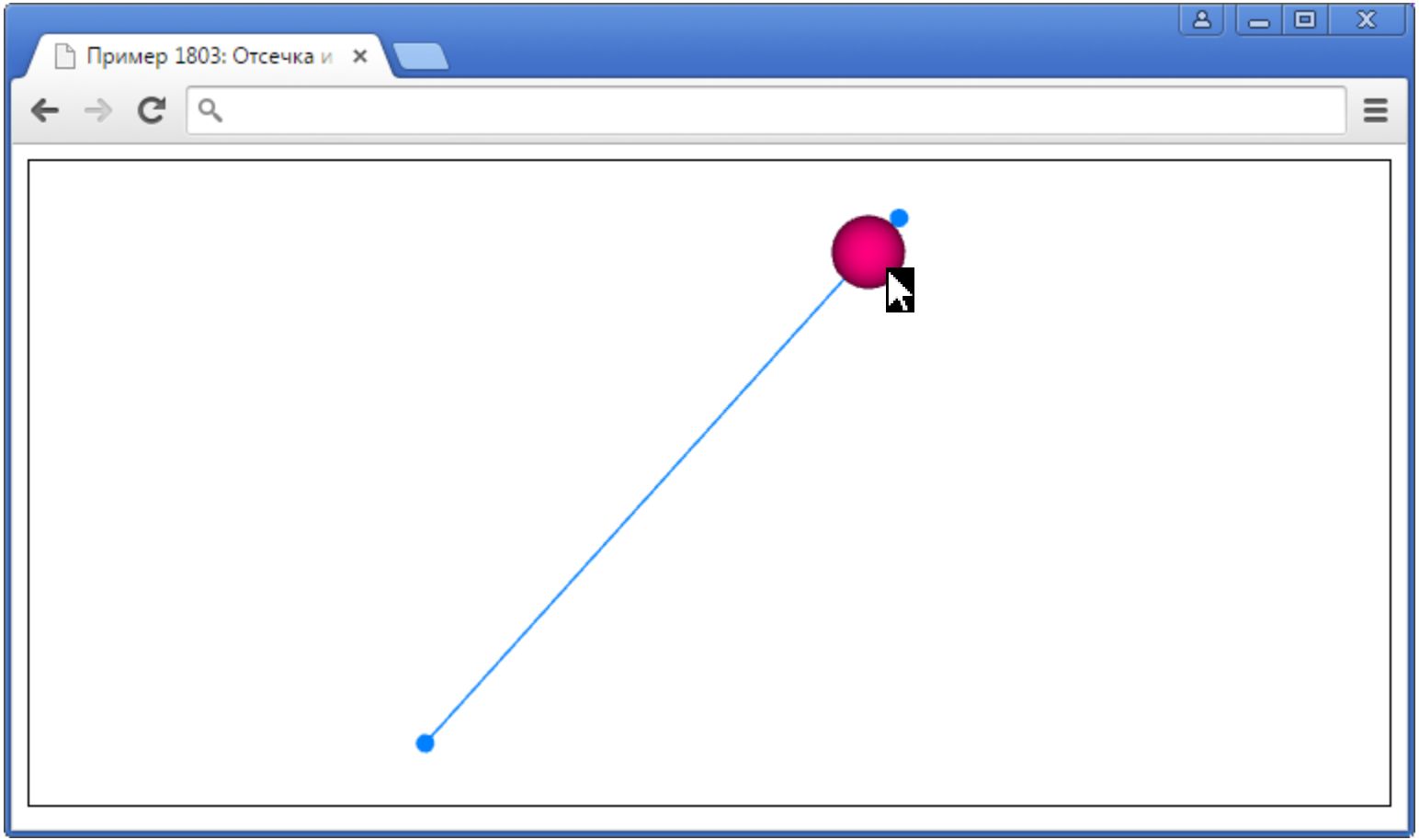
- Подвижната сфера да се движи само до ограничителите на отсечката, без да ги застъпва



Реализация

- Допустимият диапазон за k е симетрично стеснен отгоре и отдолу с kLimit, т.е. k∈[0+kLimit,1-kLimit]
- Стойността на kLimit е сумата от двата радиуса (на ограничителя и на обекта) спрямо дължината на отсечката

```
kLimit = (s.radius+5)/
        Math.sqrt( (p1[0]-p2[0])*(p1[0]-p2[0]) +
                   (p1[1]-p2[1])*(p1[1]-p2[1]) );
...
if (k<kLimit) k=kLimit;
if (k>1-kLimit) k=1-kLimit;
```



ПРОБА

Най-близка точка

Задача за най-близката точка

- Дадена е права, дефинирана от две точки A и B
- Дадена е произволна точка C
- Да се намери точка D от правата, която е най-близо до произволната точка

Решение

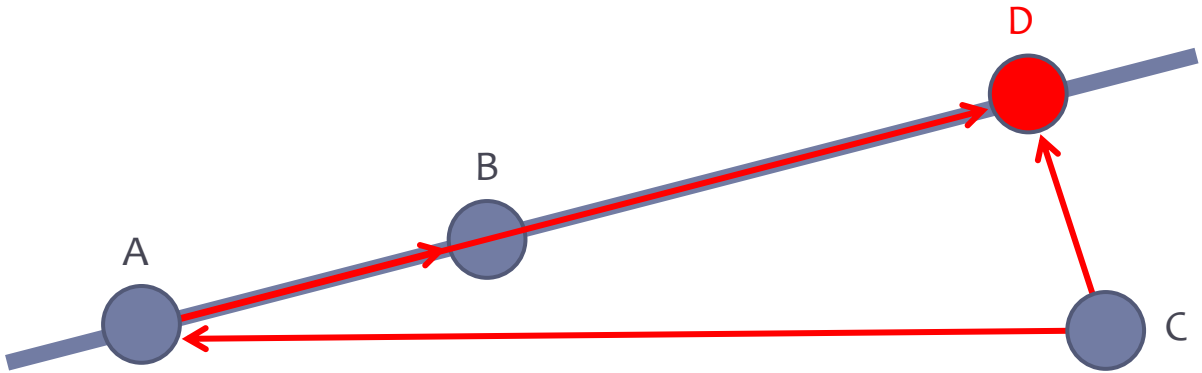
- Работим с вектори:

$$\overrightarrow{AD} = \overrightarrow{AB}.k$$

$$\overrightarrow{CD} = \overrightarrow{CA} + \overrightarrow{AD} = \overrightarrow{CA} + \overrightarrow{AB}.k$$

$$\overrightarrow{CD} \perp \overrightarrow{AD} \Rightarrow \overrightarrow{CD} \perp \overrightarrow{AB} \Rightarrow \overrightarrow{CD}.\overrightarrow{AB} = 0 \Rightarrow \left(\overrightarrow{CA} + \overrightarrow{AB}.k\right).\overrightarrow{AB} = 0$$

