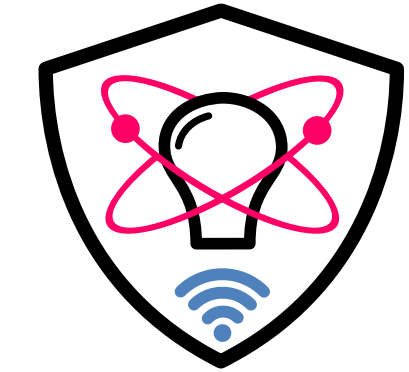


ИКТ в НОС

Анимация

Тема №13

Принципи на анимацията

Анимация

Етимология

- От латински *anima – давам живот*
- Производни думи: аниматор, анималист, аниме, реанимация

Двупосочна измама

- Аниматорът мами зрителя
  Зрителят се оставя да е мамен

Виждане

Биологическа предпоставка

- Човек гледа с очите, но вижда с мозъка
- Очите, нервните пътища и мозъкът имат ограничен капацитет

Картина в мозъчната кора

- Задържа се за около 1/15 от секундата
- При по-малко от 15 образа в секунда – отделни образи
- При повече от 15 образа в секунда – непрекъснато движение

Реализация

Реализация на високо ниво

- Създава отделни кадри
- Показват се последователно

Реализация на ниско ниво

- Промяна на свойствата на обект

  движение – промяна на центъра

  въртене – промяна на ориентацията

  надуване – промяна на размерите

  избледняване – промяна на цвета

Алгоритъм

Идея за генериране на анимация

- стъпка 1: генерира се кадър
- стъпка 2: доставя се кадър
- стъпка 3: към стъпка 1

Видове доставки

- в реално време (екран, стрийминг)
- в нереално време (видео файл)

Псевдокод

- Традиционен вид на анимационен цикъл

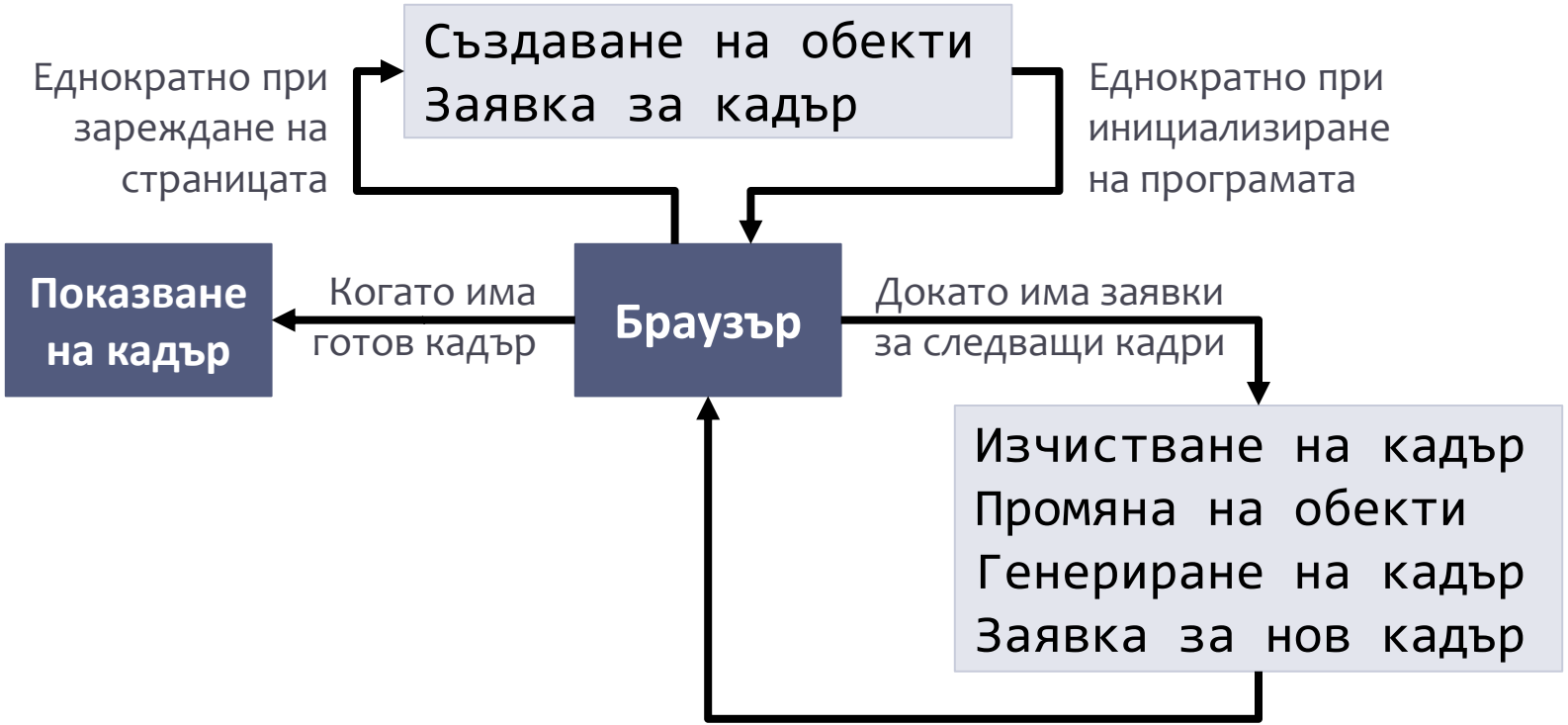
```
създаване на обекти
цикъл за всеки кадър
{
  изчистване на кадър
  промяна на обекти
  показване на кадър
}
```

За предпочитане

- В цикъла само да се променят свойствата на обектите

При работа в браузър

- Програмата заявява, че иска да покаже нов кадър
- Браузърът определя кога е удобно да стане това



