Liikumine

Eelnevas näites liigutasime nupuvajutusel ühe sammu kaupa. Samuti sai varem panna kujundi hiire järgi liikuma. Kui aga panna sammud iseenesest ja piisava sagedusega korduma, siis kasutajale jääb mulje liikumisest ilma, et selle jaoks peaks ta midagi ise pidevalt tegema.

Korduvalt käivituv käsk

Tehniliselt saab käsu korduvalt käivitumise panna kirja kujul

```
window.onload=setInterval("paremale()", 500);
```

Lahtiseletatult pannakse funktsioon `paremale()` tööle iga viiesaja millisekundi tagant ehk praegusel juhul kaks korda sekundis. Nõnda siis liigubki ruut me silmade ees vaiksete hüpetega edasi. Ooteaega vähendades või sammu pikkust suurendades saab liikumist sujuvamaks või kiiremaks muuta. Samas seab masina tehniline võimsus piltide joonistamise sagedusele piirid. Enamasti ei tasu veidigi suurema joonistustahvli korral loota ülejoonistusajale alla 100 millisekundi.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      var x=20, samm=5;
      function joonista(){
         var t=document.getElementById("tahvel");
         var g=t.getContext("2d");
         g.clearRect(0, 0, t.width, t.height);
         g.fillRect(x, 20, 50, 50); //x, y, laius, kõrgus
      }
      function vasakule(){
        x=x-samm;
        joonista();
      }
      function paremale(){
        x=x+samm;
        joonista();
      }
      window.onload=setInterval("paremale()", 500);
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
          style="background-color:yellow"></canvas><br />
    <input type="button" value="v" onclick="vasakule()" />
    <input type="button" value="p" onclick="paremale()" />
  </body>
</html>
```

Joonis

Ülesandeid

* Ruut liigub vasakult paremale, klahvidega saab ruutu samal ajal liigutada alla ja üles
* Korraga liigub kaks ruutu - üks paremale, teine vasakule
* Vasakule liikuva ruudu suurus liikumise ajal väheneb
* Sammu pikkus ja sellega koos liikumise kiirus valitakse juhuslikult

Liikumise suuna määramine

Siiani liikus ruut ühes suunas, suuna määras muutuja nimega samm. Kui sellele sammule anda negatiivne väärtus, siis liigub ruut vasakule. Liikumissuuna muutmiseks rakenduse töö ajal tuleb ka sammu väärtust muuta. Eraldi lisati muutuja nimega kiirus, mille abil võimalik määrata, kui ruttu ruut liigub. Suuna määramiseks tuleb kiiruse väärtus lihtsalt sammule üle kanda - olgu siis pluss- või miinusmärgiga. Ja kui tahta ruut seisma jätta, piisab, kui sammu pikkuseks sättida null.Alla siis vastavad nupud, et kasutaja võiks vastavaid käsklusi käivitada.

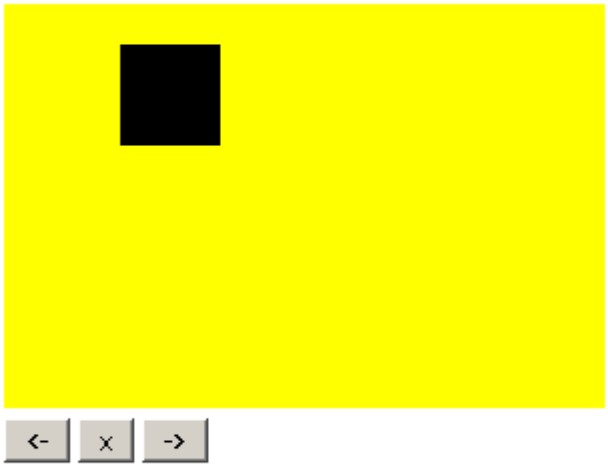
Nuppude väärtuseks olevad `<-` ja `->` võivad tunduda raskelt mõistetavad.

