Pallide mäng

Nüüdseks on üksikuid programmeerimisoskusi juba päris palju tekkinud. Järgnevalt paneme kokku näite kus neid üheskoos pruukida saab. Ning selle näite pealt saab loodetavasti juba mitmesuguseid keerukamaid lahendusi kombineerida, kus korraga mitu kujundit ekraanil liiguvad ning kasutaja neid mõjutada saab. Olgu siis tegemist ringi sõitvate autodega, pallimänguga või mõne aparaadi jäljendamisega.

Liikuvad pallid

Varem tehtud näites liikus meil ekraani peal üks pall. Tema koordinaadid olid kirjas lehel muutujatena. Liikumise tarvis muudeti iga natukese aja tagant nende väärtusi ning siis muutujate väärtuste järgi joonistati pall lehele sobivasse kohta. Kui on tarvis panna liikuma paar-kolm palli, siis saab vajadusel seda näidet lihtsalt laiendada - igale pallile oma x-i ja y-i komplekt, mille väärtusi muudetakse ning mille järgi joonistatakse kujundid pärast ekraanil sobivasse asukohta. Iga palli lisandumisega läheb nõnda küll kood mõnevõrra pikemaks. Aga kolm-neli lisanduvat koodirida palli kohta pole veel hullu, kui pallide arv on ette teada ja piiratud.

Kui aga palle võib lehel olla kümneid - või polegi nende lõplik arv teada, sellisel juhul igale pallile omi eraldi muutujaid teha pole mõistlik. Välja aitab eelmisest peatükist tuttavaks saanud massiiv.

```
var pallid=[
  {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
  {"x":100, "y":30, "r":10, "dx":0, "dy":1}
];
```

Nõnda saab iga palli kohta kirja panna tema asukoha ja liikumise tarbeks vajalikud andmed. Nagu siin näha, siis iga palli jaoks on kirja tema x ja y-koordinaat ja raadius. Samuti sammu pikkus liikumisel: `"dx":1` tähendab, et iga kaadri ehk liikumissammu puhul suurendatakse x-i väärtust ühe võrra, `"dy":0` ütleb, et palli y-koordinaati ei muudeta, see tähendab, et ta üles- ega allapoole ei liigu.

Funktsioonid iseenesest on eelnevatest liikumisnäidetest tuttavad. Ikka tuleb iga sammu jaoks uued koordinaadid arvutada (funktsioon `liigu`) ning nende põhjal pilt välja kuvada (funktsioon `joonista`). Funktsiooni sisu lihtsalt mõnevõrra teistsugune.

```
function joonista(){
      g.clearRect(0, 0, t.width, t.height);
      for(var i=0; i<pallid.length; i++){
        g.beginPath();
        g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
               0, 2*Math.PI, true);
        g.fill();
      }
}
```

Joonistamise puhul tuleb tahvel ikka puhtaks teha alguskoordinaatidest (0, 0) kuni tahvli laiuse ja kõrguseni. Lihtsalt joonistuskäsklus vaja iga palli jaoks eraldi käivitada. Tsüklimuutuja i näitab, et mitmenda palliga tegu on. Joonistuskäsu juures `pallid[i].x` ja `pallid[i].y` annavad teada vastava palli asukoha. Kui iga palli juures eraldi käsud `g.beginPath()` ja `g.fill()`, siis pole karta, et ühe kujundi juures tõmmatud jooned võiksid järgmise omi segama hakata.

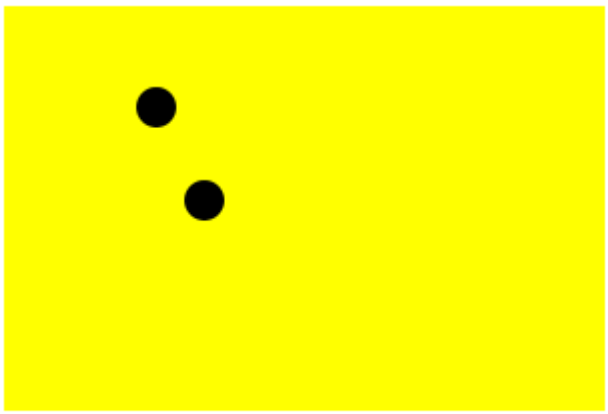
Liikumise funktsioon peab samuti arvestama, et pallide arv sõltub massiivi pikkusest.

```
            function liigu(){
              for(var i=0; i<pallid.length; i++){
                    pallid[i].x+=pallid[i].dx;
                    pallid[i].y+=pallid[i].dy;
              }
              joonista();
            }
```

Õnneks on iga palli juures kirjas, et kui palju ja kuhu poole ta liikuma peab. Käsk += lisab iga palli koordinaadile juurde liikumiseks vajaliku sammu pikkuse. Ning pärast uute asukohtade arvutamist tuleb välja kutsuda joonista-funktsioon, et ka ekraanil tulemust näha oleks. Edasi kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":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);
                 for(var i=0; i<pallid.length; i++){
                   g.beginPath();
                   g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                          0, 2*Math.PI, true);
                   g.fill();
                 }
           }

           function liigu(){
             for(var i=0; i<pallid.length; i++){
                   pallid[i].x+=pallid[i].dx;
                   pallid[i].y+=pallid[i].dy;
             }
             joonista();
           }

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

  </body>
</html>
```

Ning ekraanil võib imetleda, kuidas pallid kumbki oma suunas vaikselt liiguvad.



Ülesandeid

- Pane näide tööle
- Lisa mõned pallid
- Lisa pallile omaduseks tema värv. Tulemusena liiguvad mitut värvi pallid mööda ekraani

- Koosta kujunditena kriipsujukud. Pane nad mööda ekraani liikuma.

Põrkavad pallid

Seinast või takistuselt tagasipöördumine kuulub ikka mitmesuguste lahenduste juurde. Kui kujundeid on hulgem, tasub siingi massiivi ja tsükli võimalusi arvestada, sest suurema kujundite hulga korral ei jõua kuidagi kõike eraldi koodiridadega kontrollida. Sarnaselt varasemaga tuleb kontrollida, kas kujund ikka on lubatud ala sees et hiljem tema edasise liikumissuuna üle otsustada. Siin antakse funktsioonile kaasa terve palli objekt korraga - erinevalt enne tehtust, kus tulid eraldi parameetritena x- ja y-koordinaat. Eriti keerukamate kujundite abil on nii hea kõik andmed kätte saada ilma muretsemata, kui palju neid ühe kujundi kohta on. Kujundi sees on juba sel enda teada, mis omadused temas peituvad. Ümmarguse palli puhul saab piiridesse sobivust mugavalt arvutada. Palli vasaku serva x-koordinaadi leiab, kui keskpunktist lahutada palli raadius, palli parema serva koordinaadi leidmiseks tuleb see liita. Üles-alla suunas tuleb ülemise y-koordinaadi leidmiseks see keskpunkti omast lahutada, sest y-telg liigub arvuti joonistuskäskude puhul ülalt alla. Alumise punkti leidmiseks siis jällegi liita.