Анимационен цикъл в СУИКА

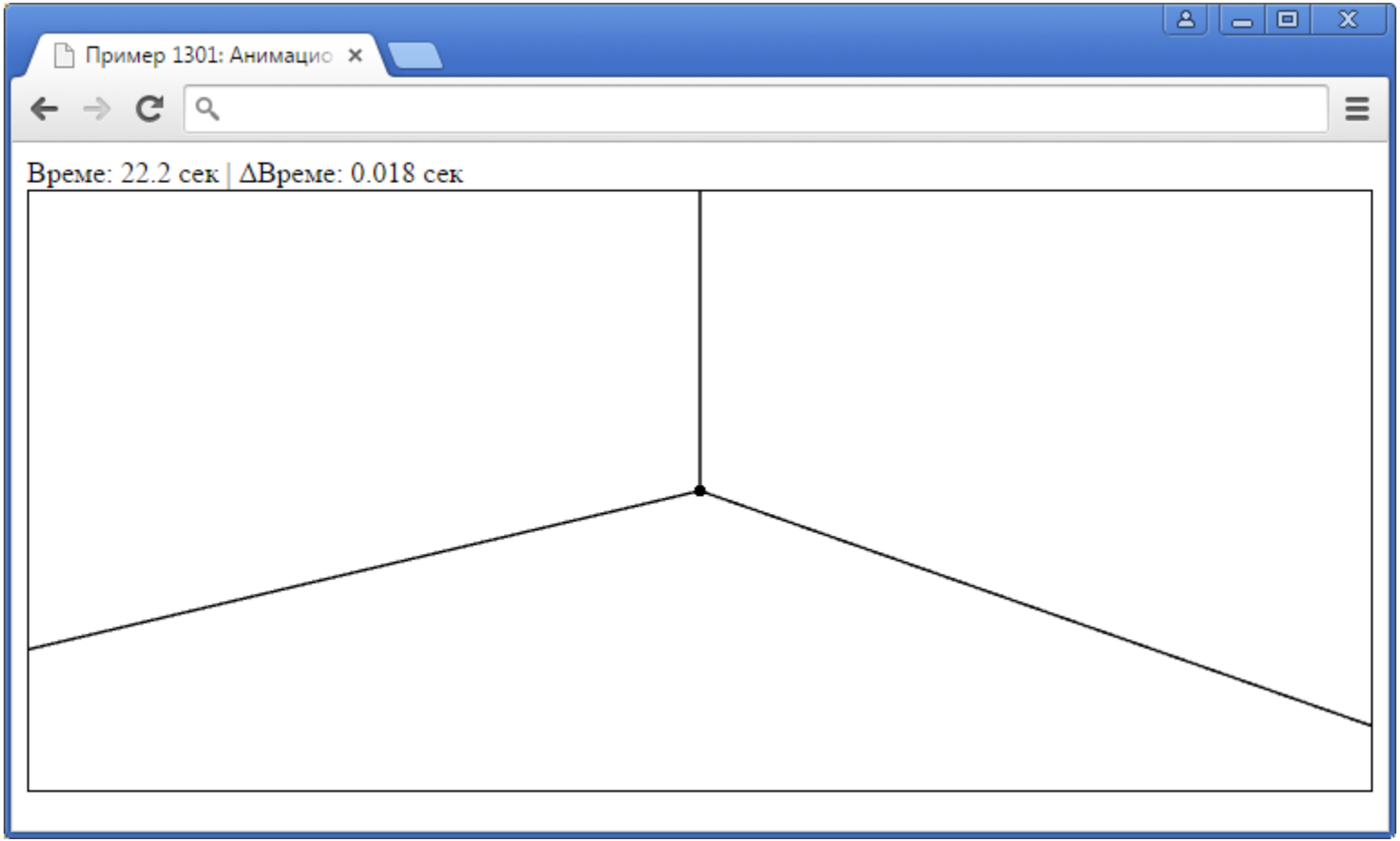
- Запомняме графичното поле в променлива
- Графичните полета имат свойство nextFrame
- Това е променлива, сочеща функция за генериране на кадър
- Извиква се автоматично, когато браузърът разреши

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

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

Пример за цикъл

- Показва изминалото време от началото на програмата – записано в променлива Suica.time
- Показва изминалото време от предходния кадър – записано в променлива Suica.dTime
- Функцията toFixed закръгля дробните числа до съответния брой цифри

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



ПРОБА

Линейно движение

Линейно движение

Различна интерпретация

- Геометрична – движение по права линия
- Параметрична – движение, описвано с един параметър
- Физична – движение, зависещо линейно от времето

Коя интерпретация се ползва

- Зависи от контекста

За начало

- Праволинейно и равномерно движение
- Движение с константна скорост по права линия, описвано с един параметър – времето

Реализации

- Чрез вектор на скоростта
- Чрез точка на целта
- Чрез уравнение на траекторията

Вектор на скоростта

Линейно движение с вектор

- Векторът указва посоката и скоростта
- Ако е фиксиран, движението е праволинейно и равномерно

Математически модел

- P – център на обект
- v – вектор на скорост
- Постъпково движение: P←P+v

Анимация в реално време

- Скоростта е единици разстояние за секунда
- Отчита се изминалото време:
  - Скоростта на анимацията не зависи от скоростта на компютъра
  - При бавен компютър – правят се по-големи междинни стъпки
- Браузърите уеднаквяват скоростта (най-често до 30 или 60 кадъра в секунда)
- Уравнение: $P_{t+\Delta t} = P_t + v.\Delta t$

