Ringjoonega seotud arvutused

Ring kipub vähegi graafilisemate lahenduste juures ikka ette tulema. Olgu siis tegemist elementide paigutamise, liikumise või vaatesuundade arvutamisega.

Ringjoone parameetriline võrrand

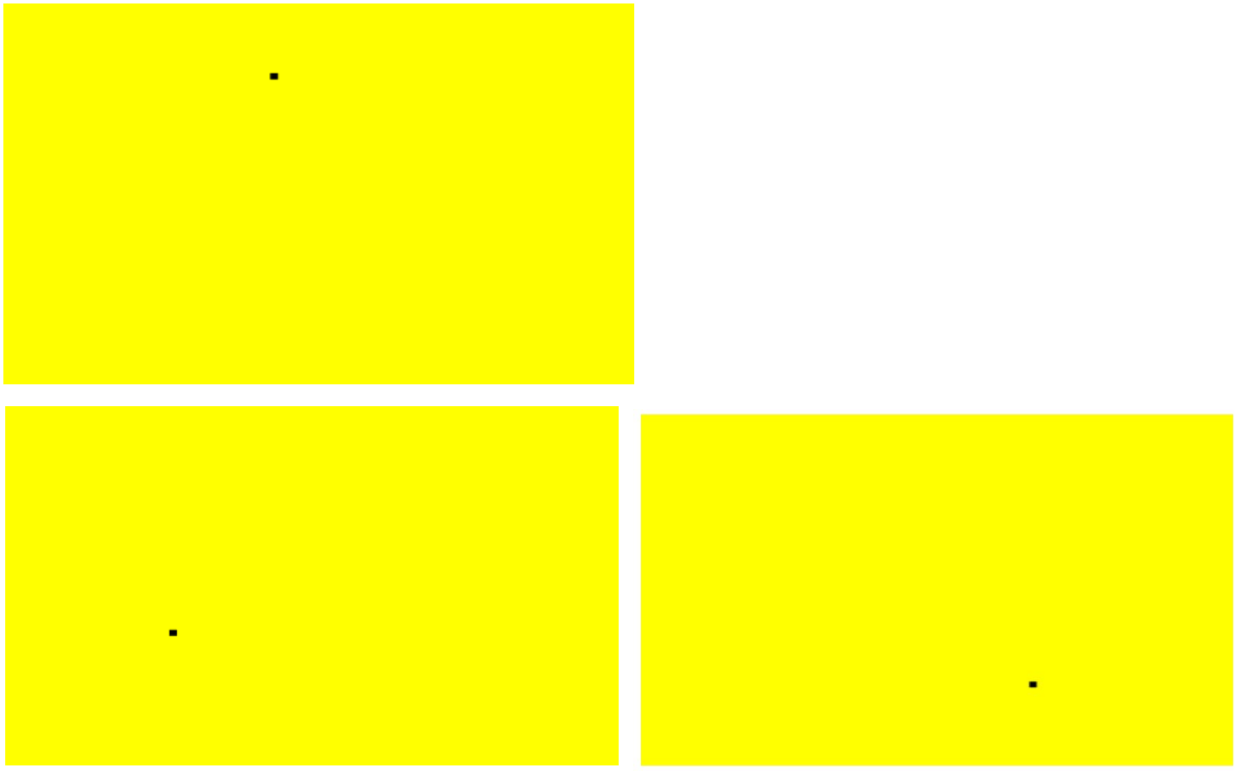
Lisaks mehhaanikas asukoha leidmiseks tarvilikule $x=x_0+v_0t+at^2/2$ ning kaugusearvutuse $\sqrt{x^2+y^2}$ arvutusele on kolmandaks tähtsaks ettetulevaks arvutustehteks ringjoone parameetriline võrrand: $x=x_0+r*\cos(a)$, $y=y_0+r*\sin(a)$. Selle abil saab lihtsamal juhul panna näiteks täpi mööda ekraani ringikujuliselt liikuma.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       var nurk=0;
       var nurgavahe=0.05; //radiaani
       function liigu(){
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
          g.fillRect(keskx+ringiraadius*Math.cos(nurk),
                     kesky-ringiraadius*Math.sin(nurk), 5, 5);
          nurk+=nurgavahe;
       }
     </script>
  </head>
  <body onload="setInterval('liigu()', 50);">
     <h1>Ringjoone parameetriline võrrand</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```