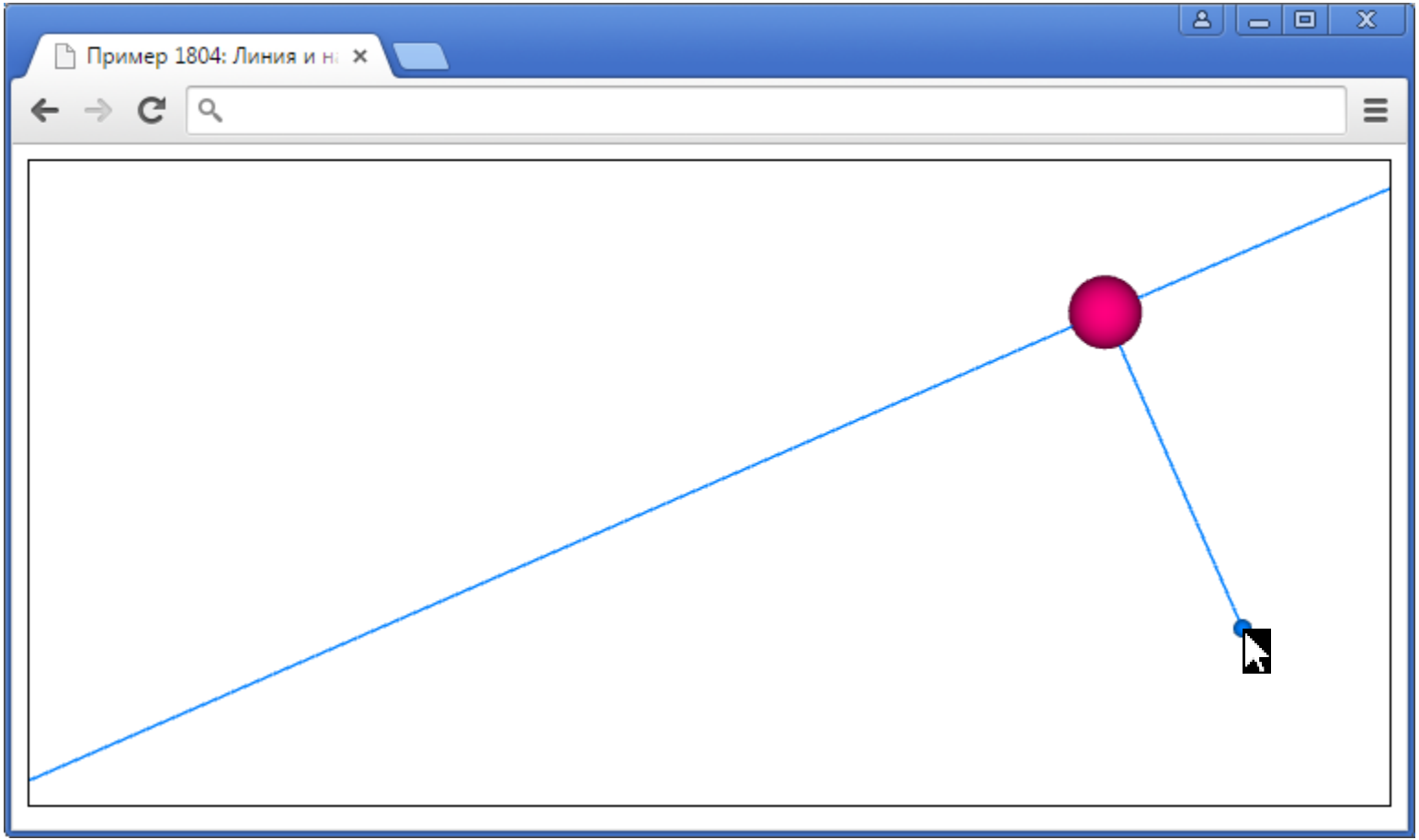
- Решава се спрямо k и от $\overrightarrow{AD} = \overrightarrow{AB}.k$ се намира D



Реализация

- Променяме точки A и B, за да имаме подвижна права
- Изчисляваме k от координатите на A, B и C (за по-кратък израз използваме и вектора $\overrightarrow{AB}$)
- Намираме точка D по правата като $D = A+\overrightarrow{AB}.k$

```
A = [...];
B = [...];

AB = vectorPoints(B,A);
k =((C.center[0]-A[0])*AB[0]+(C.center[1]-A[1])*AB[1])
                              /(AB[0]*AB[0]+AB[1]*AB[1]);
D.center[0] = A[0]+k*AB[0];
D.center[1] = A[1]+k*AB[1];
```