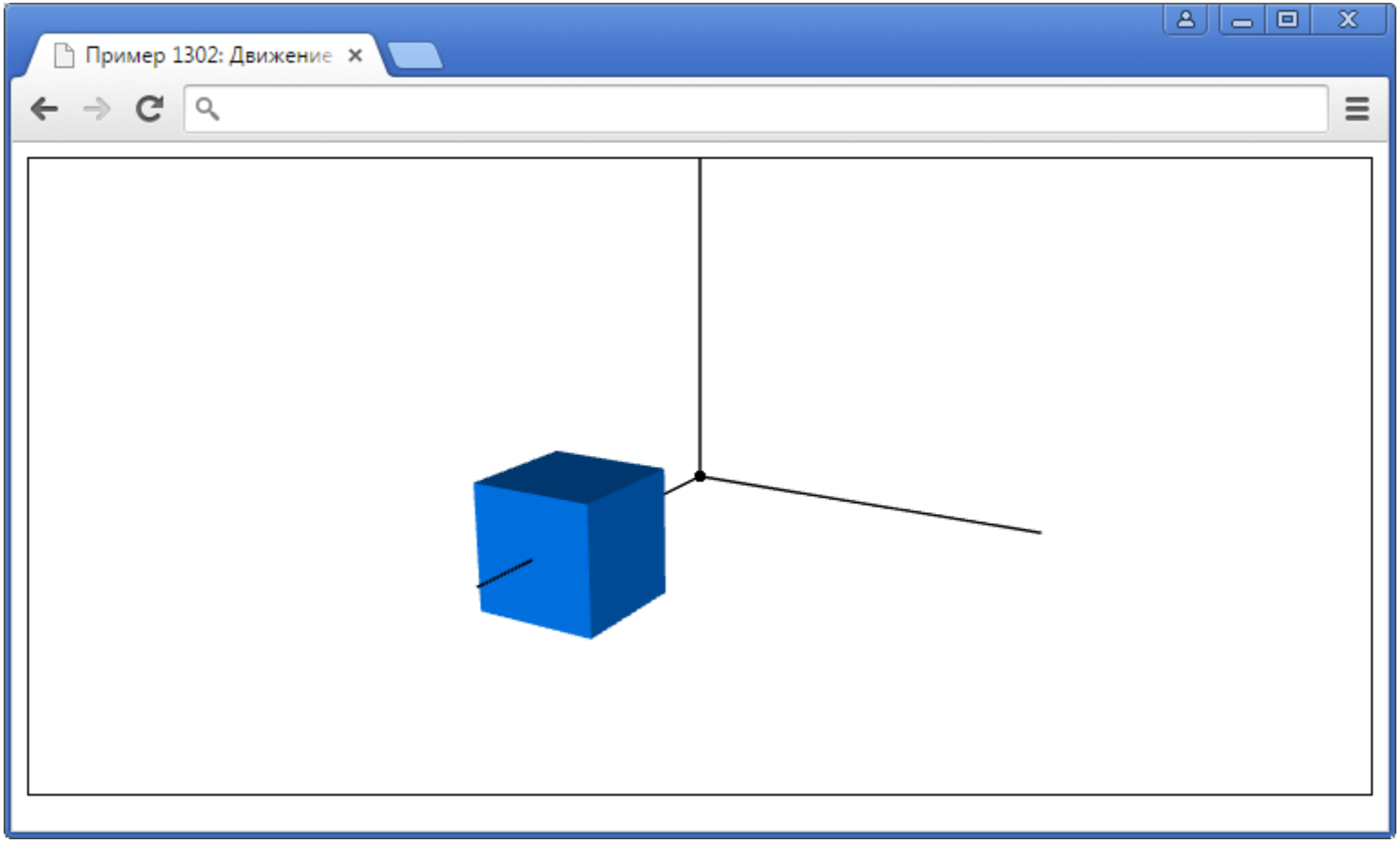
Анимация в нереално време

- Скоростта е единици разстояние за кадър
- Отчитат се изминалите кадри
  - Скоростта на анимация зависи от скоростта на компютъра
  - При бавен компютър – анимацията се забавя
- Браузърите уеднаквяват скоростта (най-често до 30 или 60 кадъра в секунда)
- Уравнение: $P_{f+1} = P_f + v$

Движение по X

- Куб се намира на оста X и се движи в посока +X
- Скоростта е 5 единици за секунда

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

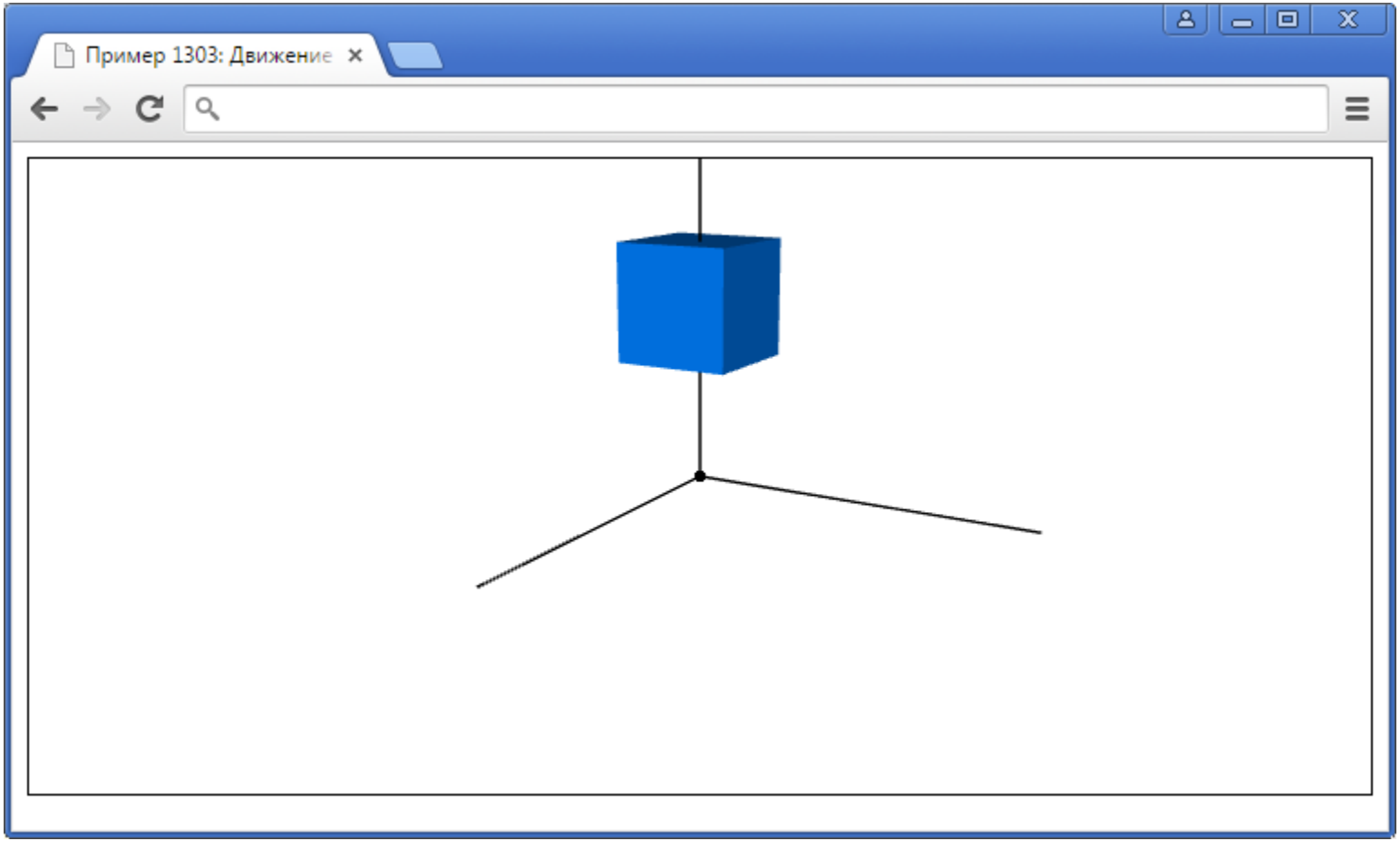


ПРОБА

Движение по Z

- Движението е по-бързо (15 единици в секунда)
- При достигане на определена височина, движението започва отново от (0,0,0)