```
function kasSees(pall){
  if(pall.x-pall.r<0){return false;}
  if(pall.x+pall.r>t.width){return false;}
  if(pall.y-pall.r<0){return false;}
  if(pall.y+pall.r>t.height){return false;}
  return true;
}
```

Põrkamise puhul peab vaatama, et pall taas soovitud suunas liikuma hakkaks. Mõndapidi oleks lihtne lihtsalt liikumissammu väärtusel märk ära vahetada nii et plussist saaks miinus ja vastupidi. Sellega kipub aga vahel tekkima eriolukord, kus pall jääb ühe serva juurde "värisema", sest ei saa aru, kas on juba põrganud või mitte ja proovib uuesti suunda vahetada. Kindlam on märk nõnda sättida, et pall taas platsi keskosa poole liikuma hakkaks. Ehk kui mindi üle vasaku serva `(pall.x-pall.r<0)`, siis tuleb hoolitseda, et liikumissammuks oleks kindlasti positiivne arv

`pall.dx=Math.abs(pall.dx)`. Absoluutväärtuse võtmise käsk Math.abs võtab arvult miinusmärgi eest ära - juhul, kui see peaks olema sinna sattunud.

```
function p6rka(pall){ //Põrkab üle läinud servast tagasi
  if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
  if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
  if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
  if(pall.y+pall.r>t.height){pall.dy=-Math.abs(pall.dy);}
}
```

Liikumise juures tuleb siis kõigepealt kontrollida, kas liigutatav pall on alas sees. Juhul kui mitte, ehk käsu ees hüüumärk, mis tulemuse vastupidiseks keerab, siis vaja lasta pallil põrgata, vajalikku suunda muuta.

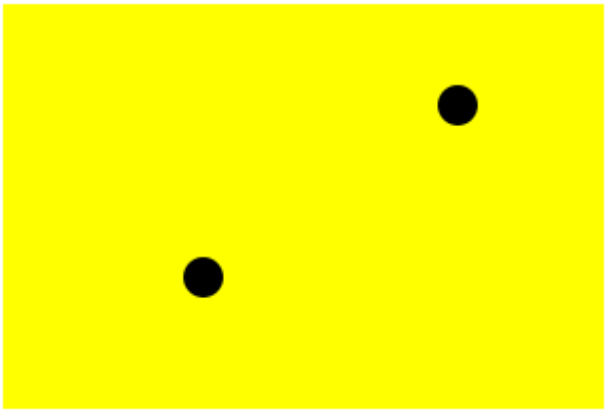
```
 if(!kasSees(pallid[i])){
    p6rka(pallid[i]);
 }
```

Nüüd jällegi peaks kood tervikuna juba mõistetav olema.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":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);
                 for(var i=0; i<pallid.length; i++){
                   g.beginPath();
                   g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                          0, 2*Math.PI, true);
                   g.fill();
                 }
           }

           function liigu(){
             for(var i=0; i<pallid.length; i++){
               if(!kasSees(pallid[i])){
                 p6rka(pallid[i]);
               }
               pallid[i].x+=pallid[i].dx;
               pallid[i].y+=pallid[i].dy;
             }
             joonista();
           }

           function kasSees(pall){
             if(pall.x-pall.r<0){return false;}
```

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

            function p6rka(pall){ //Põrkab üle läinud servast tagasi
              if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
              if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
              if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
              if(pall.y+pall.r>t.height){pall.dy=-Math.abs(pall.dy);}
            }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow" ></canvas><br />

  </body>
</html>
```

Pildi pealt palju ei näe, aga töötava rakenduse juures paistab, et kui pall jõuab servani, siis ta vahetab suunda. Kumbki omasoodu.



Ülesandeid

- Pane näide tööle
- Muuda pallide raadiused erinevateks ning veendu, et igaüks arvestab põrkamisel omi mõõtmeid
- Pane servadest põrkama liikuvad kriipsujukud. Kriipsujukude suurust määravad raadiuse asemel eraldi pikkust ja laiust tähistavad väärtused. Hoolitse, et ka põrkamisel arvestataks kriipsujukude pikkust ja laiust. Vihje: mõtle, milline on "kuum punkt", mida meeles peetakse ning mille suhtes kriipsujuku koordinaate arvestatakse.

Raskuskiirendus ja põrkamine

Reaalsete liikumiste ja hüpete juures tuleb arvestada ka kukkumisi ja kiirenemist kukkumisel. Kui liikuvaid kujundeid on palju, siis kehtib see neist igaühe puhul. Selle arvutamisel aitab y-suunaline kiirendus ehk sammu pikkuse kasv iga kaadri juures. Lihtsamal juhul võib selle väärtuse määrata

katseliselt, et tulemus võimalikult reaalne välja paistaks.

```
var ykiirendus=0.1;
```

Kiirenduse mõjumiseks saab selle liikumisfunktsiooni juures igal korral otsa liita olemasolevale dy-väärtusele. Juhul kui dy on positiivne ehk pall liigub alla, siis allasuunaline kiirus kasvab. Kui dy on negatiivne ja pall liigub üles, siis kiirenduse liitmine viib sammu pikkust nulli suunas ehk ülesliikumise kiirus väheneb. Kuni lõpuks palli ülesliikumine peatub ning asutakse taas allapoole tulema.

```
function liigu(){
  for(var i=0; i<pallid.length; i++){
    if(!kasSees(pallid[i])){
      p6rka(pallid[i]);
    }
    pallid[i].dy+=ykiirendus;
    pallid[i].x+=pallid[i].dx;
    pallid[i].y+=pallid[i].dy;
  }
  joonista();
}
```

Põrkamise juures kipub kiirenduse lisamisel tekkima anomaalia: kui pall liigub piisavalt aeglaselt ning raskuskiirendus on piisavalt suur, siis mõnikord ületab raskuskiirendusest tekkinud allaliikumise samm põrkamise poolt antud ülespoole liikuva sammu suurust ning tulemuseks on, et pall hakkab väikeste jõnksudena läbi maapinna allapoole liikuma, kuni lõpuks kaob altpoolt silmast. Väljapääsuna aitab, et pärast allasuunalist põrget tõstetakse pall igal juhul maapinnale tagasi, ehk siis y-koordinaadiks saab maapinna asukohast ehk tahvli kõrgusest palli raadiuse jagu väiksem väärtus.

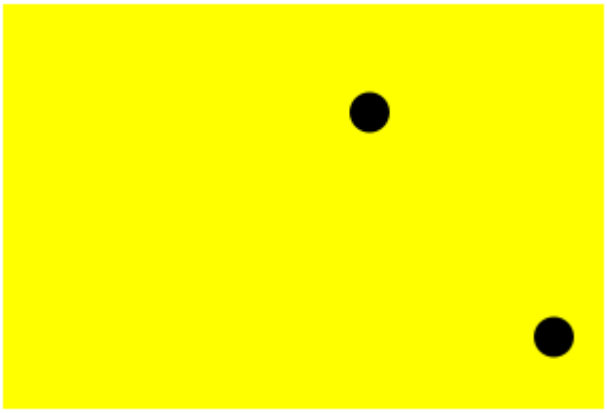
```
function p6rka(pall){
      if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
      if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
      if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
      if(pall.y+pall.r>t.height){
        pall.dy=-Math.abs(pall.dy);
        pall.y=t.height-pall.r;
      }
}
```

Töötav näide tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":-3},
             {"x":100, "y":30, "r":10, "dx":-1, "dy":1}
           ];


           var t, g; //tahvel, graafiline kontekst
           var ykiirendus=0.1;

           function algus(){
```

```
            t=document.getElementById("tahvel");
            g=t.getContext("2d");
            setInterval('liigu()', 50);
          }

          function joonista(){
                g.clearRect(0, 0, t.width, t.height);
                for(var i=0; i<pallid.length; i++){
                  g.beginPath();
                  g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                         0, 2*Math.PI, true);
                  g.fill();
                }
          }

          function liigu(){
            for(var i=0; i<pallid.length; i++){
              if(!kasSees(pallid[i])){
                p6rka(pallid[i]);
              }
              pallid[i].dy+=ykiirendus;
              pallid[i].x+=pallid[i].dx;
              pallid[i].y+=pallid[i].dy;
            }
            joonista();
          }

       function kasSees(pall){
             if(pall.x-pall.r<0){return false;}
             if(pall.x+pall.r>t.width){return false;}
             if(pall.y-pall.r<0){return false;}
             if(pall.y+pall.r>t.height){return false;}
             return true;
          }

       function p6rka(pall){
             if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
             if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
             if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
             if(pall.y+pall.r>t.height){
               pall.dy=-Math.abs(pall.dy);
               pall.y=t.height-pall.r;
             }
       }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow" ></canvas><br />

  </body>
</html>
```