ПРОБА

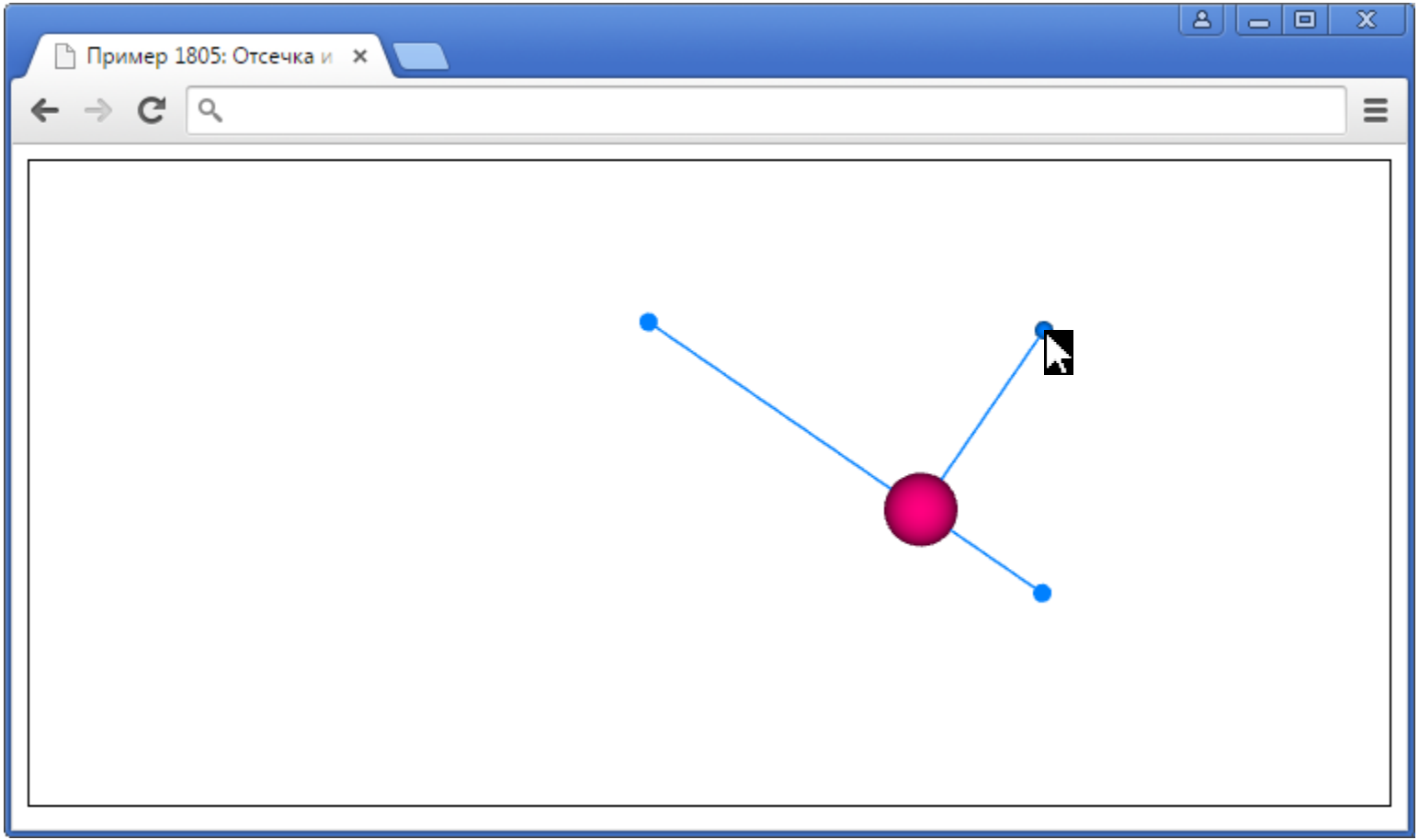
Най-близка точка до отсечка

- Аналогичен алгоритъм и изчисления
- За да бъде точката D между A и B, трябва $k \in [0,1]$
- Това изискване напомня на линейната комбинация

Случайно ли е?

- Да проверим $A+\overrightarrow{AB}.k = A+(B-A)k = A(1-k)+kB$

```
k = ...;
if (k<0) k=0;
if (k>1) k=1;
```



ПРОБА

Влачене с най-близка точка

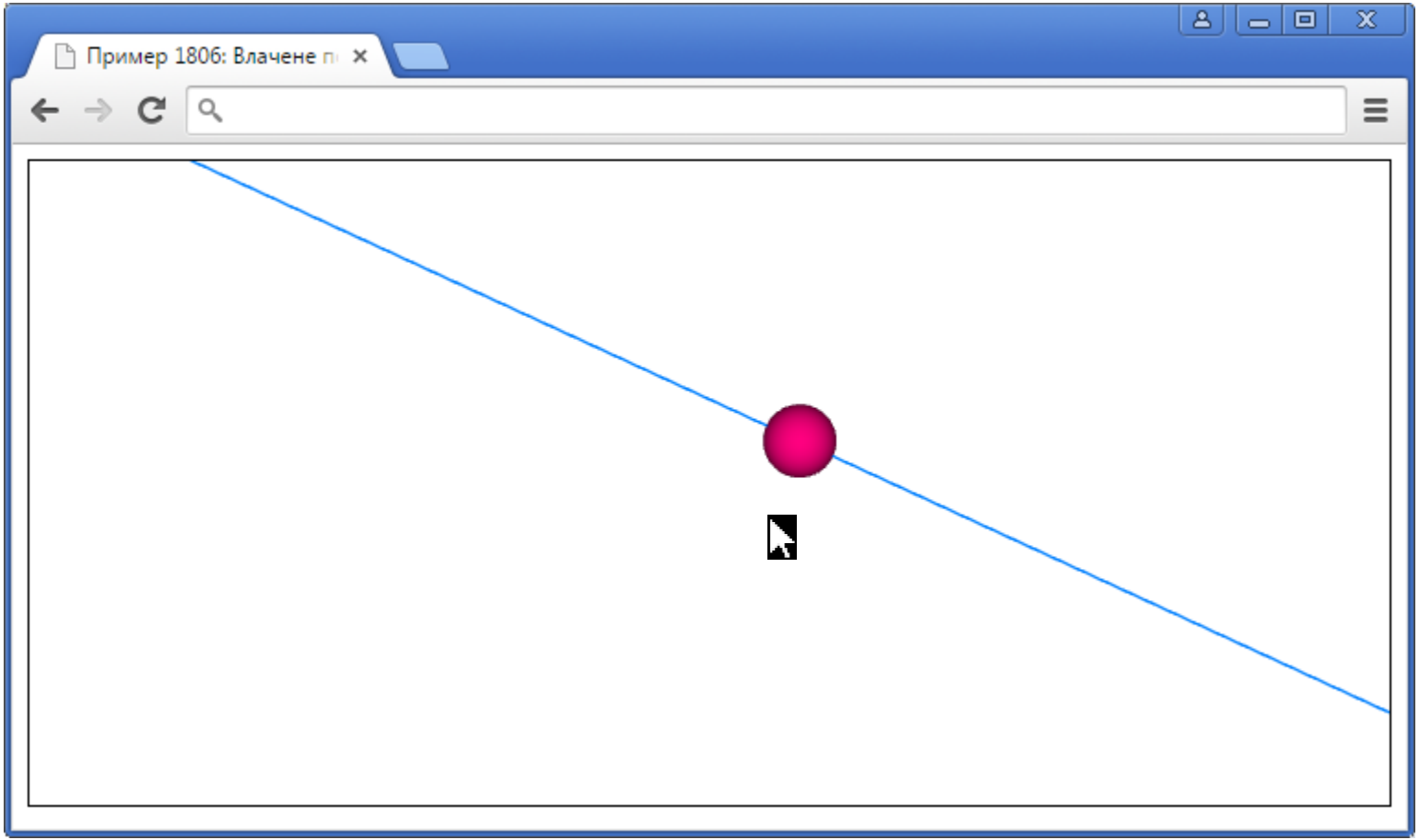
Влачене по линия

- При обработването на събитията само се запомнят последните графичните координати и се следи дали има избран обект

```
function mouseDown(event)
{
   obj = p.objectAtPoint(...);
   mouseMove(event);
}
function mouseUp(event) {obj = null;}
function mouseMove(event) {x = ...; y = -(...);}
```