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

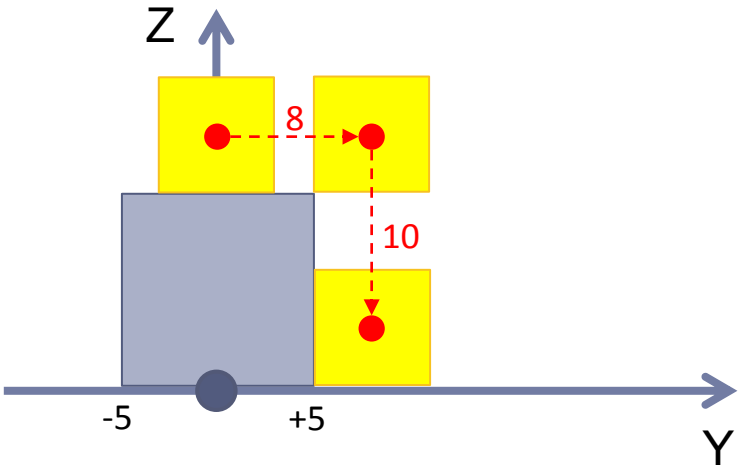


ПРОБА

Последователни движения

Две движения

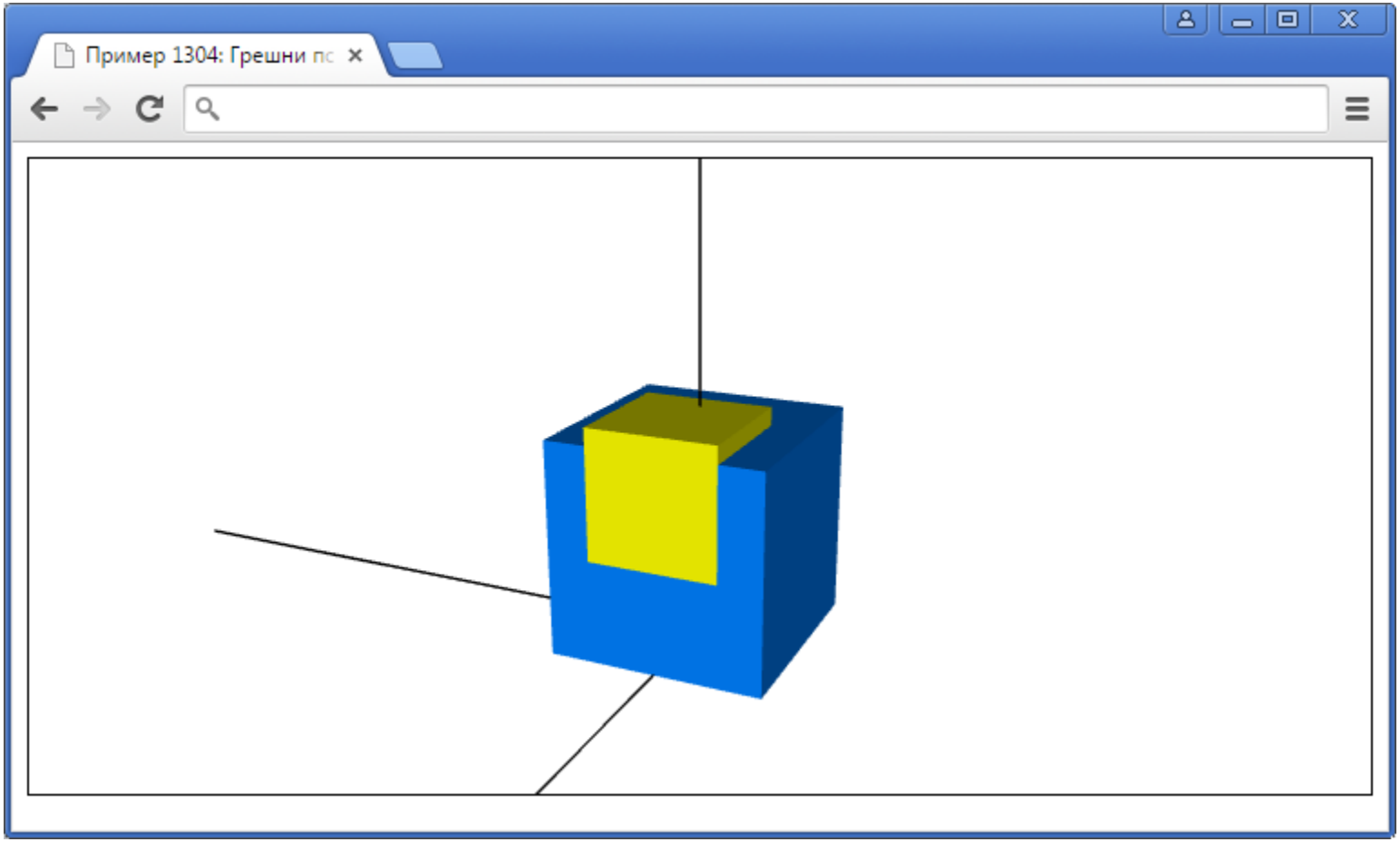
- Синьо кубче 10x10x10, жълто кубче 6x6x6 върху него
- Жълтото кубче се плъзга встрани за 2 сек и пада за 1 сек
- Векторите на скоростите са (0,4,0) и (0,0,-10)



Решение №1

- Двата вектора на движението са правилно използвани
- Добавено е връщане в началото при достигане на някакво ниско ниво
- Въпреки това, движението е грешно