```
<input type="button" value="&lt;-" onclick="vasakule()" />
<input type="button" value="x" onclick="seis()" />
<input type="button" value="-&gt;" onclick="paremale()" />
```

Tegelikult aga tegemist erisümbolitega, mille abil saab sinna noole kujutise kuvada. `<` tähendab "less than" ehk "väiksem kui" ehk siis sümbol "<". Teistpidine `>` on "greater than" ehk "suurem kui" ehk ">". Kuna need märgid niisamuti kirjutatuna tähistavad HTML koodis erisümboleid, siis ei või neid otse nuputeksti kohale kirjutada, et saada noole kuju <- tekitada. Asenduse abil aga õnnestub ning nupu silt täiesti loetav.

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
```

```
            var x=20, kiirus=2, samm=kiirus;
            function joonista(){
              var t=document.getElementById("tahvel");
              var g=t.getContext("2d");
              g.clearRect(0, 0, t.width, t.height);
              g.fillRect(x, 20, 50, 50);
            }
            function liigu(){
              x=x+samm;
              joonista();
            }
            function vasakule(){
              samm=-kiirus;
            }
            function seis(){
              samm=0;
            }
            function paremale(){
              samm=kiirus;
            }
        </script>
  </head>
  <body onload="setInterval('liigu()', 100)">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />
      <input type="button" value="&lt;-" onclick="vasakule()" />
      <input type="button" value="x" onclick="seis()" />
      <input type="button" value="-&gt;" onclick="paremale()" />
  </body>
</html>
```



Ülesandeid

* Pane näide käima, uuri. Muuda liikumise kiirust
* Võimalda ruudul liikuda vasakule-paremale suuna asemel üles-alla
* Loo nupud, et saaks määrata liikumise kõigi nelja ilmakaare suunas.

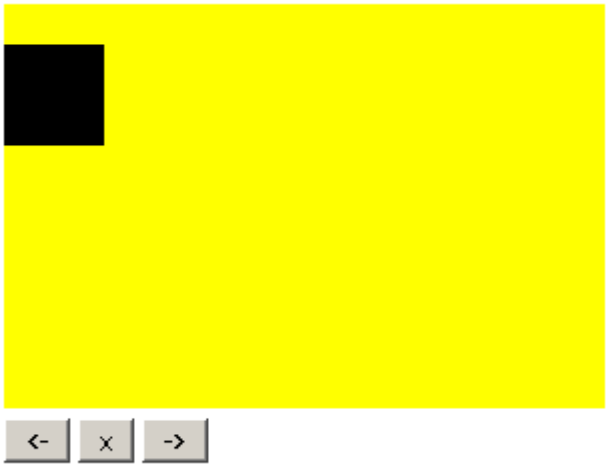
Seiskumine vasakus servas

Nõnda järjest liikudes on ruut küllalt varsti ekraanilt läinud. Ise pidevalt juhtides saab usinasti liikumist sättida. Kui aga tegemist hiljem rakendusega, kus korraga enam kujundeid liikumas, siis ei saa kõigil pidevalt käsitsi silma peal pidada ja kontrollida, et kuidas keegi käitub. Kontrollimiseks sobib tingimuslause nimega if. Praegusel juhul, kui ruut liigub liiga vasakule ehk x läheb negatiivseks, siis paras aeg öelda, et ta seisma jääks. Ning kasulik on ruudule märkida ka, et ta serva juurde tagasi tuleks (x-i väärtuseks arv 0).

```
            if(x<0){seis(); x=0;}
```

Siis pääseb ruut näiteks paremale suunava nupu vajutamisel taas liikuma. Muidu tekiks imelik olukord, kus pole võimalik ka üle ääre sattununa sealt tagasi minna, sest nullist väiksema iksi puhul antakse kohe seiskumiskäsklus.