Pilt püsib paigal, aga rakendust jälgides näeb, kuidas pallid kukkudes hoogu saavad ja servadest põrkavad.

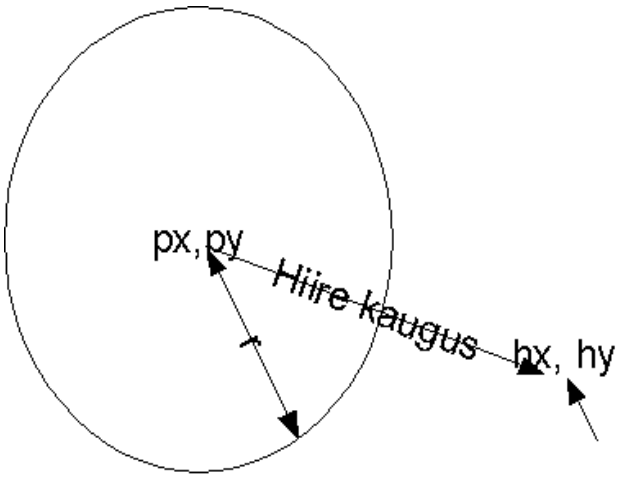


Ülesandeid

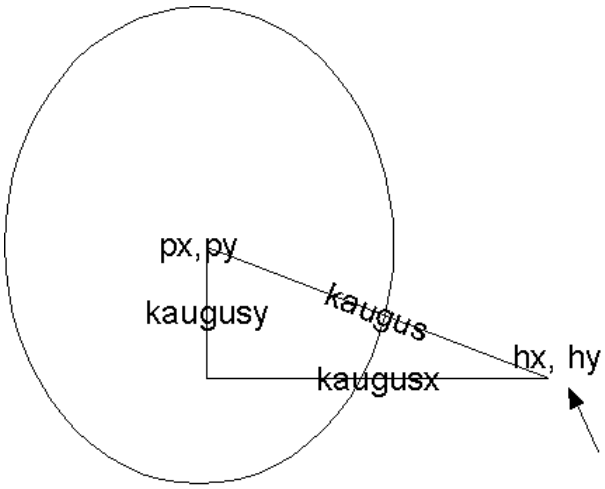
- Pane näide tööle.
- Lisa palle koos mitmesuguste algkohtade ja kiirustega.
- Muuda raskuskiirendust ning jälgi selle mõju rakenduse tööle.
- Tõmba eraldi joon maapinnaks ning lase pallidel sellelt põrgata
- Jäta pind ühest otsast lõpetamata. Tekkinud august saavad pallid alla kukkuda (põrkamisel kontrollitakse tingimust, et põrge tehakse vaid siis, kui pall on joone kohal).

Hiirevajutus ühel pallidest

Ümmarguse palli puhul piisab tabamise kindlaks tegemiseks kontrollida, kas hiire asukoht on palli keskpaigast vähem kui raadiuse kaugusel. Meil on kasutada viis arvu: palli x ja y-koordinaat (px, py), hiire x ja y-koordinaat (hx, hy) ning palli raadius (r). Silma järgi on hea vaadata, kas hiir tabas palli või mitte. Arvuti peab aga arvudega hakkama saama. Õnneks on nende andmete puhul ülesanne täiesti lahenduv.



Hiire kauguse ringi keskpunktist saab leida kahe punkti vahelise kaugusena tasandil. Selle juures omakorda aitab täisnurkse kolmnurga pikimat külge arvutada lubav Pythagorase teoreem.



Kolmnurga kaatetite pikkused saab koordinaatide lahutamise teel: hx-px on hiire asukoha ning ringi keskpunkti x-i suunaline kaugus, hy-py y-suunaline kaugus. Kas tema väärtus on pluss- või miinusmärgiga - see sõltub juba, kus suunas hiir ringi keskpunktist on. Aga õnneks arvu ruutu võtmisel arvu märk ei muuda tulemust - nii saab kauguse sarnaselt kätte ka siis, kui hiir juhtub pallist vasakul pool olema.

Koodina vormistatuna näeb arvutus välja allpool järgmine:

```
function hiirAlla(e){
  var tahvlikoht=t.getBoundingClientRect();
  var hx=e.clientX-tahvlikoht.left;
  var hy=e.clientY-tahvlikoht.top;
  for(var i=0; i<pallid.length; i++){
     var kaugusx=hx-pallid[i].x;
     var kaugusy=hy-pallid[i].y;
     var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
     if(kaugus<pallid[i].r){pallid[i].r=pallid[i].r*0.8;}
  }
  joonista();
}
```

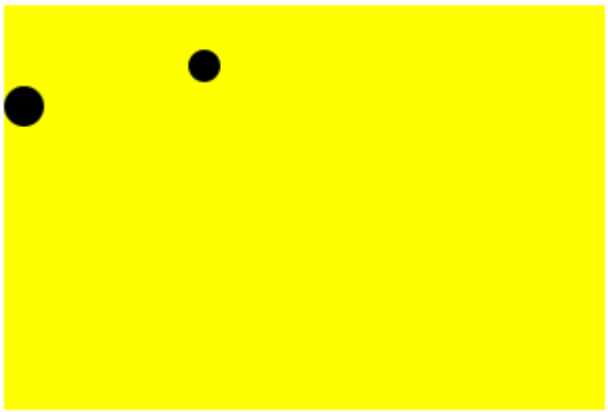
Muutujad `kaugusx` ja `kaugusy` arvutatakse kõigepealt välja, et hiljem poleks ruutu võtmise ajal vaja valemit mitu korda sisse kirjutada - `hx-pallid[i].x` annab punktide `x`-i suunalise kauguse. Math.sqrt arvutab ruutjuure ning edasi võib arvutatud kaugust juba võrrelda konkreetse palli raadiusega. Kui kaugus on piisavalt väike, siis järelikult hiir tabas palli ning palliga võib midagi ette võtta. Praegusel juhul tehakse pall veidi väiksemaks, ehk tema raadius korrutatakse 0.8ga. Nii võetakse suurest pallist rohkem maha ja väikesest pallist vähem. Vajutuse järgi tuleb eraldi välja kutsuda ka joonistuskäsk, et arvutuse tulemust saaks ka ekraanil näha.