Ülesandeid

- Käivita näide
- Muuda liikumise keskkohta ja raadiust
- Pane ekraanil korraga ringikujuliselt liikuma kaks täppi

Ringjoone parameetriline võrrand

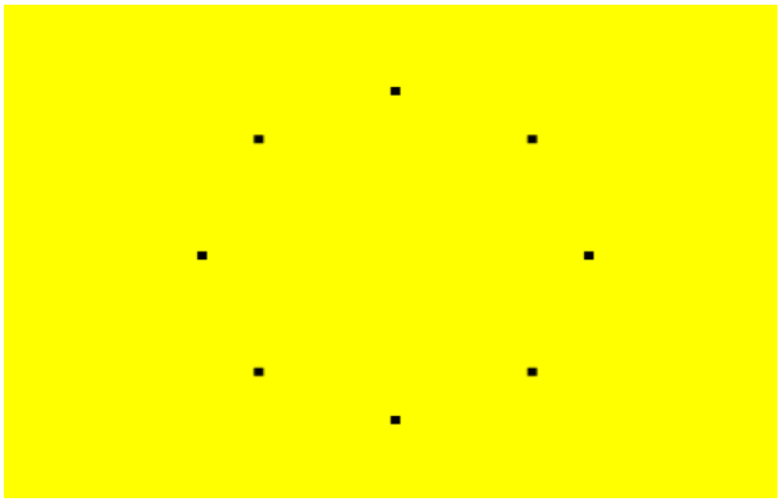


Ruudud ringjoonel

Kui ühekorraga ruudud valmis joonistada ja mitte andmeid muuta ja ekraani tühjendada, siis saab soovitud kujundid ilusti ringi peale paigutada.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       var nurk=0;
       var punktidearv=8;
       var nurgavahe=2*Math.PI/punktidearv;
       function joonista(){
          var g=document.getElementById("t1").getContext("2d");
          for(var i=0; i<punktidearv; i++){
            g.fillRect(keskx+ringiraadius*Math.cos(nurk),
                       kesky-ringiraadius*Math.sin(nurk), 5, 5);
            nurk+=nurgavahe;
          }
       }
     </script>
  </head>
  <body onload="joonista();">
     <h1>Ruudud ringjoonel</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```

Ruudud ringjoonel



Ülesandeid

- Käivita näide
- Koosta kella numbrilaud - ringikujuliselt arvud 1-st 12ni.
- Pane iga numbri juurde ka keskkoha poole suunduv joon.

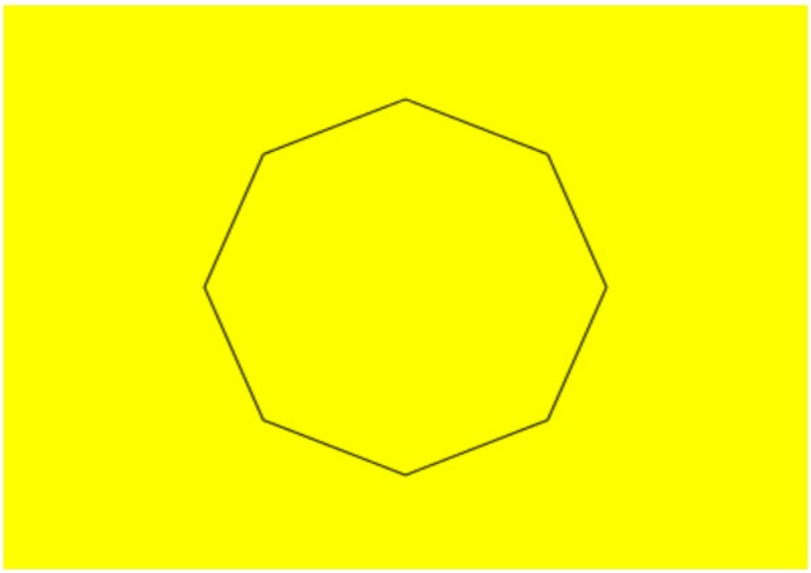
- Eelpool olevat ringjoonel liikumise näidet arvestades püüa tööle panna osutitega kell.