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

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

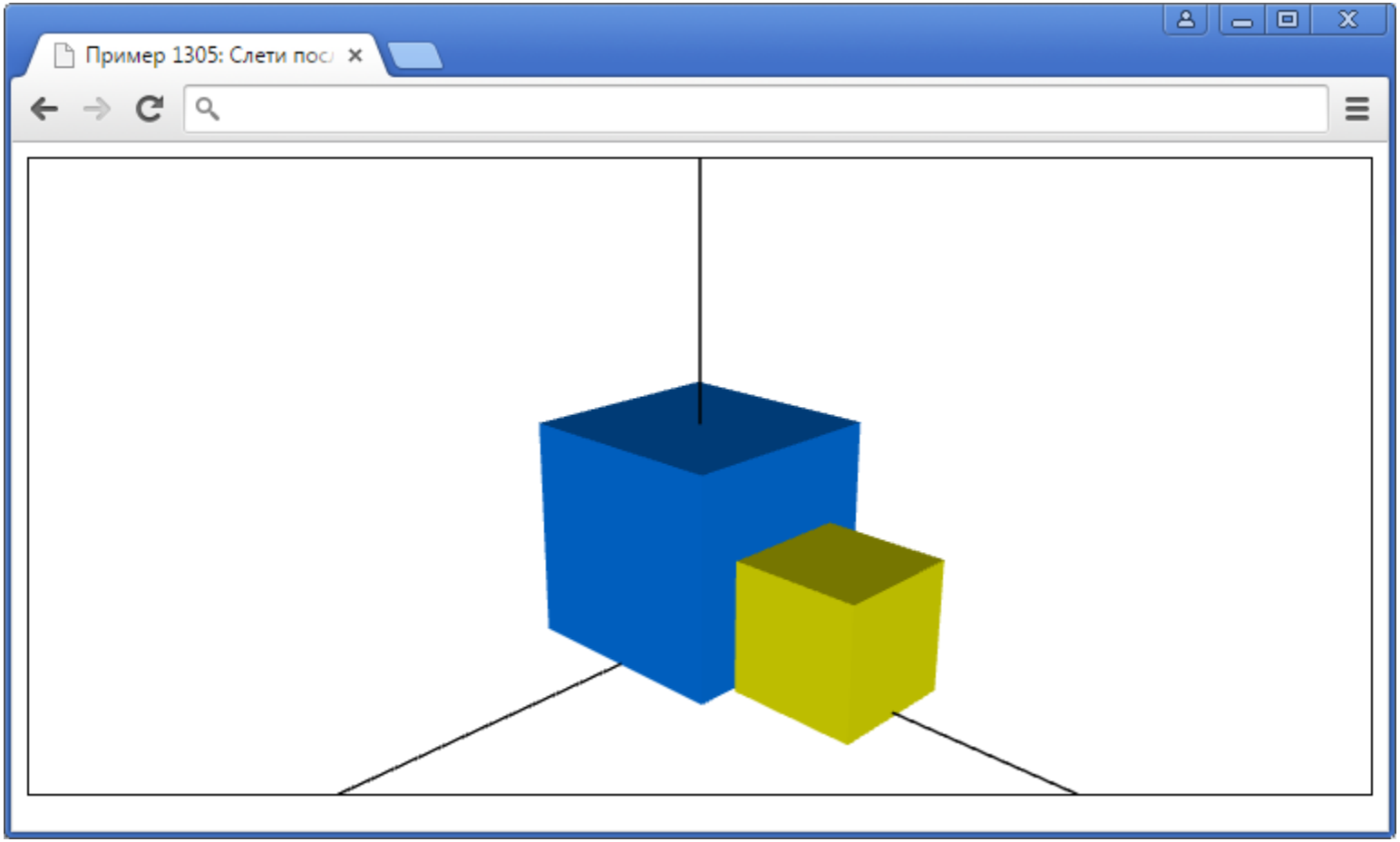


ПРОБА

Решение №2

- Общ цикъл, разделен с условия според координатите
- Плъзгане при y<8, падане при y≥8 и z>0, иначе без движение

```
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;
   }
}
```

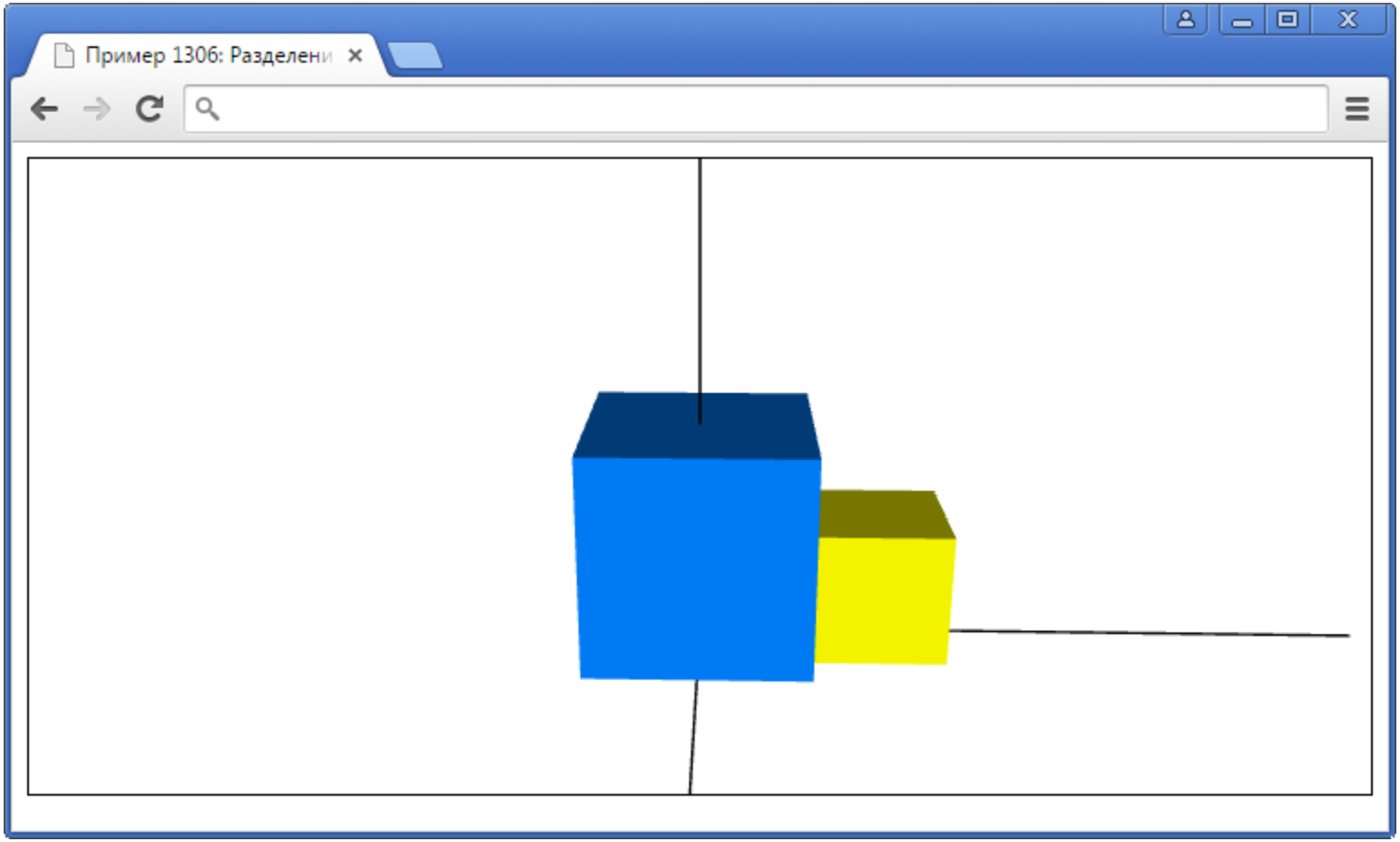


ПРОБА

Решение №3

- Отделни цикли за отделните фази на анимацията
- Започваме само с цикъл с плъзгане slide
- После се включва само цикъл падане fall
- Накрая премахваме цикъл за генериране на кадър

```
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;}
```



ПРОБА

От точка до точка

От точка до точка

Движение от точка до точка

- Най-често срещано движение
- Примитивна форма на движение по траектория

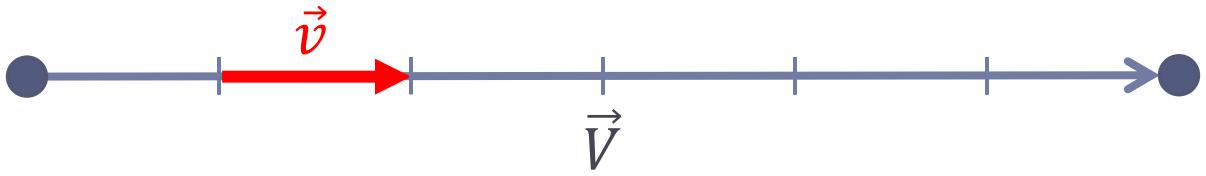
Реализация

- Чрез вектор на скоростта
- Чрез линейна комбинация

Вектор на скоростта

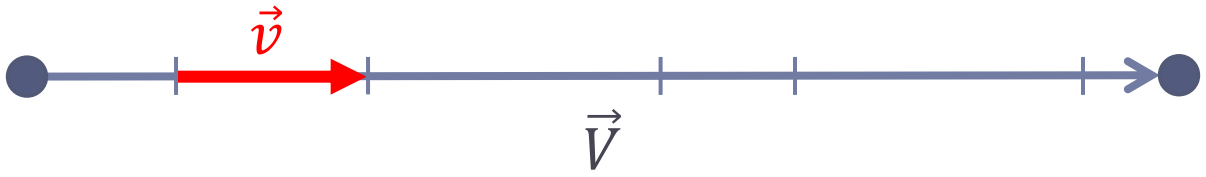
Пресмятане на вектора

- Разглеждаме отсечката като вектор $\vec{V}$
- Определяме желания брой стъпки (кадри) n
- Векторът на скоростта е $\vec{v} = \frac{1}{n}\vec{V}$
- Особености: $\vec{v}$ се пресмята еднократно, но движението не отчита изминалото време, а само брой кадри