Edasi juba kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":1}
           ];
```

```
            var t, g; //tahvel, graafiline kontekst

            function algus(){
              t=document.getElementById("tahvel");
              g=t.getContext("2d");
              joonista();
            }

            function joonista(){
                  g.clearRect(0, 0, t.width, t.height);
                  for(var i=0; i<pallid.length; i++){
                    g.beginPath();
                    g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                           0, 2*Math.PI, true);
                    g.fill();
                  }
            }

            function hiirAlla(e){
              var tahvlikoht=t.getBoundingClientRect();
              var hx=e.clientX-tahvlikoht.left;
              var hy=e.clientY-tahvlikoht.top;
              for(var i=0; i<pallid.length; i++){
                 var kaugusx=hx-pallid[i].x;
                 var kaugusy=hy-pallid[i].y;
                 var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
                 if(kaugus<pallid[i].r){pallid[i].r=pallid[i].r*0.8;}
              }
              joonista();
            }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
               onmousedown="hiirAlla(event)" ></canvas><br />

  </body>
</html>
```

Pildil näha, kuidas üks pall on hiirevajutuse tõttu väiksemaks jäänud.



Ülesandeid

- Tee näide läbi.
- Lisa mitmesuguse asukoha ja suurusega palle ning veendu, et lahendus töötab.
- Pane pallid muutuma hiirevajutuse puhul suuremaks.
- Pane pall hüppama hiirevajutuse puhul paremale
- Pane pall hüppama hiirevajutuse puhul hiirest eemale. St, et hiirega palli serva juurde vajutades valib pall uue asukoha, nagu oleks teda sealt serva juurest lükatud. Vihje: endiselt saab kätte palli keskpunkti ning hiire vahelise x-i ja y-i suunalise kauguse ning nende põhjal saab juba arvutada, kui palju ja millises suunas on mõistlik palli asukohta muuta.
- Joonista ekraanile lisaks pallidele üks suur ringjoon. Määra pallidele algul juhuslikud asukohad. Nõnda hiirega lükates tuleb pallid suure ringi sisse saada. Programm teatab, kui kõik pallid on suure ringi sees.

Liikumine ja hiirevajutus üheskoos

Eelnevad näited saab omavahel kokku tõsta. Igasugu ühendamiste puhul peab sageli vaatama, et osad omavahel tülli ei läheks. Ka siin on üheks ohuks näiteks, et samal ajal, kui setInterval-käsu kaudu tööle pandud liikumine ning kasutajalt tulnud hiirevajutuse teada püüavad samal ajal samade pallide andmeid muuta, siis võib sellega muresid tekkida. Õnneks praegusel juhul Javaskript hoolitseb, et käsud korraga tööle ei läheks ning hiire andmeid asutakse töötlema alles pärast seda, kui järjekordne liikumise samm tehtud. Tuleb lihtsalt funktsioonid sobivatele kohtadele paigutada. Ning kuna praegu kasutasid mõlemad näited samade nimedega muutujaid, siis õnneliku juhuse tõttu hakkabki rakendus tööle. Muul juhul niisama katsetades võib ühendamisel vaja olla näiteks jälgida ja sättida, et millise muutuja nime alt millised väärtused kätte saadakse.

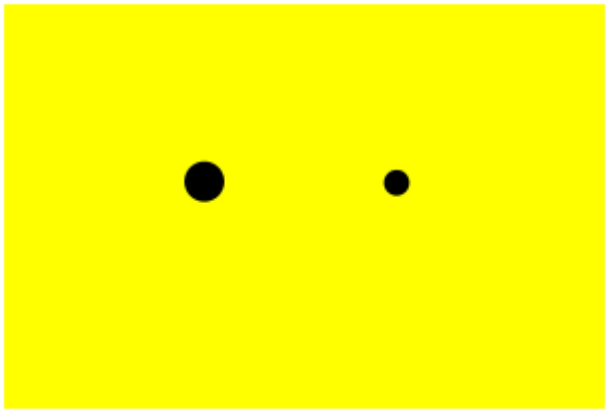
Funktsioonide järjekord Javaskriptis pole üldiselt tähtis - seetõttu võib neid lehe päiseossa vabalt sobivasse kohta paigutada lihtsalt selle järgi, kuidas neid mugavam leida on.

Näide tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":1}
           ];


           var t, g; //tahvel, graafiline kontekst
           var ykiirendus=0.1;

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
             setInterval('liigu()', 50);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 for(var i=0; i<pallid.length; i++){
                   g.beginPath();
                   g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
```

```
                    0, 2*Math.PI, true);
            g.fill();
          }
        }

        function hiirAlla(e){
          var tahvlikoht=t.getBoundingClientRect();
          var hx=e.clientX-tahvlikoht.left;
          var hy=e.clientY-tahvlikoht.top;
          for(var i=0; i<pallid.length; i++){
            var kaugusx=hx-pallid[i].x;
            var kaugusy=hy-pallid[i].y;
            var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
            if(kaugus<pallid[i].r){pallid[i].r=pallid[i].r*0.8;}
          }
          joonista();
        }

        function liigu(){
          for(var i=0; i<pallid.length; i++){
            if(!kasSees(pallid[i])){
              p6rka(pallid[i]);
            }
            pallid[i].dy+=ykiirendus;
            pallid[i].x+=pallid[i].dx;
            pallid[i].y+=pallid[i].dy;
          }
          joonista();
        }

        function kasSees(pall){
          if(pall.x-pall.r<0){return false;}
          if(pall.x+pall.r>t.width){return false;}
          if(pall.y-pall.r<0){return false;}
          if(pall.y+pall.r>t.height){return false;}
          return true;
        }

        function p6rka(pall){
          if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
          if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
          if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
          if(pall.y+pall.r>t.height){
            pall.dy=-Math.abs(pall.dy);
            pall.y=t.height-pall.r;
          }
        }

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

  </body>
</html>
```

Tulemusena siis pallid liiguvad, põrkavad ning neile vajutamisel palli suurus väheneb.



Ülesandeid

- Pane näide tööle
- Palli juures on meeles tema värv. Hiirevajutusega saab palle kordamööda punaseks ja mustaks muuta.
- Mustadele pallidele mõjub suurem kiirendus kui punastele.
- Arvuti teatab juhusliku arvu, mitu palli peaks mustad, mitu punased olema (kokku teevad need ikka algse pallide arvu). Kasutaja saab vajutustega pallide värve muuta. Arvuti teatab, kui sobiv arv musti ja punaseid on saadud.