Hulknurk

Järgnevalt veidi pikem näide, kus üheaegselt tegeldakse mitme punkti keeramisega. Korrapärase hulknurga puhul tuleb teada nurkade arvu ning nende kaugust keskpunktist. Edasi saab selle põhjal juba jooned tõmmata. Esimesse punkti tasub minna moveTo-ga, edasi igasse järgmisse nurka saab lineTo abil joone tõmmata kui tahta ühtlaselt ühendatud hulknurka saada. Ning viimasest punktist sammu võrra edasi minnes jõutakse jälle algusesse tagasi.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       var nurk=0;
       var punktidearv=8;
       var nurgavahe=2*Math.PI/punktidearv;
       function joonista(){
          var g=document.getElementById("t1").getContext("2d");
          g.beginPath();
          g.moveTo(keskx+ringiraadius*Math.cos(nurk),
```

```
                    kesky-ringiraadius*Math.sin(nurk));
            nurk+=nurgavahe;
            for(var i=1; i<=punktidearv; i++){
              g.lineTo(keskx+ringiraadius*Math.cos(nurk),
                       kesky-ringiraadius*Math.sin(nurk));
              nurk+=nurgavahe;
            }
            g.stroke();
          }
      </script>
  </head>
  <body onload="joonista();">
      <h1>Ruudud ringjoonel</h1>
      <canvas id="t1" width="400" height="300"
        style="background-color: yellow" ></canvas>
  </body>
</html>
```



Ülesandeid

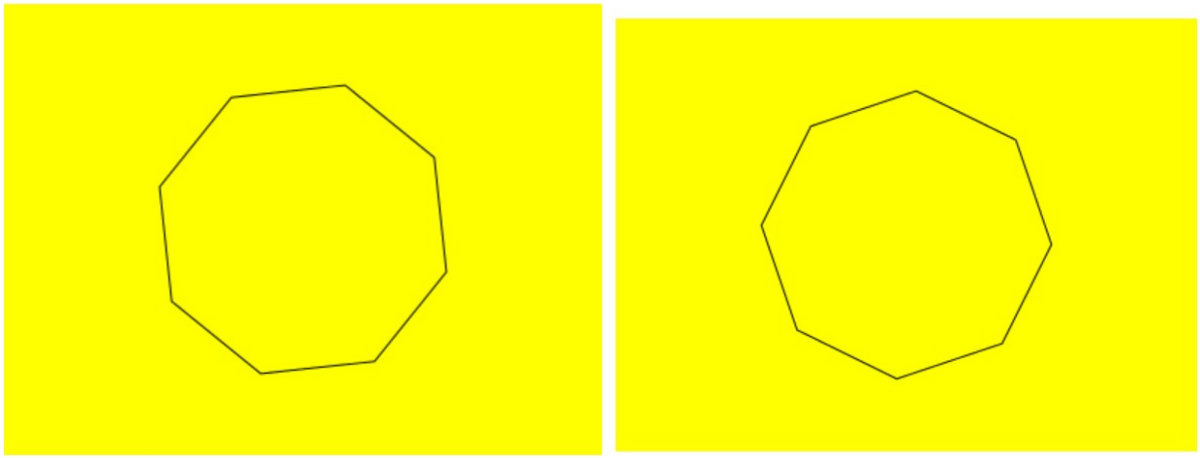
- Käivita näide
- Muuda nurkade arvu ja raadiust
- Loo lehele slaiderid raadiuse ja nurkade arvu sujuvaks muutmiseks

Keerlev hulknurk

Hulknurga ühtlaselt keerlema panekuks tuleb esimese punkti algusnurka sujuvalt muutma hakata. Kui teised esimese järgi arvutada, siis hakkabki nõnda kogu hulknurk keerlema.

