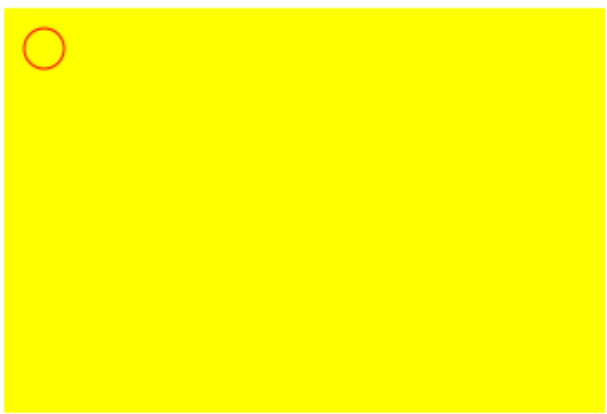
Kukkumine

Tegelikus elus toimuvaid liikumisi jäljendades tuleb mõnigikord ka kukkumist ehk vaba langemist järgi teha. Järgnevalt mõned näited, seletused ja soovitused selle kohta.

Ring ekraanil

Kukkumine kipub millegipärast vahel kergesti palliga seostuma. Pall aga teadaolevalt enamasti ringikujuline. Seetõttu tuletame meelde, kuidas sai ringi joonistada ekraanile. Ristküliku puhul piisas joonistamiseks ühest käsust. Ringi ja muude kujundite puhul aga tuleb joonistamist alustada käsuga `beginPath()`. Seejärel luua vajalikud kujundid. Ning lõpuks öelda `stroke()` või `fill()` vastavalt sellele, kas soovitakse näha vaid piirjooni või siis kogu kujund seest ära täita. Ning ringi joonistamiseks sobib kaare tegemise käsklus (`arc`). Määratakse, keskpunkt, raadius, kaare algnurk ja kaarepikkus radiaanides ning joonistamise suund. Ringi puhul võib alustada algusest (ehk nullkraadist), täisringiks on 2 piid ehk `2*Math.PI` ning joonistamise suund pole tähtis, siin näites jätsime selle `true`-ks.

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10, samm=0;
           function joonista(){
             var t=document.getElementById("tahvel");
                 var g=t.getContext("2d");
                 g.clearRect(0, 0, t.width, t.height);
                 g.strokeStyle="red";
                 g.beginPath()
                 g.arc(x, y, r, 0, 2*Math.PI, true);
                 g.stroke()
           }

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

  </body>
</html>
```



Ühtlane kukkumine

Ülalt alla liikumine toimub sarnaste arvutuste tulemusena nagu ennist vaadatud vasakult paremale liikumine - vaid muuta tuleb y-koordinaati. Algusfunktsioonis küsitakse ligipääs tahvlile ja temale joonistamiseks vajalikule graafilisele kontekstile (muutujad `t` ja `g`). Käsklus

```
setInterval('liigu()', 100);
```

teatab, et funktsioon `liigu()` tuleb käivitada iga 100 millisekundi tagant, ehk siis 10 korda sekundis. Seda ooteaega saab katsetada ja muuta, aga 100 on osutunud suhteliselt kuldseks keskteeks. Pikem ooteaeg näib juba silmale viivitusena. Lühema aja puhul aga ei pruugi seade suuta enne pilti korralikult valmis joonistada.

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10, ysamm=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.strokeStyle="red";
             g.beginPath()
             g.arc(x, y, r, 0, 2*Math.PI, true);
             g.stroke()
           }

           function liigu(){
             y=y+ysamm;
             joonista();
           }

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

  </body>
</html>
```

Ülesandeid

* Tee näide läbi. Muuda ringi suurust ja kukkumise kiirust
* Pane kukkuma mitme joonistuskäsuga loodud kujund - nt pall, millele paar täppi peale joonistatud.
* Kujundi suurust ja kukkumise kiirust saab all olevas tekstiväljas muuta.