Kellaaeg

Ajast on arvutiprogrammide juures kasu mitut moodi. Vahel on lihtsalt hea kuupäeva ja kellaaega näha. Vahel näidatakse sündmusi, mis mingi aja jooksul toimunud või tulemas. Siis jälle saab mõõta aega, et kui palju mingist hetkest kulunud on. Javaskripti juures aitab nende toimetuste juures sisseehitatud objekt tüübist Date. Kõige lihtsamal juhul väljastatakse hetke kuupäev ja kellaaeg.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      function algus(){
         document.getElementById("vastus").innerHTML=new Date();
      }
    </script>
  </head>
  <body onload="algus()">
    <div id="vastus">

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

Väljund:

Thu Oct 18 2012 21:09:53 GMT+0300 (FLE Daylight Time)

Ajavahemik

Aja arvutamise juures tuleb kasuks Date-objekti käsklus getTime(). Käsklus tagastab millisekundite arvu alates 1. jaanuarist 1970, mida peetakse mitmegi programmeerimiskeele puhul "aegade alguseks". Ehkki käsklus väljastab vaid ühe arvu, saab seda programmide sees mitmel moel kasulikult pruukida. Kulunud aja arvestamiseks tasub meelde jätta algusaeg. Edasi saab juba iga funktsioonikäivituse juures leida uus kaugus aegade algusest millisekundites ning arvestada, kui palju vahepeal aega on kulunud. Kui millisekundites kaugus käes, siis edasi saab seda arvutuste teel juba sobivatesse ühikutesse teisendada.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      var algusaeg=new Date().getTime();
      function algus(){
         setInterval('liigu()', 1000);
      }
      function liigu(){
         document.getElementById("vastus1").innerHTML=new Date();
         document.getElementById("vastus2").innerHTML=
           new Date().getTime() + " millisekundit 1. jaanuarist 1970";
         document.getElementById("vastus3").innerHTML=
           "Kulunud "+parseInt((new Date().getTime()-algusaeg)/1000)+" sek ";
      }
    </script>
  </head>
  <body onload="algus()">
    <div id="vastus1"></div>
    <div id="vastus2"></div>
    <div id="vastus3"></div>
  </body>
</html>
```

Väljund:

Thu Oct 18 2012 21:11:07 GMT+0300 (FLE Daylight Time)
1350583867598 millisekundit 1. jaanuarist 1970
Kulunud 49 sek

Ülesandeid

- Pane ajaga seotud näited käima
- Näita, mitu minutit ja mitu sekundit on kulunud lehe avamise algusest. Täisosa leidmiseks sobib käsklus parseInt.
- Kasutajale näidatakse juhuslikku arvu, mitme millisekundi möödumist ta peaks püüdma tabada. Kui kasutaja arvates on sobiv aeg, siis ta vajutab nuppu. Arvuti näitab, kui pikka ajavahemikku inimeselt küsiti, kui palju kulus tegelikult, kui palju oli nende aegade vahe ning mitu protsenti eksiti.
- Püüa Internetist leida näide, kuidas töötab Javaskripti Date-objekti käsklus getHours(). Pane ekraanile tiksuma kell, kus näha tunnid, minutid ja sekundid.

Liikumine, hiir ja aeg

Ka ajaarvestuse saab eelnevale rakendusele küllalt viisakasti lisada. Algusesse muude muutujate kõrvale tuleb algusaeg, pärast hea sellega võrreldes arvestada, et kui palju on lehe töö algusest aega kulunud. Muutuja pihtasloendur aitab kokku lugeda, et mitu korda mõnd palli on tabatud.

Võimalusel on hea keele sõnakujudega arvestada. Hädapärast ju saab teadetega hakkama kujul "avatud on 1 leht(e)" - aga jäägu sellised sõnaväänamised olukordadesse, kus tuleb koodi muutmata tõlkida võõrkeelset rakendust ning pole võimalik vastavalt arvule sõna sättida. Viisakamal juhul aga tasub tingimuslausega vaadata, et ikka tulemusele vastav sõna kirja saaks. Üheks mooduseks on anda algul muutujale üks väärtus ning siis edasi tingimuslause abil kontrollida, kas seda äkki asendada on vaja. Hiljem lauses saab siis muutuja kaudu sobival kujul sõna välja trükkida.

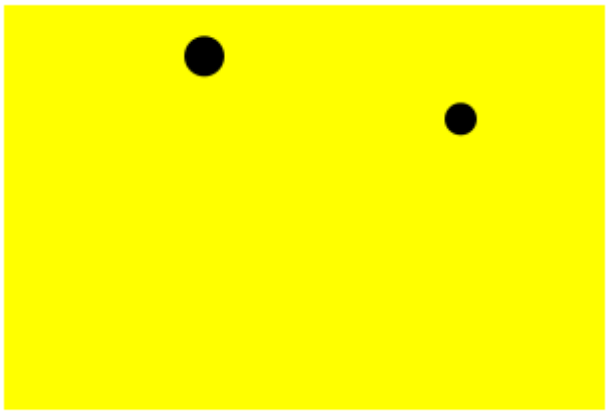
```
var lopusona="pall";
if(pihtasloendur>1){
  lopusona="palli";
}
document.getElementById("vastus2").innerHTML=
   "Tabatud "+pihtasloendur+" "+lopusona;
}
```

Kood tervikuna, mille abil saab hiirega tabada liikuvaid palle ning loetakse kokku, kui kaua aega läinud ning mitu palli pihta saadud.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":1}
           ];


           var t, g; //tahvel, graafiline kontekst
           var ykiirendus=0.1;
           var algusaeg=new Date().getTime();
           var pihtasloendur=0;

           function algus(){
             t=document.getElementById("tahvel");
                 g=t.getContext("2d");
                 setInterval('liigu()', 50);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 for(var i=0; i<pallid.length; i++){
                   g.beginPath();
                   g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                          0, 2*Math.PI, true);
                   g.fill();

                 }
           }

           function hiirAlla(e){
             var tahvlikoht=t.getBoundingClientRect();
             var hx=e.clientX-tahvlikoht.left;
```

```
            var hy=e.clientY-tahvlikoht.top;
            for(var i=0; i<pallid.length; i++){
              var kaugusx=hx-pallid[i].x;
              var kaugusy=hy-pallid[i].y;
              var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
              if(kaugus<pallid[i].r){
                pallid[i].r=pallid[i].r*0.8;
                pihtasloendur++;
                var lopusona="pall";
                if(pihtasloendur>1){
                  lopusona="palli";
                }
                document.getElementById("vastus2").innerHTML=
                   "Tabatud "+pihtasloendur+" "+lopusona;
                }
            }
            joonista();
          }

          function liigu(){
            for(var i=0; i<pallid.length; i++){
              if(!kasSees(pallid[i])){
                p6rka(pallid[i]);
              }
              pallid[i].dy+=ykiirendus;
              pallid[i].x+=pallid[i].dx;
              pallid[i].y+=pallid[i].dy;
            }
            document.getElementById("vastus1").innerHTML="Kulunud "+
                    parseInt((new Date().getTime()-algusaeg)/1000)+" sek ";
            joonista();
          }

          function kasSees(pall){
              if(pall.x-pall.r<0){return false;}
              if(pall.x+pall.r>t.width){return false;}
              if(pall.y-pall.r<0){return false;}
              if(pall.y+pall.r>t.height){return false;}
              return true;
          }

          function p6rka(pall){
            if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
            if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
            if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
            if(pall.y+pall.r>t.height){
            pall.dy=-Math.abs(pall.dy);
            pall.y=t.height-pall.r;
          }
       }

        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
             onmousedown="hiirAlla(event)" ></canvas><br />
      <div id="vastus1"></div>
      <div id="vastus2"></div>
  </body>
</html>
```

Rakenduse ekraanikuva:



Kulunud 11 sek
Tabatud 1 pall

Ülesandeid

- Pane näide tööle
- Korralda nõnda, et iga viie sekundi tagant lähevad pallid uuesti ühesuurusteks tagasi. Jäetakse meelde suurim tabamuste arv, mida ühe sellise viiesekundise ajaga õnnestunud saada.
- Igal järgmisel korral ühesuuruseks minnes on pallid eelmisest korrast natuke väiksemad ning tabamiseks mõeldud aeg kasvab.