- Векторът AB и квадратът на дължината му L се изчисляват еднократно, понеже линията в примера не се движи

```
AB = vectorPoints(B,A);
L = AB[0]*AB[0]+AB[1]*AB[1];
...
function animate()
{
   if (obj)
   {
      k = ((x-A[0])*AB[0]+(y-A[1])*AB[1])/L;
      D.center[0] = A[0]+k*AB[0];
      D.center[1] = A[1]+k*AB[1];
   }
}
```

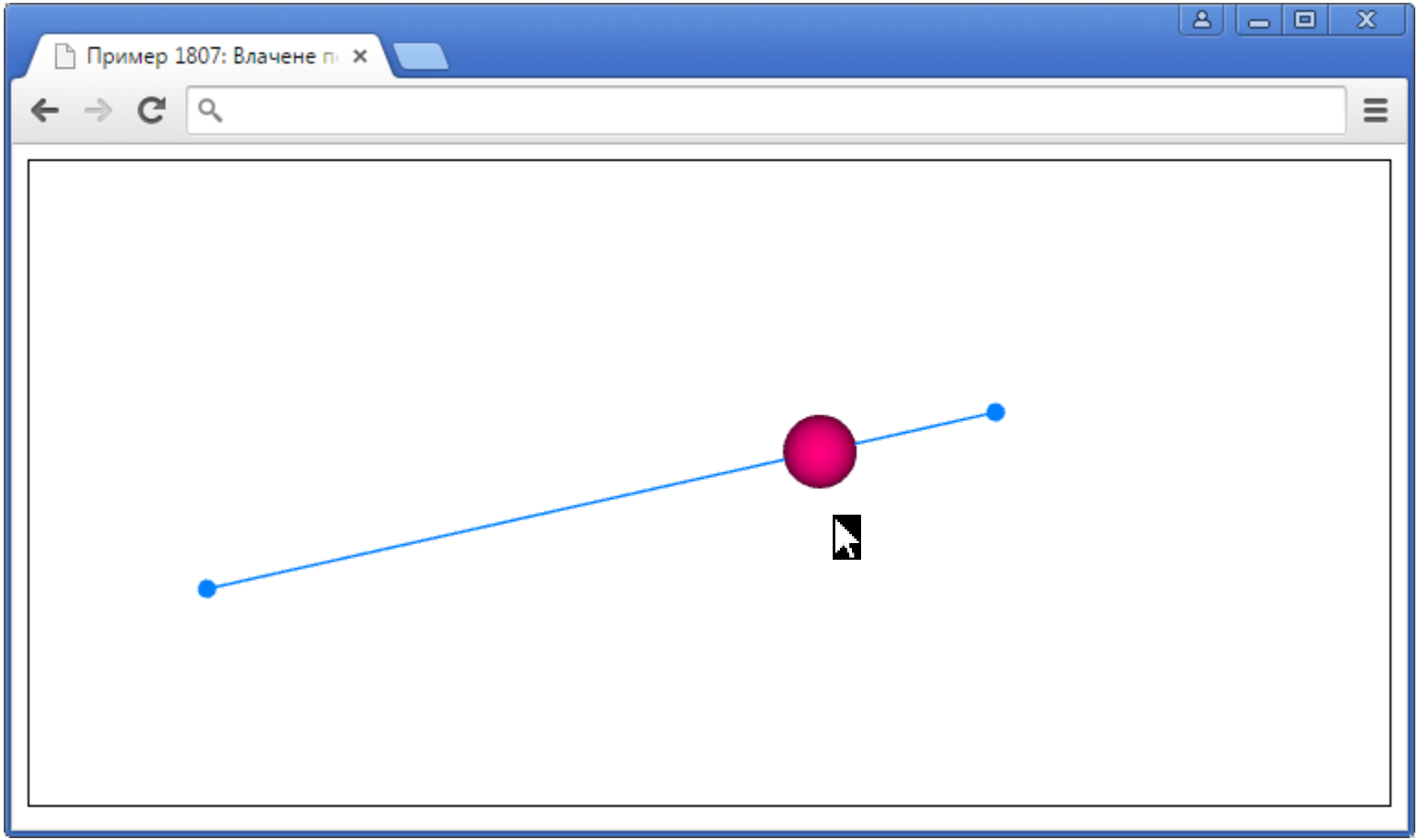


ПРОБА

Влачене по отсечка

- Добавя се само ограничение по k

```
function animate()
{
   if (obj)
   {
      k =((x-A[0])*AB[0]+(y-A[1])*AB[1])/L;
      if (k<0) k=0;
      if (k>1) k=1;

      D.center[0] = A[0]+k*AB[0];
      D.center[1] = A[1]+k*AB[1];
   }
}
```



ПРОБА

Влачене по окръжност

Влачене по окръжност

С ъгъл

- С мишката контролираме параметър
- Използваме го като ъгъл на полярни координати
- Може да се използва при влачене по дъга