Kiirenev kukkumine

Tavaelu jälgides saab kõrgemalt kukkudes kätte suurema kiiruse. Ehk siis kiirus kukkumise ajal kasvab. Mingist suurusest hakkab kiirust piirama õhutakistus, aga lihtsamate arvutuste juures ka füüsikaõpikus jäetakse õhutakistus välja. Väliolukorda üsna täpselt saaks jäljendada, kui arvestada, mitu ekraanipunkti vastab ühele meetrile looduses ning siis vastavalt esemed ja raskuskiirendus paika ajada. Lihtsamal juhul aga võib kukkumissammule igal ringil veidi otsa liita, et sammu pikkus suureneks. Ning sobivat tulemust saab juba silma järgi hinnata. Siin näites said kõigepealt üles muutujad sammu ja kiirenduse kohta

```
var ysamm=1, ykiirendus=0.4;
```

Liikumisfunktsioonis kõigepealt suurendatakse sammu kiirenduse jagu ning siis liigutakse igrekiga sammu jagu allapoole.

```
ysamm=ysamm+ykiirendus;
y=y+ysamm;
```

Kood tervikuna

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10;
           var ysamm=1, ykiirendus=0.4;
           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.strokeStyle="red";
            g.beginPath()
            g.arc(x, y, r, 0, 2*Math.PI, true);
            g.stroke()
        }

        function liigu(){
          ysamm=ysamm+ykiirendus;
          y=y+ysamm;
          joonista();
        }

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

  </body>
</html>
```

Ülesandeid

* Pane näide käima, katseta erinevaid algseid sammu pikkusi ja kiirenduse väärtusi
* Katseta liikumist negatiivse sammu korral
* Määra algul kiirendus ja samm nulliks. Käivita kukkumine nupuvajutuse peale.
* Kiirendust ning palli suurust saab tekstiväljas määrata.

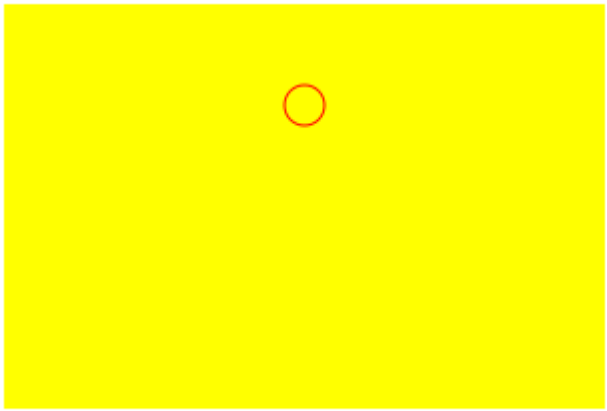
Kukkumiskoha määramine hiirega

Liikumist jällegi mugavam ekraanil juhatada hiirega. Siin näites leitakse kõigepealt hiire asukoht tahvli suhtes (muutujad `hx` ja `hy`) ning pärast paigutatakse ring vastavale kohale. Vajalik on samuti ysamm'u ehk allakukkumiskiiruse määramine nulliks, sest muul juhul jätkaks pall hiirevajutuse kohast endise kiirusega kukkumist ning varsti poleks teda ekraanil võimalik enam kuigivõrd jälgida.