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=100, kiirus=2, samm=-kiirus;
           function joonista(){
             var t=document.getElementById("tahvel");
             var g=t.getContext("2d");
             g.clearRect(0, 0, t.width, t.height);
             g.fillRect(x, 20, 50, 50);
           }
           function liigu(){
             if(x<0){seis(); x=0;}
             x=x+samm;
             joonista();
           }
           function vasakule(){
             samm=-kiirus;
           }
           function seis(){
             samm=0;
           }
           function paremale(){
             samm=kiirus;
           }
        </script>
  </head>
  <body onload="setInterval('liigu()', 100)">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />
      <input type="button" value="&lt;-" onclick="vasakule()" />
      <input type="button" value="x" onclick="seis()" />
      <input type="button" value="-&gt;" onclick="paremale()" />
  </body>
</html>
```



Ülesandeid

* Tee näide läbi
* Pane ruut liikuma alt üles. Ruut seiskub ülesse serva jõudmisel
* Ruutu saab liigutada iga ilmakaare suunas. Seiskub nii vasakusse kui ka ülemisse serva jõudmise puhul.

Seespüsimise kontroll

Vasaku ja ülemise serva tabamise kontroll on suhteliselt lihtne vähemasti ruudu puhul. Kuna ruut joonistatakse vasaku ülemise nurga koordinaatide järgi, siis juhul, kui need on nullist väiksemad, siis järelikult ollakse servast üle läinud. Parema ja alumise poolega on veidi rohkem arvutamist. Parema serva koordinaadi saab kätte liites vasaku serva koordinaadile ruudu laiuse. Kui nüüd selle parema serva koordinaadi väärtus ületab tahvli laiust ehk tahvli parema serva koordinaadi väärtust, sellisel juhul on ruut sealtpoolt üle läinud.

Iseenesest on võimalik nõnda tingimusi sättides ruudu seiskamist täiesti kontrollida. Arvestades aga tulevikku ning võimalust, et kontrollitavaid kujundeid tuleb rohkem, siis mitmekülgsema kontrolli tarbeks on hea teha omaette funktsioon. Näiteks selline:

```
function kasSees(uusX){
  if(uusX<0){return false;}
  if(uusX+laius>t.width){return false;}
  return true;
}
```

Funktsiooni nimeks on kasSees - nagu nimest aimata võib, siis sealt vastatakse kas jah või ei. Ehk siis true või false. Selle järgi siis võimalik hiljem otsustada kuidas käituda - kui ruudu uuritav asukoht on tahvli sees, võib sinna julgelt asuda. Kui mitte, ei tasu sinna ronida. Funktsiooni ümarsulgude sees olev `uusX` tähendab, et funktsioonile etteantav väärtus nimetatakse funktsiooni sees kasutamiseks `uusX`-i nime alla ning sealtkaudu saab tema sisu pruukida. Edasi juba järgnevad kontrollid ja vastused.

```
  if(uusX<0){return false;}
```

teatab, et kui etteantud parameetri (`uusX`) väärtus on väiksem kui null, siis funktsioon tagastab

(`return`) false, ehk siis ruut pole tervikuna tahvli sees. Sama lugu parema serva kontrolliga. Muutujas t meil eelnevalt tahvli andmed olemas. Seega siis avaldis