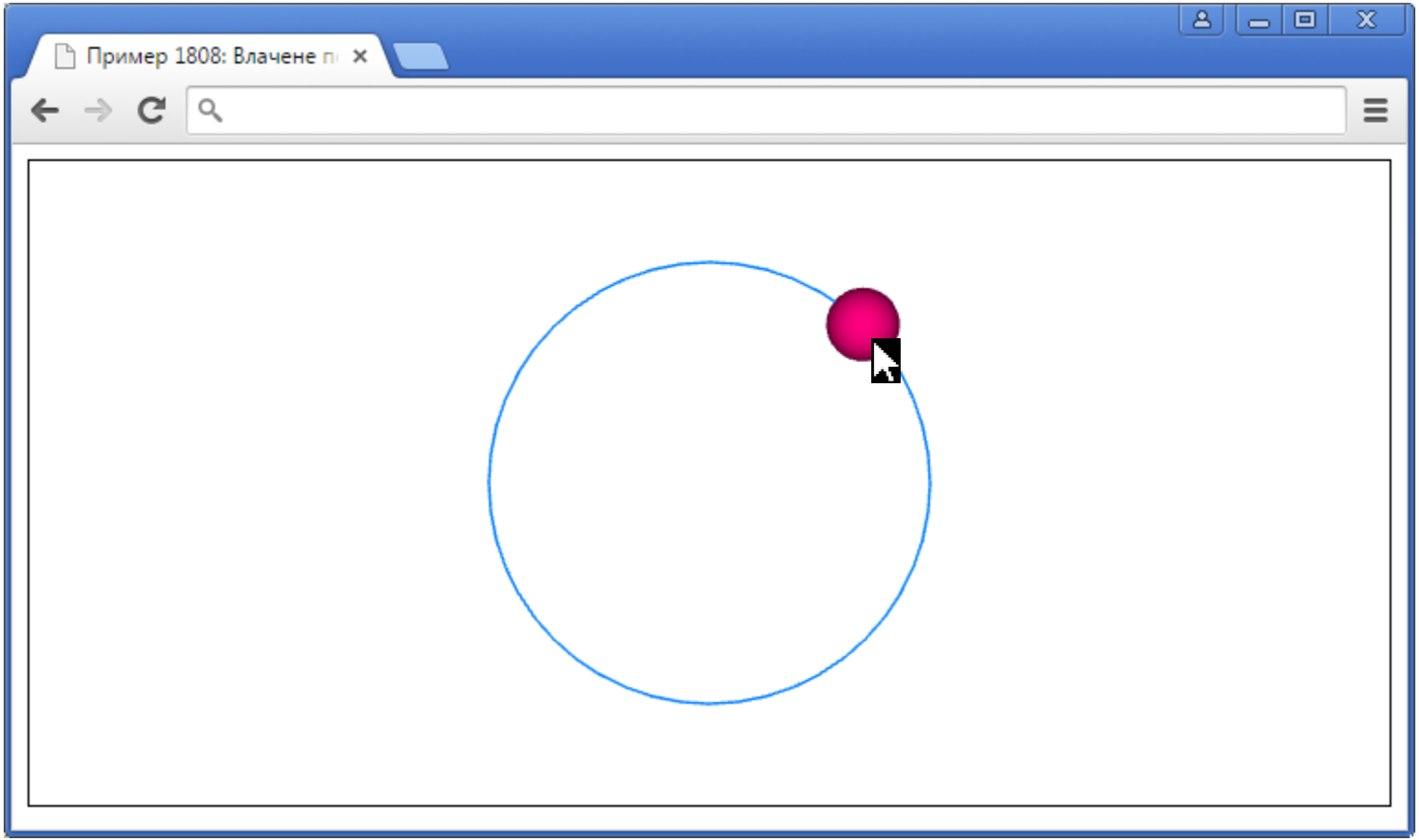
С най-близка точка

- Проектираме точката върху окръжността
- Малко по-трудно е при дъга

Влачене с ъгъл

- Обектът си помни в локалното свойство alpha на какъв ъгъл по окръжността се намира
- Отместването се дели на 100 – т.е. 100 пиксела = 1 радиан

```
function mouseMove(event)
{
   if (obj)
   {
      obj.alpha -= (event.clientX-x)/100;
      obj.center = [120*Math.cos(obj.alpha),
                    120*Math.sin(obj.alpha),0];
   }
   x = event.clientX;
}
```