```
        function hiirAlla(e){
          var tahvlikoht=t.getBoundingClientRect();
          var hx=e.clientX-tahvlikoht.left;
          var hy=e.clientY-tahvlikoht.top;
          x=hx;
          y=hy;
          ysamm=0;
        }
```

Kui nüüd nõndamoodi toimetada, siis igal hiire vajutamise korral hüppab sinna pall ja asub sealt vaikselt alla kukkuma. Kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10;
           var ysamm=1, ykiirendus=0.4;
           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.strokeStyle="red";
                 g.beginPath()
                 g.arc(x, y, r, 0, 2*Math.PI, true);
                 g.stroke()
           }

           function liigu(){
             ysamm=ysamm+ykiirendus;
             y=y+ysamm;
             joonista();
           }

           function hiirAlla(e){
             var tahvlikoht=t.getBoundingClientRect();
             var hx=e.clientX-tahvlikoht.left;
             var hy=e.clientY-tahvlikoht.top;
             x=hx;
             y=hy;
             ysamm=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 tööle, muuda palli värvi ja suurust.
* Iga järgmise hiirevajutuse peale läheb pall veidi väiksemaks.
* Joonista tahvli alumine pool siniseks. Kui pall on jõudnud allapoole veepiiri, siis hakkab ta vaikselt väiksemaks minema.
* Veepiirist allpool jaguneb pall kaheks ning kumbki tükk hakkab kukkudes vaikselt vähenema.
* Vee all palli kiirus on aeglasem ja ei muutu.

Peatumine servas

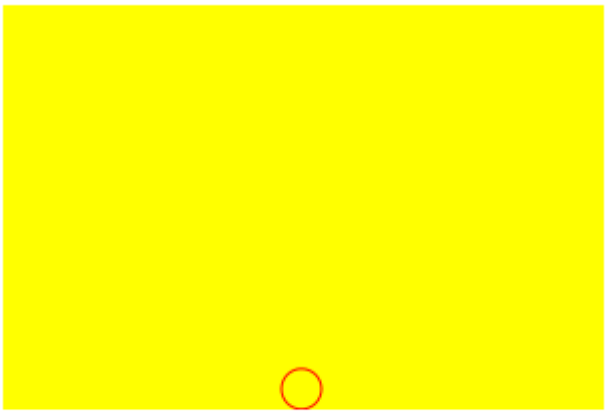
Nagu niisama liikumise juures, nii ka kukkumise puhul on vahel hea, kui kokkupandud kujundid meie tahtmata silmapiirilt ei kao. Taas tuleb kontrollida, et soovitav järgmine asukoht oleks lubatud ala sees, vaid siis liigutakse edasi. Ning tulevikule mõeldes on lisatud muutuja ka x-suunalise liikumise tarbeks. Lihtsalt kuna xsamm on parajasti 0, siis kukutakse otse alla. Kui juhtub, et uus leitud asukoht satuks tahvlilt välja, siis pannakse x ja y-sammud nulliks.