Tulemuste loetelu

Edetabelid ja tulemuste loetelud kipuvad mängude ja võistluste juurde kuuluma - ehkki nendega põhjust ka mujal kokku puutuda. Eelnevalt otsingu juures koostasime mõne, aga siin väike meeldetuletus, kuidas mälus olevaid andmeid veebilehel esitada kannatab. Kõigepealt koostatakse näitamise tarbeks muutuja ning for-tsükli abil HTML-kujul loetelu ning edasi näidatakse selle väärtus kihi innerHTML-omaduse kaudu lehele. Kuna võrrelduna eelneva <ul> - loetelu asemel (unordered list) kasutatakse <ol> - loetelu (ordered list), siis kuvamisel näidatakse automaatselt välja osaliste järjekorranumbrid.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      var tulemused=[
        {"eesnimi":"Juku", "punkte": 7},
        {"eesnimi":"Kati", "punkte": 6},
        {"eesnimi":"Mati", "punkte": 5}
      ];
      function algus(){
         var t="<ol>";
         for(var i=0; i<tulemused.length; i++){
           t+="<li>"+tulemused[i].eesnimi+": "+
             tulemused[i].punkte+" punkti</li>";
         }
```

```
        t+="</ol>";
        document.getElementById("vastus").innerHTML=t;
      }
    </script>
  </head>
  <body onload="algus()">
    <div id="vastus">

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

1. Juku: 7 punkti
2. Kati: 6 punkti
3. Mati: 5 punkti

Loetellu lisamine

Eelnevalt vaid näidati massiivis olevaid andmeid. Töötava rakenduse käigus aga andmed täienevad ja muutuvad ning ka neid on põhjust sobival ajal välja näidata.
Tulemusi saab massiivi lisada push-käsklusega. Kuna siin iga massiivi element koosneb osaleja eesnimest ja tema punktide arvust, siis tasub järjepidevuse hoidmiseks ka uue osaleja andmed samal kujul lisada. Lihtsuse mõttes võtame esialgu andmed tekstikastidest. Looksulgude vahel luuakse üks osaleja kirje mil väljadeks "eesnimi" ja "punkte". Kogu see uus kirje läheb push-käsu ümarsulgude vahele, mille tulemusena ta massiivi lõppu jõuab. Pärast lisamist on põhjust tulemused uuesti kuvada, et ikka näeks mis lisatud.

```
function lisaTulemus(){
  tulemused.push(
    {
      "eesnimi":document.getElementById("kast1").value,
      "punkte": parseInt(document.getElementById("kast2").value)
    }
  );
  kuvaTulemused();
}
```

Ning näite kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      var tulemused=[
        {"eesnimi":"Juku", "punkte": 7},
        {"eesnimi":"Kati", "punkte": 6},
        {"eesnimi":"Mati", "punkte": 5}
      ];
      function algus(){
        kuvaTulemused();
```

```
      }
      function kuvaTulemused(){
         var t="<ol>";
         for(var i=0; i<tulemused.length; i++){
           t+="<li>"+tulemused[i].eesnimi+": "+
             tulemused[i].punkte+" punkti</li>";
         }
         t+="</ol>";
         document.getElementById("vastus").innerHTML=t;
      }
      function lisaTulemus(){
        tulemused.push(
          {
            "eesnimi":document.getElementById("kast1").value,
            "punkte": parseInt(document.getElementById("kast2").value)
          }
        );
        kuvaTulemused();
      }
    </script>
  </head>
  <body onload="algus()">
    Eesnimi: <input type="text" id="kast1" />
    Punkte: <input type="text" id="kast2" />
    <input type="button" value="Lisa tulemus" onclick="lisaTulemus()" />
    <div id="vastus">

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

Eesnimi: Siim Punkte: 8 Lisa tulemus

1. Juku: 7 punkti
2. Kati: 6 punkti
3. Mati: 5 punkti
4. Siim: 8 punkti

Ülesandeid

- Pane näide tööle, muuda andmeid.
- Trüki välja vaid need osalejad, kel on rohkem kui viis punkti
- Koosta massiiv, kus on kirjas vaid punktide arvud, kuva arvude loetelu lehele. Vihjena sobib vaadata nimede loetelu otsingu peatükis. Arvudele pole jutumärke ümber vaja.
- Võimalda veebilehe kaudu arve loetellu lisada. Koos lisamisega kuva loetelu olemasolevatest arvudest.
- Püüa leida loetelust suurim punktide arv. Vihjeks aitab abimuutuja, mille väärtuseks kõigepealt esimene väärtus. Siis iga järgmise väärtuse puhul kontrollitakse, kas leitud väärtus on abimuutuja omast suurem. Kui jah, siis omistatakse leitud väärtus abimuutujasse. Pärast kõikide väärtuste võrdlemist on nõnda suurim käes ja võib välja trükkida.

Tulemuste järjestamine

Paljude kasutajatega edetabelis tuntakse sageli huvi oma koha vastu. Õnneks mõistab Javaskript väärtusi ka mõne tunnuse alusel ritta sättida ning sealtkaudu järjestatud loetelu kokku panna. Tavalise massiivi puhul võib öelda lihtsalt sort. Ehk siis eesnimede järjestamiseks

```
var eesnimed=new Array("Siim", "Anu", "Sass");
eesnimed.sort();
```

ning tulemuseks ongi Anu eespool ja poisid tagapool. Kui aga igal kirjel mitu tunnust - praegusel juhul "eesnimi" ja "punkte", siis ei tea Javaskript, mille järgi järjestada soovitakse, see tuleb eraldi teada anda. Teada andmiseks tuleb sorteerimisfunktsioonil aidata võrrelda korraga kahte massiivielementi omavahel - et kumb neist peaks pärast sortimist järjekorras eespool olema. Võrdluse vastusena tagastatakse arv: nullist väiksem vastus tähendab, et eespool peaks olema esimene võrreldav element, nullist suurem vastus et teine võrreldav element. Ning kui vastava tunnuse järgi järjestamisel kahe võrreldava elemendi väärtustel vahet pole, siis tuleb funktsioonil tagastada arv 0. Õnneks saab sellise võrdluse lihtsasti teha lahutustehte abil. Arvutus `t2.punkte-t1.punkte` teeb punktide arvutamisel just seda mida vaja. Kuna soovitakse suurema punktisummaga osalejad panna loetelus ettepoole, siis t1 peaks jääma loetelus eespoole vaid siis, kui seal on rohkem punkte kui t2 juures. Sellisel juhul peaks väljastatav tulemus olema alla nulli ning nii see tulebki, kui suurema punktiväärtusega t1 on miinusmärgi taga.