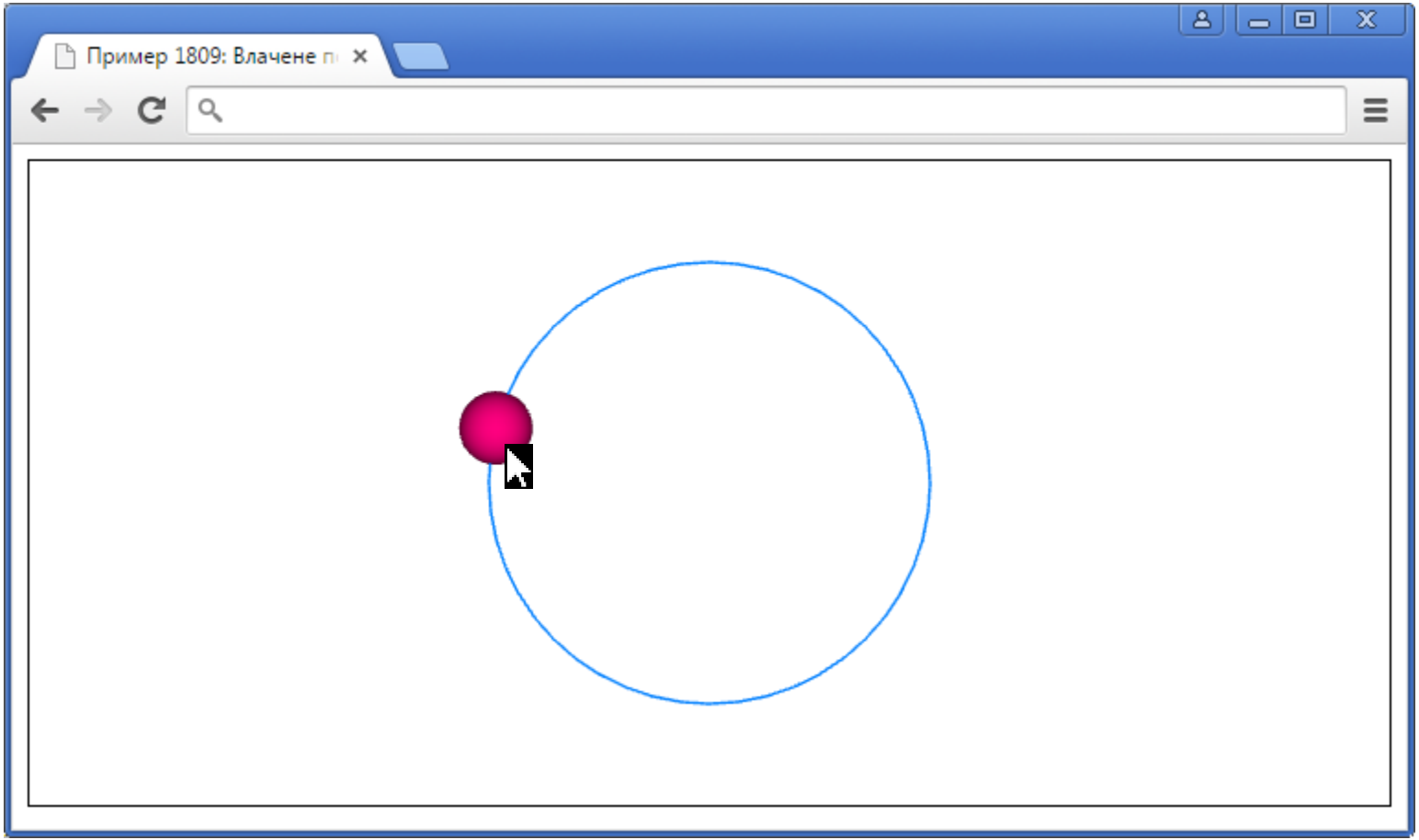


ПРОБА

Влачене с най-близка точка

- Позицията на мишката я мащабираме така, че да стане на разстояние колкото е радиуса на окръжността 120
- Тогава тя е вече позиция и на влачения обект

```
function mouseMove(event)
{  if (obj)
   {
      var x = ...;
      var y = -(...);
      var d = Math.sqrt(x*x+y*y);
      obj.center[0] = 120*x/d;
      obj.center[1] = 120*y/d;
   }
}
```



ПРОБА

Сравнение на влачене по окръжност

Чрез параметър-ъгъл	Чрез най-близка точка
Загуба на интуитивност в някои части от влаченето	Интуитивност във всеки момент от влаченето
Лесна адаптация за влачене по дъга	Трудно приложение при влачене по дъга
Проекцията и гледната точка не са значими	Подходящо при ортографска проекция и отвесна гледна точка
Няма особени точки – влаченето е еднакво плавно	При влачене около центъра на окръжността обектът прави резки движения

Други влачения

Влачене на посока

Влачене на посока

- Дадени са n циферблата с по една стрелка всеки
- Пръснати на случайни места
- Със случайни размери

Цел

- Интерактивно да се избере циферблат и да се върти стрелката му

Реализация на циферблатите

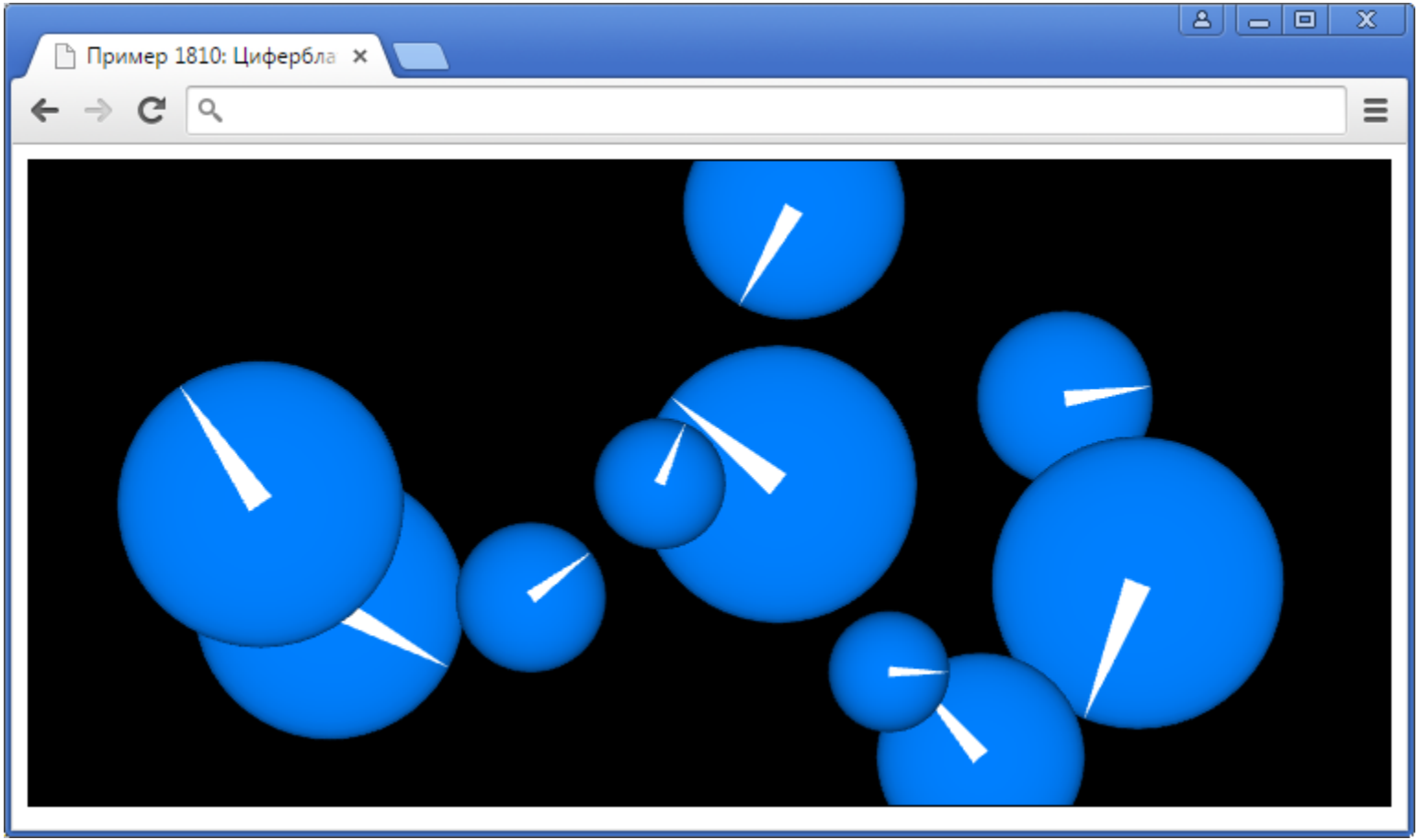
- Сплескани сфери на случайни места (по Z радиусът е 20)
- На 20 единици над всяка сфера има интерактивен конус с височина колкото е радиуса на сферата
- Конусът не е изправен към Z, а е ориентиран в посока Y

```
var c = [random(-300,300),random(-150,150),100*i];
var r = random(30,80);
spheroid(c,[r,r,20]).custom({color:[0,0.5,1]});
cone([c[0],c[1],c[2]+20],r/10,r).custom({
   focus:[0,1,0],
   light:false,
   color:[1,1,1],
   interactive:true});
```