Отчитане на времето

- Разглеждаме отсечката като вектор $\vec{V}$
- Определяме желаното време за обхождане T
- Векторът на скоростта е $\vec{v} = \frac{\Delta t}{T}\vec{V}$
- Особености: $\vec{v}$ се пресмята на всеки кадър, понеже Δt се променя непрекъснато



Пример

- Две сфери a и b на случайни места
- Трета сфера c се движи от първата до втората
- Времето за движение е t=3 секунди

Реализация

- Сфера c дублира центъра на a със sameAs

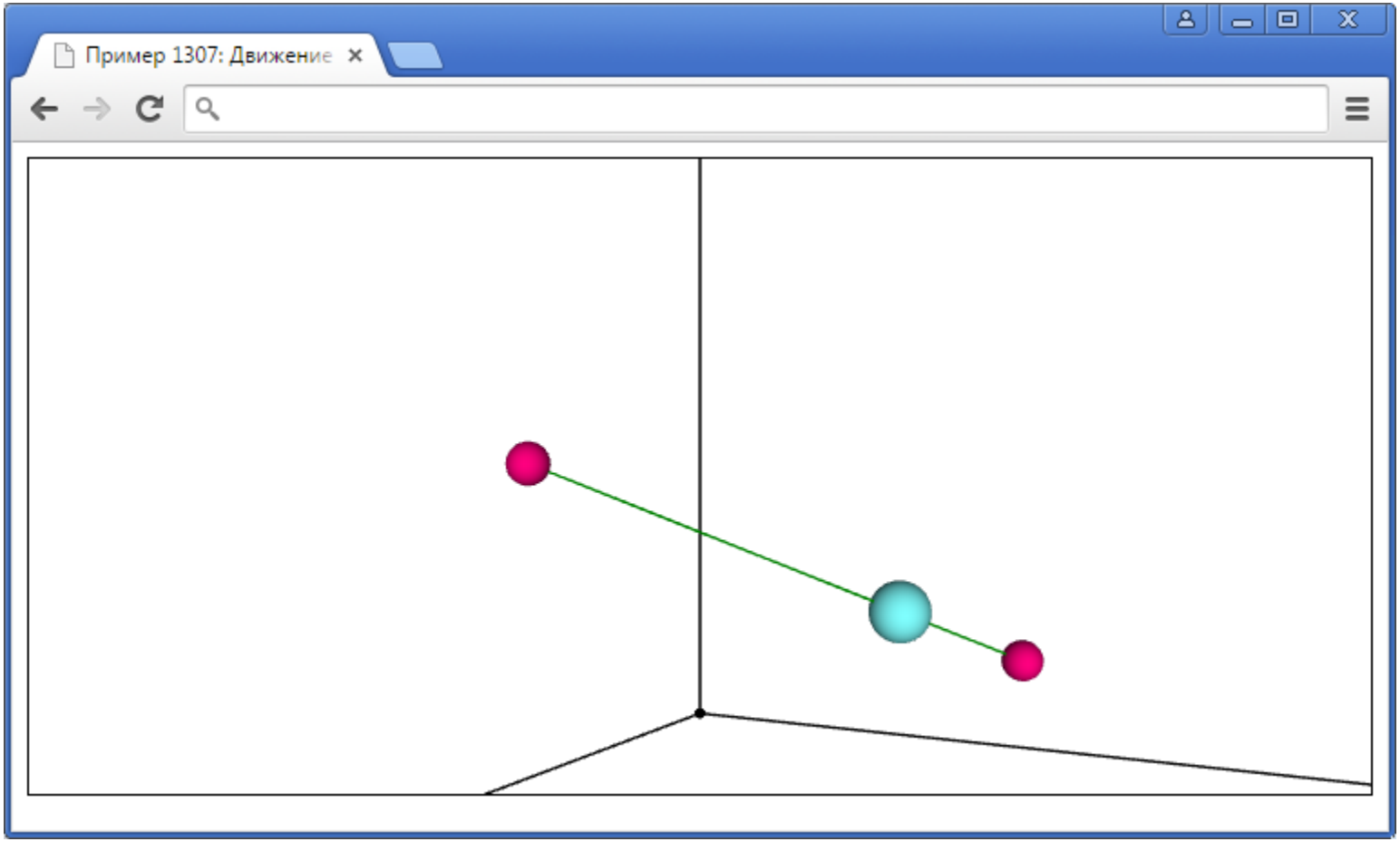
```
a = sphere([random(-15,15),random(-15,15),...);
b = sphere([random(-15,15),random(-15,15),...);
c = sphere(sameAs(a.center),3).custom({...});

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

Основен цикъл

- Докато времето t не е настъпило, правим движение
- Центърът на c се променя покоординатно:

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

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



ПРОБА

Движение по пръстен

- Пръстен от начупена линия
- Обект се движи по линията
- При достигане на чупка прави плавно обръщане

Идея

- Пръстенът ще е множество от отсечки
- Движението се извършва от началото до края на една от отсечките, после по следващата и т.н.
- Ориентацията на обекта ще е с промяна на основната му ос

Реализация на пръстена

- Има n сфери, разположени в кръг, но с отместване по Z
- Между всеки две съседни сфери има отсечка