```
uusX+laius>t.width
```

kontrollib, et kas ruudu parema serva koordinaadi väärtus (`uusX+laius`) ületab tahvli laiust (`t.width`).

Kui mõlemad kontrollid edukalt läbitud, siis järelikult ruut x-telge pidi alas sees ning funktsioon võib tagastada väärtuse, et on sees küll (`return true`).

Liikumisfunktsioonis saab eelneva kontrolli tulemust kasutada.

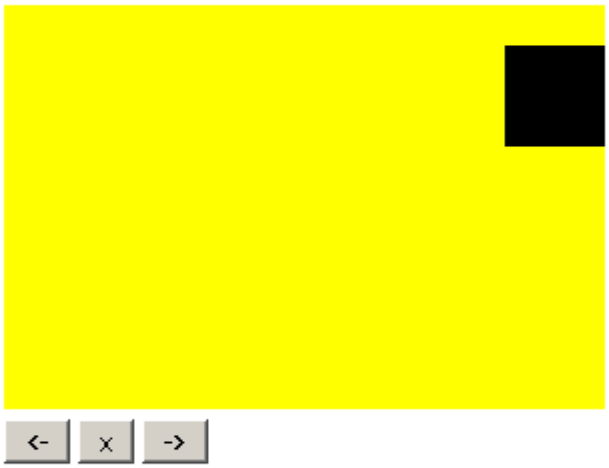
```
if(kasSees(x+samm)){
  x=x+samm;
} else {
  seis();
}
```

Esimene lause siis vaatab, et kui ruudu eeldatav uus asukoht (`x+samm`) on tahvli sees, siis minnakse sinna uude kohta kohale (`x=x+samm`). Muul juhul jäädakse seima (`seis()`).

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=100, laius=50;
           var kiirus=2, samm=-kiirus;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
             setInterval('liigu()', 100);
           }
           function joonista(){
             g.clearRect(0, 0, t.width, t.height);
             g.fillRect(x, 20, laius, 50);
           }
           function kasSees(uusX){
             if(uusX<0){return false;}
             if(uusX+laius>t.width){return false;}
             return true;
           }
           function liigu(){
             if(kasSees(x+samm)){
               x=x+samm;
             } else {
               seis();
             }
             joonista();
           }
           function vasakule(){
             samm=-kiirus;
           }
           function seis(){
             samm=0;
           }
```

```
            function paremale(){
              samm=kiirus;
            }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />
      <input type="button" value="&lt;-" onclick="vasakule()" />
      <input type="button" value="x" onclick="seis()" />
      <input type="button" value="-&gt;" onclick="paremale()" />
  </body>
</html>
```



Ülesandeid

* Pane näide tööle. Muuda ruudu suurust ning veendu, et näide töötab endiselt.
* Pane ruut liikuma ülalt alla ning selles suunas mõlemast servast seisma
* Ruutu saab liigutada iga ilmakaare suhtes. Ruut seiskub, kui jõuab serva.

Ruudu kutsumine hiirega

Nuppudega saab lehel toimetada küll. Mõnikord aga animatsioonide ja mängude juures mõistlik, kui õnnestub otse lehel olles kujundeid juhatada. Siinjuures aitavad samamoodi hiirevajutused nagu õpikus eespool kirjeldatud. Väike hoiatus: kui lehte vaadata mobiiliekraanil, siis seal tavalised hiiresündmused ei toimi, sest mobiil püüab korraga mitut näppu jälgida ning selle tarbeks on loodud touch-event. Aga sellest edasipidi.

Tahvli küljes olevale `onmousedown`-sündmuse külge sidusime funktsiooni `hiirAlla`, mille parameetrina tulnud e-ks nimetatud väärtuse kaudu saab kätte hiire andmed. Arvutus `e.clientX-tahvlikoht.left` annab hiire x-koordinaadi väärtuse tahvli asukoha suhtes. See püüti praegu muutujasse nimega `hx` (hiire x).

Praegusel juhul otsustatakse hiire järgi, et kas ruut peaks liikuma vasakule või paremale. Kui hiire

x-koordinaadi väärtus on suurem, kui ruudu x-koordinaadi väärtus, siis asub hiir ruudust paremal ning järelikult peaks ruut liikuma paremale, kui soovime, et ta hiire poole tuleks. Ehk siis ruudu liikumise sammuks saab koodi algul märgitud kiirus (mis hetkel positiivse väärtusega). Kui hiir sattus ruudust vasakule, siis sammuks saab kiiruse vastandarv, ehk siis ruut hakkab liikuma vasakule.

```
            function hiirAlla(e){
              var tahvlikoht=t.getBoundingClientRect();
              var hx=e.clientX-tahvlikoht.left;
              if(hx>x){samm=kiirus;}
              else{samm=-kiirus;}
            }
```