```
<!doctype html>
<html>
  <head>
      <title>Ringi arvutused</title>
      <script>
        var keskx=200;
        var kesky=150;
```

```
      var ringiraadius=100;
      var algnurk=0;
      var keeramissamm=0.1;
      var punktidearv=8;
      var nurgavahe=2*Math.PI/punktidearv;

      function liigu(){
        algnurk+=keeramissamm;
        joonista();
      }

      function joonista(){
         var nurk=algnurk;
         var g=document.getElementById("t1").getContext("2d");
         g.clearRect(0, 0, 400, 300);
         g.beginPath();
         g.moveTo(keskx+ringiraadius*Math.cos(nurk),
                  kesky-ringiraadius*Math.sin(nurk));
         nurk+=nurgavahe;
         for(var i=1; i<=punktidearv; i++){
           g.lineTo(keskx+ringiraadius*Math.cos(nurk),
                    kesky-ringiraadius*Math.sin(nurk));
           nurk+=nurgavahe;
         }
         g.stroke();
      }
    </script>
  </head>
  <body onload="setInterval('liigu()', 100);">
    <h1>Ruudud ringjoonel</h1>
    <canvas id="t1" width="400" height="300"
      style="background-color: yellow" ></canvas>
  </body>
</html>
```



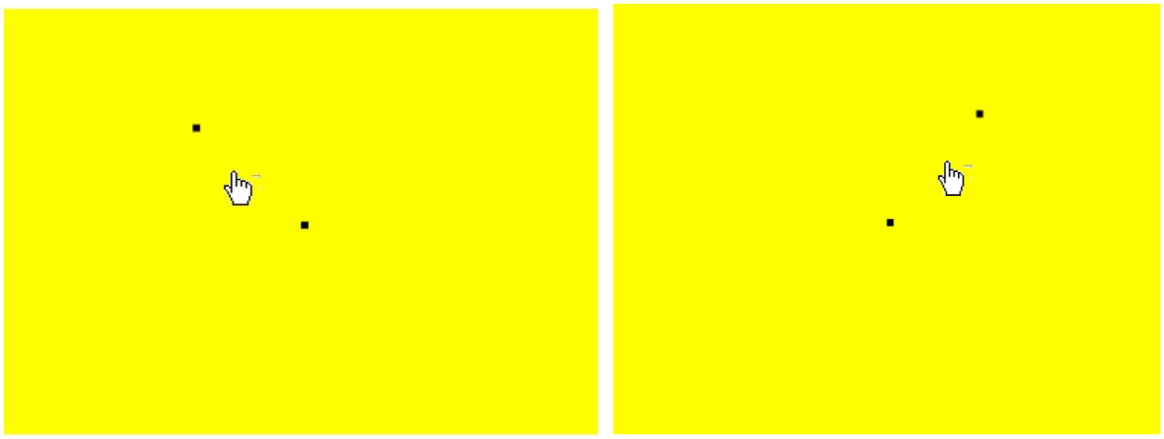
Ülesandeid

- Pane näide käima
- Võimalda hulknurga pöörlemiskiirust slaideri abil muuta.
- Võimalda ta ka teistpidi pöörlema panna

Nurga määramine

Ümmarguste asjade joonistamisel ning ringi ja kaart pidi liigutamisel on hea ringjoone parameetrilise võrrandi abil nurga, keskpunkti ja raadiuse järgi koordinaate leida. Hiirega rakendust juhtides aga tuleb mõnikord kasuks vastupidine - tuleb leida hiire asukohale vastav nurk mõne punkti suhtes, et saaks ekraanil kujundeid sobivalt pöörata ja liikuma panna. Õnneks on Javaskriptis selle tarbeks kohandatud arkustanensi funktsioon Math.atan2(x, y), mis annab etteantud koordinaatidega punkti paiknemise nurga nullpunkti suhtes. Nõnda on asimuudi leidmine lihtsaks tehtud, tuleb vaid rakendusele sobivad andmed ette sööta. Järgnevas näites näeb ekraani peal kahte punkti. Üks püsib paigal, teise paiknemise suunda selle keskpunkti suhtes saab aga hiirega määrata. Hiire asukoha ja keskpunkti kauguse põhjal leitakse keskpunktist hiire poole suunduv nurk. Sinnapoole püsiva raadiuse kaugusele joonistatakse ringjoone parameetrilise võrrandi abil täpp. Nii saabki hiire abil praegu ühte täppi, aga soovi korral ka tunduvalt keerukamat kujundit ümber keskpunkti pöörata.