```
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({...});
}
```

Реализация на движението

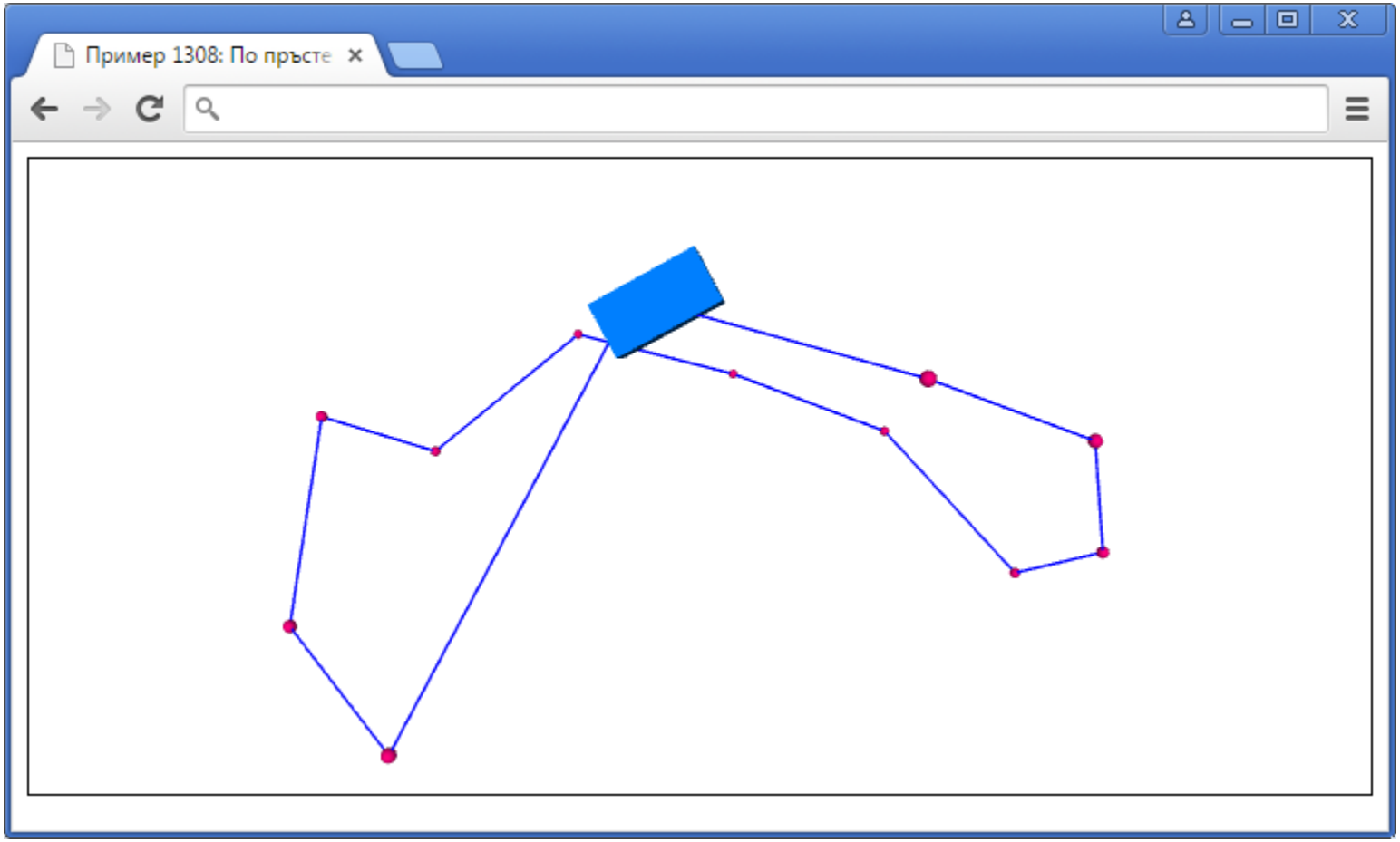
- Броим кадрите във frame
- Една отсечка изминаваме за 50 кадъра
- На всеки 50ти кадър минаваме към следващата отсечка

```
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;
```

Реализация на подвижния обект

- Обектът ще е правоъгълен паралелепипед
- Локалната му ос Z (focus) е по посока на движението
- Завиването става с линейна комбинация – ползват се 80% от текущата посока на обекта и 20% от желаната посока във 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];
}
```



ПРОБА

Линейна комбинация

Движение с линейна комбинация

- Началната и крайната точка
- Параметър $k \in [0,1]$ за координатите
- Междинна точка $A.(1-k)+k.B$ (ако движението е $A \rightarrow B$)

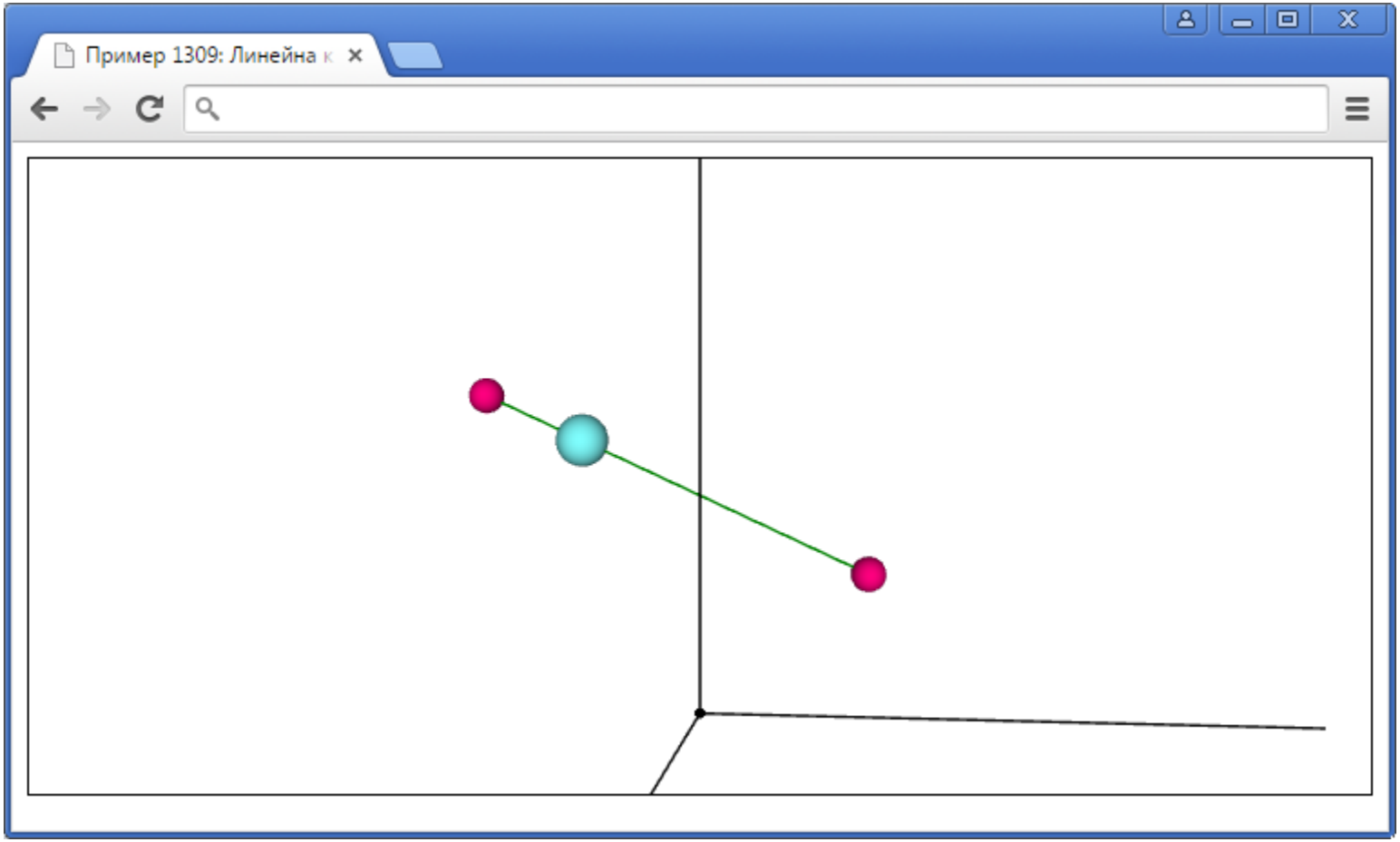
Преимущества

- По-пълен контрол над движението
- Равномерно и неравномерно движение
- Променливи крайни точки