Edasi kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=100, laius=50;
           var kiirus=2, samm=-kiirus;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
             setInterval('liigu()', 100);
           }
           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 g.fillRect(x, 20, laius, 50);
           }
           function kasSees(uusX){
             if(uusX<0){return false;}
             if(uusX+laius>t.width){return false;}
             return true;
           }
           function liigu(){
             if(kasSees(x+samm)){
               x=x+samm;
             } else {
               seis();
             }
             joonista();
           }
           function hiirAlla(e){
             var tahvlikoht=t.getBoundingClientRect();
             var hx=e.clientX-tahvlikoht.left;
             if(hx>x){samm=kiirus;}
             else{samm=-kiirus;}
           }
           function seis(){
             samm=0;
           }

        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
                          onmousedown="hiirAlla(event)"></canvas><br />
```

```
  </body>
</html>
```

Ülesandeid

* Pane näide käima.
* Pane ruut liikuma ülalt-alla suunas (vertikaalselt). Võimalda selles sihis ruutu enese poole kutsuda.
* Muuda koodi nõnda, et hiirevajutus lükkaks ruutu pigem hiirest eemale.
* Ruut saab liikuda kõigi nelja ilmakaare suhtes. Võimalda igal pool ruutu hiirest eemale liikuma lükata (puhuda).

Liikumiskiiruse määramine hiirega

Eelmises näites lihtsalt vaadati, kuidas ruutu hiire poole meelitada. Arvestades aga kaugust hiirest, saab nõnda määrata ruudu liikumise kiiruse. Olgu siis nii, et lähemale vajutus tõmbab/lükkab tugevamini, või kehtib sama kaugema vajutuse kohta. Üheks mooduseks oleks öelda, et uueks ruudu liikumise sammuks saab kaugus hiire ja ruudu vahel (`samm=hx-x`). Sellisel puhul aga oleks ruut esimese sammuga juba hiire juures, teise sammuga sama palju möödas. Ning vähegi suurema sammude tiheduse ehk kaadrisageduse ja pikema sammu korral juba ekraanilt väljas. Sammu võib veidi väiksemas võtta seda nulli ja ühe vahel oleva kiiruskoefitsiendiga korrutades. Näiteks, kui see kordaja on 0,1 (arvutikoodis vaja koma asemel punkt kirjutada), siis kui hiir vajutatakse ruudust 30 punkti kaugusele, saab tegelikuks sammu pikkuseks kolm punkti, mida on silmaga juba suhteliselt sujuv vaadata.

```
            samm=(hx-x)*kiiruskoef;
```

Muu jääb üllatuslikult samaks, aga kiiruse muutmise paindlikkus on märgatavalt suurem.

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=100, laius=50;
           var kiirus=2, samm=-kiirus;
           var kiiruskoef=0.1;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
                 g=t.getContext("2d");
                 setInterval('liigu()', 100);
           }
           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 g.fillRect(x, 20, laius, 50);
           }
           function kasSees(uusX){
             if(uusX<0){return false;}
```

```
                if(uusX+laius>t.width){return false;}
                return true;
            }
            function liigu(){
              if(kasSees(x+samm)){
                x=x+samm;
              } else {
                seis();
              }
              joonista();
            }
            function hiirAlla(e){
              var tahvlikoht=t.getBoundingClientRect();
              var hx=e.clientX-tahvlikoht.left;
              samm=(hx-x)*kiiruskoef;
            }
            function seis(){
              samm=0;
            }

        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
                          onmousedown="hiirAlla(event)"></canvas><br />

  </body>
</html>
```

Ülesandeid

* Võimalda liikumise kiirust määrata ülalt-alla liikuva ruudu puhul.
* Ruut saab liikuda iga ilmakaare suunas. Samuti saab hiire abil määrata liikumise kiirusi.

* Pane ristkülik ekraanil laiustpidi suurenema
* Pane ristkülik ekraanil suurenema hiirevajutuse peale
* Üks hiirevajutus paneb ristküliku suurenema, teine vähenema
* Vähenemisel jääb ristküliku parem külg paigale. Nõnda on võimalik kordamööda hiirevajutuste abil jäljendada ussikese liikumist edasi.
* Ussikene pole enam ristkülik, vaid on ovaal
* Ussikene koosneb mitmest üksteise sappa ühendatud ovaalist.