```
function liigu(){
     ysamm=ysamm+ykiirendus;
     if(kasSees(x+xsamm, y+ysamm)){
        x=x+xsamm;
        y=y+ysamm;
      } else {
        xsamm=0; ysamm=0;
      }
      joonista();
}
```

Ringi peatamise võimega kood tervikuna:

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10;
           var ysamm=1, xsamm=0, ykiirendus=0.4;
           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.strokeStyle="red";
                g.beginPath()
                g.arc(x, y, r, 0, 2*Math.PI, true);
                g.stroke()
```

```
            }

            function liigu(){
                 ysamm=ysamm+ykiirendus;
                 if(kasSees(x+xsamm, y+ysamm)){
                    x=x+xsamm;
                    y=y+ysamm;
                  } else {
                    xsamm=0; ysamm=0;
                  }
                  joonista();
            }

            function kasSees(uusX, uusY){
              if(uusX-r<0){return false;}
                  if(uusX+r>t.width){return false;}
              if(uusY-r<0){return false;}
                  if(uusY+r>t.height){return false;}
                  return true;
            }

            function hiirAlla(e){
              var tahvlikoht=t.getBoundingClientRect();
              var hx=e.clientX-tahvlikoht.left;
              var hy=e.clientY-tahvlikoht.top;
              x=hx;
              y=hy;
              ysamm=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

* Katseta näidet, muuda palli värvi, joonista seest täis.

* Tõmba ekraanile maapinda tähistav joon, jäta pall selle juures seisma.
* Maapinnani jõudnud pall hakkab uuesti ülevalt kukkuma.
* Ülevalt uue kukkumise asukoht valitakse juhuslikult.

Palli loopimine

Nüüdseks on juba piisavalt oskusi kogunenud, et midagi täiesti elavat ja usutavat kokku panna. Näitena palli heitmise rakendus, kus pall hakkab liikuma hiire poole. Mida kaugemal hiir on, seda kiiremini. Servade juures kontrollitakse, et pall sealt välja ei läheks. Kui serv ette tuleb, siis võetakse hoog maha. Kui võimalust, hakkab ta sellest kohast uuesti allapoole kukkuma. Kui alla jõudnud, siis peab lihtsalt jääma ootama, kuni hiirega uuesti vajutatakse ja pall liikuma meelitatakse.

Tehniliselt midagi väga eripärast võrreldes eelnenuga polegi. Mõlema koordinaadi puhul pannakse pall sammuma hiire suunas. Sammu pikkus võrdeline kaugusega hiirest.

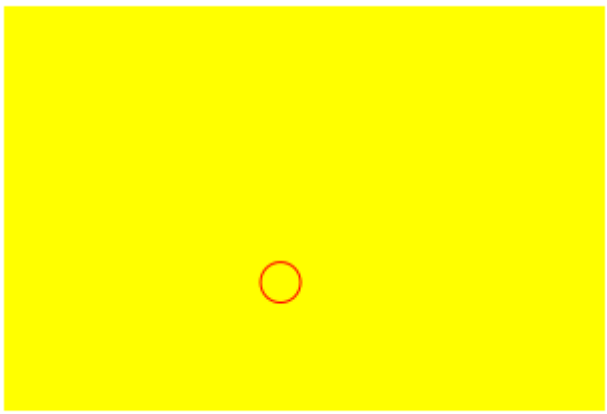
```
xsamm=(hx-x)*kiiruskoef;
ysamm=(hy-y)*kiiruskoef;
```

Tulemusena aga valmib täiesti mitmekülgne palli heitmise rakendus, mida saab edaspidi mitmesuguste liikuvate mudelite juures kasutada.

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10;
           var ysamm=1, xsamm=0, ykiirendus=0.4, 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.strokeStyle="red";
                 g.beginPath()
                 g.arc(x, y, r, 0, 2*Math.PI, true);
                 g.stroke()
           }

           function liigu(){
              ysamm=ysamm+ykiirendus;
              if(kasSees(x+xsamm, y+ysamm)){
                   x=x+xsamm;
                   y=y+ysamm;
              } else {
                   xsamm=0; ysamm=0;
              }
               joonista();
           }
           function kasSees(uusX, uusY){
             if(uusX-r<0){return false;}
```

```
            if(uusX+r>t.width){return false;}
            if(uusY-r<0){return false;}
            if(uusY+r>t.height){return false;}
            return true;
          }

          function hiirAlla(e){
            var tahvlikoht=t.getBoundingClientRect();
            var hx=e.clientX-tahvlikoht.left;
            var hy=e.clientY-tahvlikoht.top;
            xsamm=(hx-x)*kiiruskoef;
            ysamm=(hy-y)*kiiruskoef;
          }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
                          onmousedown="hiirAlla(event)"></canvas><br />

  </body>
</html>
```

Nagu piltidelt näha, saab nõnda panna palli oma asukohta muutma.



Ülesandeid

* Pane näide tööle. Muuda muuda kiiruskoefitsienti, jälgi, kuidas see mõjutab palli liikumist.
* Muuda y-suunalist kiirendust ja jälgi, kuidas see mõjutab palli kukkumist.

Täpsusviskamine

* Võta aluseks palli loopimise näide. Paiguta pall tahvli ühele poolele ning kast tahvli teisele poolele maha
* Kui pall õnnestub kasti sisse visata, siis muudab kast värvi
* Eraldi arvuga näidatakse, mitu korda on kasti tabatud. Pärast tabamist hüppab pall algkohta tagasi
* Kasutajad viskavad palli kordamööda. Loetakse, mitu korda on kummalgi pall kasti pihta läinud.

Hiirt jälitav pall

* Tutvu palli loopimise näitega. Eemalda raskuskiirendus

* Sammu arvutamine käivitatakse mitte hiire vajutamise, vaid hiire liigutamise peale (sündmus onmousemove). Pall võiks hakata hiire poole liikuma.
* Lisa süsteemi teine pall teiste algkoordinaatidega. Mõlemad pallid püüavad sarnase valemiga liikuda hiire poole.
* Teine pall ei püüa liikuda mitte hiire, vaid esimese palli poole. Nii võiks hiire liigutamise tulemusena tekkida hiire järgi pallidest ahel.