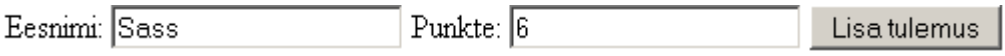
```
function sordiTulemusedPunktideJ2rgi(){
  tulemused.sort(function(t1, t2){return t2.punkte-t1.punkte});
}
```

Edasi taas töötav rakendus.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      var tulemused=[
        {"eesnimi":"Juku", "punkte": 7},
        {"eesnimi":"Kati", "punkte": 8},
        {"eesnimi":"Mati", "punkte": 5}
      ];
      function algus(){
        kuvaTulemused();
      }
      function sordiTulemusedPunktideJ2rgi(){
        tulemused.sort(function(t1, t2){return t2.punkte-t1.punkte});
      }
      function kuvaTulemused(){
         sordiTulemusedPunktideJ2rgi();
         var t="<ol>";
         for(var i=0; i<tulemused.length; i++){
           t+="<li>"+tulemused[i].eesnimi+": "+
             tulemused[i].punkte+" punkti</li>";
         }
         t+="</ol>";
         document.getElementById("vastus").innerHTML=t;
```

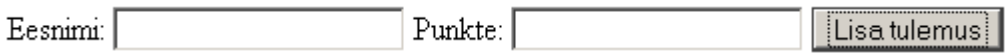
```
      }
      function lisaTulemus(){
        if(document.getElementById("kast1").value.length==0){return;}
        tulemused.push(
          {
            "eesnimi":document.getElementById("kast1").value,
            "punkte": parseInt(document.getElementById("kast2").value)
          }
        );
        document.getElementById("kast1").value="";
        document.getElementById("kast2").value="";
        kuvaTulemused();
      }
    </script>
  </head>
  <body onload="algus()">
    Eesnimi: <input type="text" id="kast1" />
    Punkte: <input type="text" id="kast2" />
    <input type="button" value="Lisa tulemus" onclick="lisaTulemus()" />

    <div id="vastus">

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

Kõigepealt näha algul massiivi pandud tegelased.



1. Kati: 8 punkti
2. Juku: 7 punkti
3. Mati: 5 punkti

Neile lisatakse Sass kuune punktiga. Ning omaloodud abifunktsiooniga sort-käsklus suudab Sassi punktide järgi just õigele kohale paigutada.



1. Kati: 8 punkti
2. Juku: 7 punkti
3. Sass: 6 punkti
4. Mati: 5 punkti

Ülesandeid

- Tee näide läbi
- Sorteeri sisestatud andmed nõnda, et vähem punkte saanud on eespool
- Sorteeri sisestatud andmed eesnimede pikkuste järgi kasvavasse järjekorda. Teksti pikkuse saab kätte käsuga length, näiteks inimene1.eesnimi.length

- Sorteeri sisestatud andmed eesnimede järgi tähestikulisse järjekorda. Sel puhul aitab võrdlemiseks if-lause ning sobivas suuruses arvu tagastamine. Näiteks
  `if(inimene1.eesnimi<inimene2.eesnimi){return -1;}`

Valmis mäng

Eraldi tükid nüüdseks mitut moodi läbi proovitud. Edasi juba tasub tulemus kasutatavaks mänguks viimistleda ning sobiv kasutajaskond leida. Või siis tehtut pruukida toimiva alusena, mille peale omi mitmesuguseid lahendusi looma hakata. Väidetakse, et pärast seda, kui lahendus kuidagi tööle saadud, kulub vähemalt veerand kui mitte pool tööd selle tarbeks, et kasutajatel sellega ka mugav toimetada oleks. Püüame siingi mõned täiendused teha, et lahendus rohkem "päris" mängu moodi välja paistaks.

Edetabeli lisandumisega on hea mäng konkreetseteks etappideks jagada. Ehk siis igale mängija saab omale piiratud aja, mille jooksul võimalikult rohkem palle tabada. Tema tulemus jäetakse meelde ning salvestatakse edetabelisse. Punktide/tabamuste arvu ei küsita enam tekstiaknast, vaid need saab kätte konkreetse mängu tulemuste kaudu. Andmete lisamiseks eraldi funktsioon lisaTulemus. Parameetrina on põhjust ette anda vaid kasutaja eesnimi, punktide arv tuleb muutujast pihtasloendur. Looksulgude vahel luuakse uus kirje ning lisatakse tulemuste loendisse.

```
function lisaTulemus(enimi){
  tulemused.push(
    {
      "eesnimi":enimi,
      "punkte": pihtasloendur
    }
  );
  kuvaTulemused();
}
```

Eraldi peetakse arvet mängu seisundi üle. Muutuja `m2ngK2ib` on `true` või `false` vastavalt sellele, kas kasutaja saab parajasti hiirega palle väiksemaks klõpsida või mitte. Kõigepealt ei saa. Kui aga kasutaja vajutab alustamise nupule, siis sellest hetkest alates saab. Kui mängu kestuse jagu aega mööda tiksunud, siis taas ei saa. Ning kui kasutaja on oma eesnime sisestanud ja punktid salvestunud, siis jääb mäng uue mängija algusnupuvajutust ootama. Selle toimumisel taastatakse algseaded ning too võib taas algusest peale pallikesi püüdma asuda. Algseadete taastamisel siis nullitakse pihtasloendur, võetakse uus algusaeg, peidetakse alustusnupp ning muudetakse kõikide pallide raadius algseks. Samuti määratakse seisinud m2ngKäib true-ks, et rakendus taas hiirevajutustele oleks valmis reageerima.

```
function m2nguAlgus(){
   pihtasloendur=0;
   algusaeg=new Date().getTime();
   document.getElementById("nupp1").style.visibility="hidden";
   for(var i=0; i<pallid.length; i++){
     pallid[i].r=algraadius;
   }
   m2ngK2ib=true;
}
```

Edasi võib juba rahus töötavat rakendust imetleda ning mõelda, et mis temaga peale võiks hakata.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
```

```
  <script>
var pallid=[
       {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
       {"x":100, "y":30, "r":10, "dx":0, "dy":1}
];

var tulemused=[
  {"eesnimi":"Juku", "punkte": 7},
  {"eesnimi":"Kati", "punkte": 8},
  {"eesnimi":"Mati", "punkte": 5}
];

var t, g; //tahvel, graafiline kontekst
var ykiirendus=0.1;
var algusaeg=new Date().getTime();
var pihtasloendur=0, m2ngK2ib=false;
var m2nguKestus=10000; //millisekundit
var algraadius=10;

function algus(){
       t=document.getElementById("tahvel");
       g=t.getContext("2d");
       setInterval('liigu()', 50);
}

function m2nguAlgus(){
   pihtasloendur=0;
   algusaeg=new Date().getTime();
   document.getElementById("nupp1").style.visibility="hidden";
   for(var i=0; i<pallid.length; i++){
     pallid[i].r=algraadius;
   }
   m2ngK2ib=true;
}

function joonista(){
    g.clearRect(0, 0, t.width, t.height);
    for(var i=0; i<pallid.length; i++){
      g.beginPath();
      g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                0, 2*Math.PI, true);
      g.fill();
    }
}