Реализация

- Параметър k е синусоида, трансформирана от [-1,1] до [0,1]
- Координатите на подвижната сфера са линейна комбинация от координатите на двете крайни сфери
- Видим ефект – в краищата си движението е по-бавно

```
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];
}
```



ПРОБА

Пример

Тенис

- Две хилки се движат в две успоредни равнини
- Накланят се леко докато се движат
- Между тях напред-назад лети топче

Идея

- Хилките ще правят хармонични движения
- Топчето ще е линейна комбинация между центровете им

Реализация на хилки

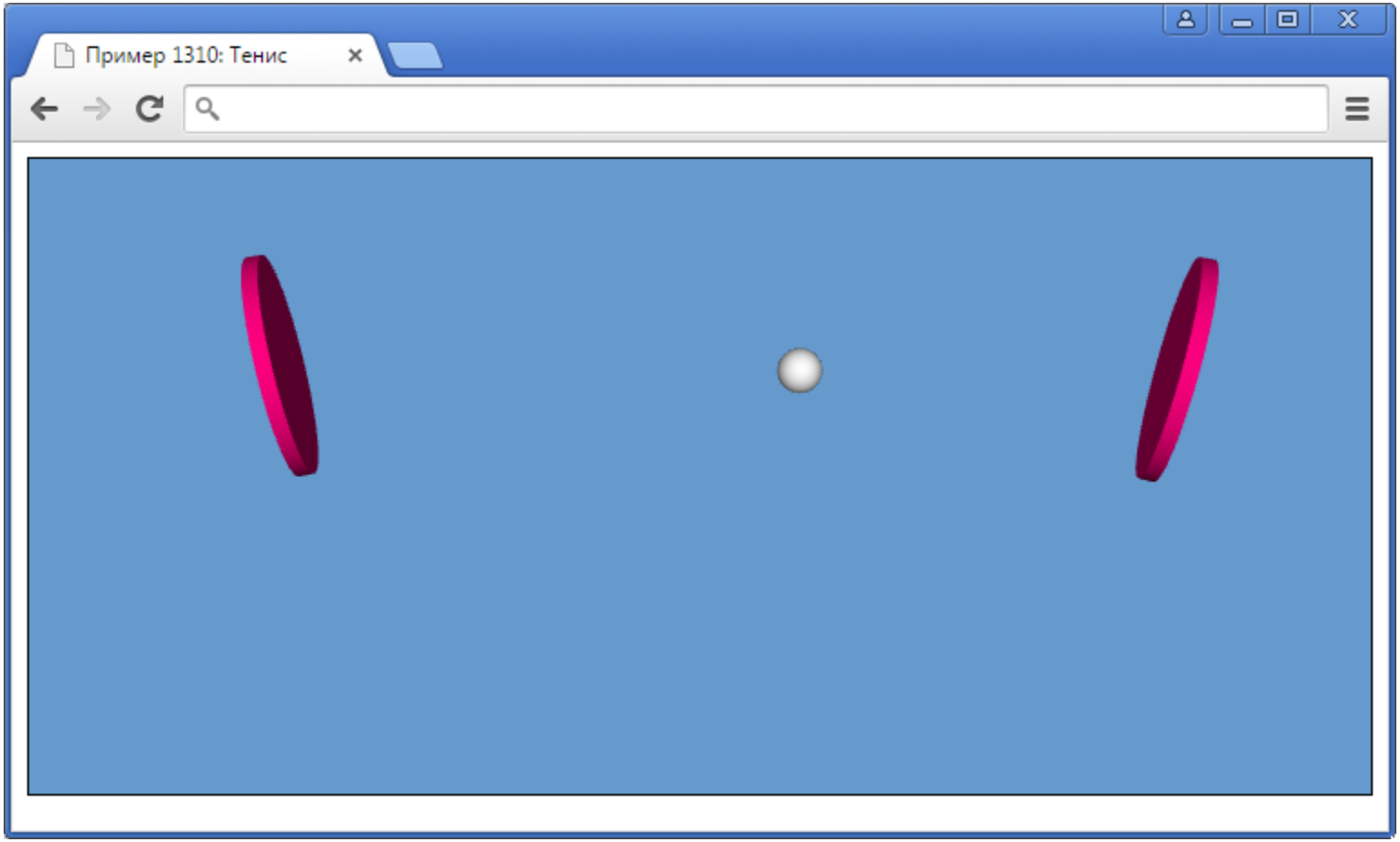
- Хилките ще са плоски цилиндри
- Движенията им по X и Z ще са синусоиди с различни коефициенти, за да се симулира хаотичност
- Координатите на всяка хилка се ползват като локална Z ос на другата хилка – това генерира накланянето им

```
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;
```

Реализация на топчето

- Топчето е сфера
- Линейна комбинация между центровете на двете хилки
- Коефициентът на линейната комбинация k не е [0, 1], а [0.04, 0.96], за да не влиза топчето в хилките
- Скоростта на топчето се контролира с коефициента 8, т.е. за топчето времето е ускорено осемкратно

```
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];
```



ПРОБА

Обобщение

Анимация

Същност

- Достатъчно бързо показване на кадри
- Възприемането на движение е илюзия
- При работа с браузър:

  Приложението заявява готовност за създаване на кадър

  Браузърът определя кога във времето да стане това

Анимационен цикъл

- Рисуването на кадър е във функция, сочена от nextFrame
- В Suica.time и Suica.dTime са изминалите времена от пускането на програмата и от предходния кадър
- Функция toFixed закръгля дробни числа

Фази на анимацията

- Ако има ясно разграничими фази на анимацията, може тя да се представи като няколко цикъла
- Ако фазите са преплетени, ползва се един цикъл с условно изпълнение на отделните фази

Линейно движение

Вектор на скоростта

- Определя посоката и скоростта на движение
- Подходящо при фиксирана цел и скорост на движение

Линейна комбинация

- Допуска динамична промяна на началната и крайната точка
- Допуска нелинейно движение

ИКТ в НОС

Край

Коментари, въпроси