Интерактивност

- Векторът focus на „влачения“ обект се насочва към позицията (x,y) на мишката
- Това завърта конуса-стрелка в посока на мишката

```
function mouseMove(event)
{
   if (obj)
   {
      var x =...;
      var y = -(...);
      obj.focus = [x-obj.center[0],y-obj.center[1],0];
   }
}
```



ПРОБА

Влачене на височина

Пример

- Правилна пирамида
- С хоризонтално влачене я въртим (т.е. влачим неин връх при основата по окръжност)
- С вертикално влачене ѝ сменяме височината (т.е. влачим върха по вертикална отсечка)

Реализация

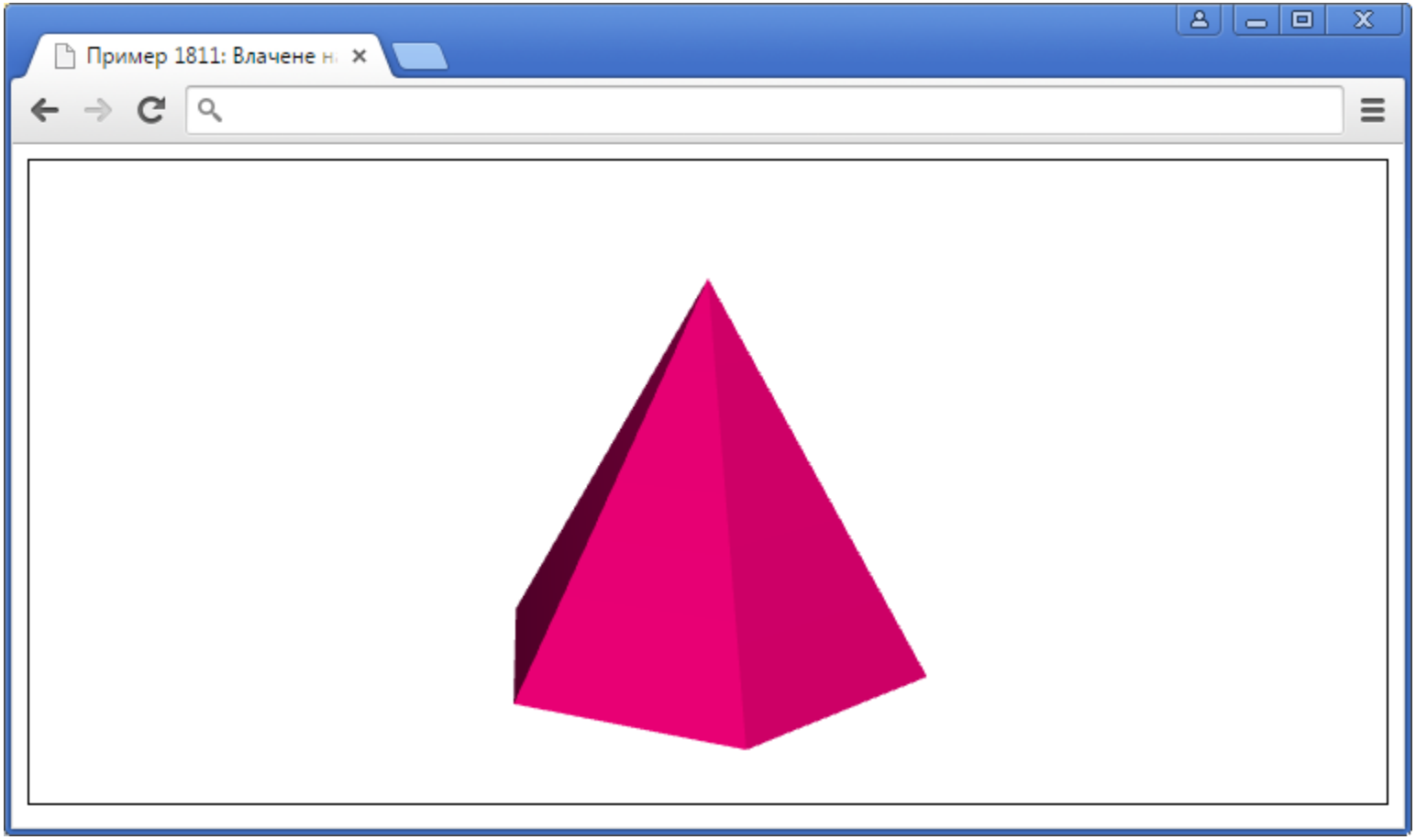
- В променливите dx и dy е отместването на мишката
- Чрез тях се пресмята отместването на спина spin и височината height

```
if (obj)
{
   var dx = event.clientX-x;
   var dy = event.clientY-y;

   obj.spin -= dx/100;
   obj.height -= dy/5;
}
```

Ограничения във влаченията

- Поради неидеалното движение на мишката, искаме да не става смесване на хоризонтално и вертикално движение
- Ползваме само движението с по-голямо отместване
- Височината е ограничена отдолу до 5 (от физически съображения) и отгоре до 40 (от естетически)

```
if (Math.abs(dx)>Math.abs(dy))
   obj.spin -= dx/100;
else
   obj.height -= dy/5;

if (obj.height<5) obj.height=5;
if (obj.height>40) obj.height=40;
```



ПРОБА

Миниигра

Летящи кръстачки

Правила на играта

- Тримерни кръстачки летят в кръг и се въртят
- Всички са големи и червени
- При кликване върху някоя, тя става бяла и плавно се свива