function hiirAlla(e){
   if(!m2ngK2ib){return;}
   var tahvlikoht=t.getBoundingClientRect();
   var hx=e.clientX-tahvlikoht.left;
   var hy=e.clientY-tahvlikoht.top;
   for(var i=0; i<pallid.length; i++){
      var kaugusx=hx-pallid[i].x;
      var kaugusy=hy-pallid[i].y;
      var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
      if(kaugus<pallid[i].r){
        pallid[i].r=pallid[i].r*0.8;
        pihtasloendur++;
        var lopusona="pall";
        if(pihtasloendur>1){
          lopusona="palli";
        }
        document.getElementById("vastus2").innerHTML=
          "Tabatud "+pihtasloendur+" "+lopusona;
      }
```

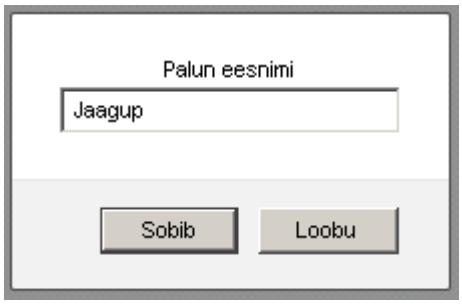
```
    }
    joonista();
}

function liigu(){
    if(!m2ngK2ib){return;}
    for(var i=0; i<pallid.length; i++){
      if(!kasSees(pallid[i])){
        p6rka(pallid[i]);
      }
      pallid[i].dy+=ykiirendus;
      pallid[i].x+=pallid[i].dx;
      pallid[i].y+=pallid[i].dy;
    }
    document.getElementById("vastus1").innerHTML=
      "Kulunud "+parseInt((new Date().getTime()-algusaeg)/1000)+" sek ";

    joonista();
    if(new Date().getTime()-algusaeg>m2nguKestus){
      m2ngK2ib=false;
      lisaTulemus(prompt("Palun eesnimi", "Niptiri"));
      document.getElementById("nupp1").style.visibility="visible";
    }
  }

  function kasSees(pall){
        if(pall.x-pall.r<0){return false;}
        if(pall.x+pall.r>t.width){return false;}
        if(pall.y-pall.r<0){return false;}
        if(pall.y+pall.r>t.height){return false;}
        return true;
  }

  function p6rka(pall){ //Põrkab üle läinud servast tagasi
        if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
        if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
        if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
        if(pall.y+pall.r>t.height){
          pall.dy=-Math.abs(pall.dy);
          pall.y=t.height-pall.r;
        }
  }


 function sordiTulemusedPunktideJ2rgi(){
      tulemused.sort(function(t1, t2){return t2.punkte-t1.punkte});
 }
 function kuvaTulemused(){
    sordiTulemusedPunktideJ2rgi();
    var t="<ol>";
    for(var i=0; i<tulemused.length; i++){
      t+="<li>"+tulemused[i].eesnimi+": "+
        tulemused[i].punkte+" punkti</li>";
    }
    t+="</ol>";
    document.getElementById("vastus3").innerHTML=t;
 }
 function lisaTulemus(enimi){
   tulemused.push(
     {
       "eesnimi":enimi,
       "punkte": pihtasloendur
     }
   );
```

```
        kuvaTulemused();
      }

        </script>
  </head>
  <body onload="algus()">
      <h1>Vajuta hiirega pallile</h1>
      <canvas id="tahvel" width="300" height="200"
              style="background-color:yellow"
              onmousedown="hiirAlla(event)" ></canvas><br />
      <div id="vastus1"></div>
      <div id="vastus2"></div>
      <div id="vastus3"></div>
      <input type="button" id="nupp1" value="Alusta" onclick="m2nguAlgus()" />
  </body>
</html>
```

Mängu algus tervitab meid tühja ekraaniga. Kasutaja saab ise valida, millal mängu alustada ja aeg käima panna. Alustusnupp kaob ning võimalik asuda palle püüdma.

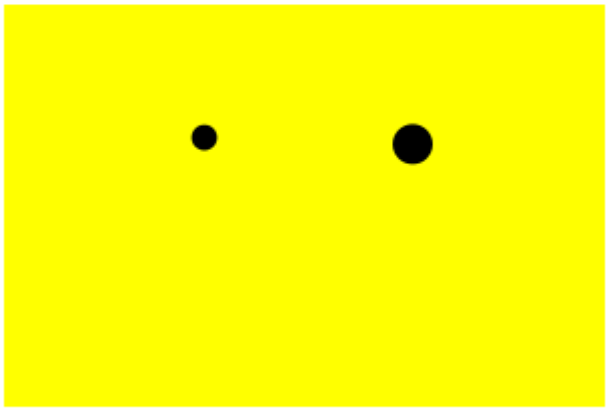
Hiirevajutuse tulemusena läheb pihtasaadud pall väiksemaks. Näidatakse sellel kasutajajal kulunud aega ning tabatud pallide arvu.

Kui kümme sekundit täis, siis küsitakse mängija eesnimi.



Sisestatud nimi koos punktidega läheb edetabelisse sobivale kohale ning võib oma tulemust teiste omadega võrrelda. Alla tekib nupp, millest saab uus mängija taas pallid liikuma lükata.

Vajuta hiirega pallile



Kulunud 10 sek
Tabatud 2 palli

1. Kati: 8 punkti
2. Juku: 7 punkti
3. Mati: 5 punkti
4. Jaagup: 2 punkti

Alusta

Ülesandeid

Pane näide tööle
Kui tulemuse salvestamisel juba olemasoleva nimega mängija saab parema tulemuse kui tal enne oli, siis salvestatakse tema nime alla uus tulemus. Kui aga sellist mängijat veel pole, siis lisatakse ta koos punktidega uuena.
Iga mängija juures loetakse lisaks saadud punktidele ka, et mitu mängu ta ühes avanenud aknas mänginud on. Vormista tulemused tabelina.

Palliga pallide tabamine

- Hulk palle põrkab ekraanil omasoodu. Ühe teist värvi palli liikumist saab kasutaja hiirega eraldi määrata.
- Iga iseliikuva palli peal on arv, mitu korda on ta juhitava pallis vastu põrganud. Kui juhitav pall puutub vastu ise liikuvat palli, siis vastav number suureneb ning tabatud pall hüppab juhuslikku kohta.
- Iseliikuvaid palle on rohelisi ja punaseid, mõlemil tabamisarvud peal. Rohelisi tuleb punktide saamiseks tabada, punastest hoiduda. Loetakse kokku, kui palju kumbagi tüüpi palle tabatud on.

Pallivise üle seina

- Tahvli keskel on ligikaudu poole tahvli kõrguseni ulatuv püstine sein (joon). Liikuv ja põrkav pall jääb seina puudutades pidama.

- Pall tekib algul vasakusse alanurka. Kasutaja saab palli suunda ja kiirust hiirega määrata. Palli saab loopida üle seina, aga seina tabamisel jääb pall kinni.
- Platsil on kaks seina, üks kasvab välja põrandast teine laest. Mängija ülesandeks on anda pallile selline paras hoog, et pall lendaks esimesest seinast üle ja teise alt läbi vajadusel maast põrgates.