Цел

- Да се кликне върху всички кръстачки за най-малко време

Реализация на кръстачка

- Три взаимно перпендикулярни правилни паралелепипеда
- Те са организирани в група, а групите са елементи на масив
- Групата е интерактивна и оцветена в червено
- Групата е с двойно увеличен размер

```
cross[i] = group( [ cuboid([0,0,0],[10,2,2]),
                    cuboid([0,0,0],[2,10,2]),
                    cuboid([0,0,0],[2,2,10])])
   .custom({
        interactive: true,
        color: [1,0,0.2],
        sizes: [2,2,2]});
```

Допълнителни настройки

- Свойства offset и speed за разбъркване на обектите
- Свойства shrink за това дали обект е в процес на свиване
- С метода mergeColor се прави цялата група едноцветна
- С метода merge се прави цялата група като един обект от гледна точка на objectAtPoint – така ще се избира цяла група, а не отделни паралелепипеди

```
cross[i] = group(...).custom({...,
        offset: random(0,10*Math.PI),
        speed: random(1,2),
        shrink: false});
cross[i].merge();
cross[i].mergeColor();
```

Събитие на мишката

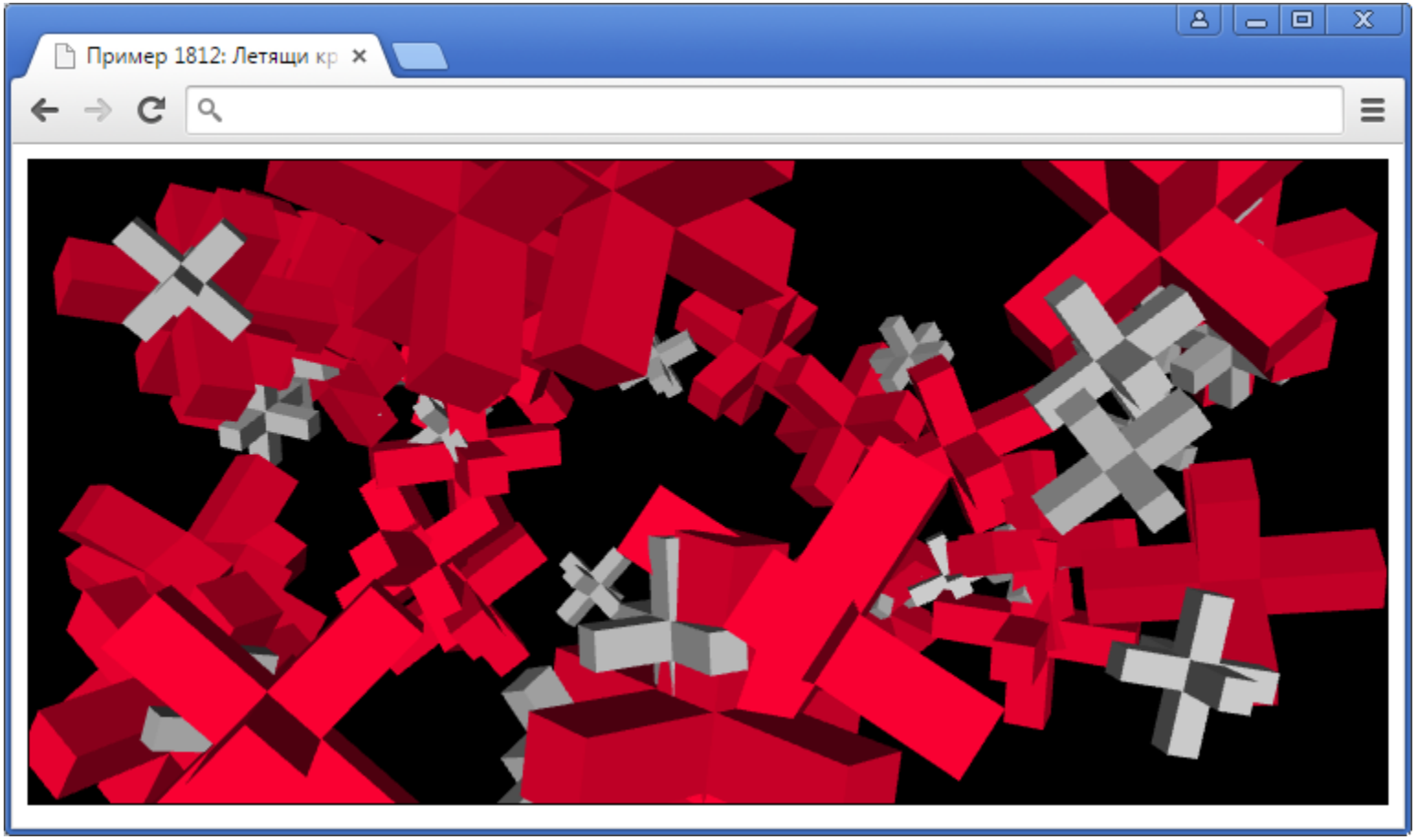
- Използва се само натискане на бутон
- Ако това е станало над обект, правим го бял и позволяваме плавното му свиване

```
function mouseDown(event)
{
   obj = p.objectAtPoint(...);
   if (obj)
   {
      obj.color = [0.8,0.8,0.8];
      obj.shrink = true;
   }
}
```

Главен цикъл

- Всеки обект ползва собствено време t
- От него се изчислява центъра, ориентацията и завъртяността
- Ако е позволено свиване, размерът се смалява с линейна комбинация спрямо текущия размер и 1

```
t = Suica.time/3*cross[i].speed+cross[i].offset;
cross[i].center = [40*cos(t),40*sin(t),15*cos(1.5*t+i)];
cross[i].focus = [cos(2*t),sin(1.3*t),sin(-1.5*t)];
cross[i].spin = (t-i)*cross[i].speed;
if (cross[i].shrink)
{
   var s = cross[i].sizes[0]*0.9+0.1*1;
   cross[i].sizes = [s,s,s];
}
```



ПРОБА

Обобщение

Влачене с ограничения

Ограничения

- Обективни – например формата на траекторията определя позволените положения при влачене
- Субективни – например от естетически причини някои положения при влачене са недостъпни

Реализация

- Ограничаване на входните параметри на движението
- Ограничаване на изходните координати на движението

Влачене по линия

- Чрез линейна комбинация над образуващите две точки
- Чрез намиране на най-близката точка
- Линията може да не е права – движение по окръжност

Други влачения

- Влачене на некоординатни свойства (ориентация, завъртяност, височина)

ИКТ в НОС

Край

Коментари, въпроси