```
<!doctype html>
<html>
  <head>
     <title>Nurga leidmine</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       function hiirLiigub(e){
          var tahvlikoht=document.getElementById("t1").
               getBoundingClientRect();
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
          g.fillRect(keskx, kesky, 5, 5);
          var hx=e.clientX-tahvlikoht.left;
          var hy=e.clientY-tahvlikoht.top;
          //g.fillRect(hx, hy, 5, 5);
          var nurk=Math.atan2(hy-kesky, hx-keskx);
          g.fillRect(keskx+ringiraadius*Math.cos(nurk),
                     kesky+ringiraadius*Math.sin(nurk), 5, 5);

       }
     </script>
  </head>
  <body>
     <h1>Ruudud ringjoonel</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow"
       onmousemove="hiirLiigub(event)"></canvas>
  </body>
</html>
```



Ülesandeid

- Pane näide käima
- Kujunda ekraanile nupp, mida on võimalik koos sellel oleva kriipsuga hiire abil pöörata
- Pane selliseid nuppe ekraanile kaks. Kummagi nupu piires saab just selle nupu pöördenurka sättida.

Rool

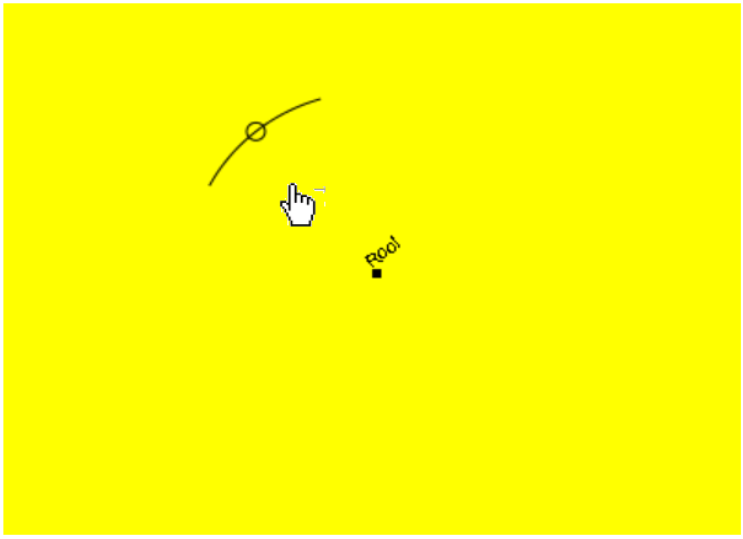
Täpi keeramise edasiarenduseks on lihtsa rooli keeramine. Hiire asukohast leitud nurga järgi joonistatakse kaar punktist mõlemale poole. Ringjoone parameetrilise võrrandi abil saab ümmarguse täpi kaare keskele soovitud suunda. Teksti pööramiseks rooli keskosas sobib aga reeperi /koordinaatteljestiku nihutus ja pööre. Salvestuskäsklusega talletatakse algne olek, et selle juurde saaks tagasi minna. Edasi translate nihutab nullpunkti rooli keskkoha juurde. Siis on võimlik tekst selle koha juures sobiva nurga alla keerata ja ekraanile joonistada. Ning lõpuks restore-käsklus taastab endise olukorra, et muud joonistuskäsud jälle arusaadavatesse kohtadesse läheksid.

```
g.save();
g.translate(keskx, kesky);
g.rotate(nurk+(Math.PI/2));
g.fillText("Rool", 0, 0);
g.restore();
```

Edasi juba töötav kood tervikuna.

```
<!doctype html>
<html>
  <head>
     <title>Nurga leidmine</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       function hiirLiigub(e){
          var tahvlikoht=document.getElementById("t1").
               getBoundingClientRect();
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
```

```
            g.fillRect(keskx, kesky, 5, 5);
            var hx=e.clientX-tahvlikoht.left;
            var hy=e.clientY-tahvlikoht.top;
            var nurk=Math.atan2(hy-kesky, hx-keskx);
            g.beginPath();
            g.arc(keskx, kesky, ringiraadius, nurk-0.4, nurk+0.4, false);
            g.stroke();
            g.beginPath();
            g.arc(keskx+ringiraadius*Math.cos(nurk),
                        kesky+ringiraadius*Math.sin(nurk), 5, 0, 2*Math.PI, false);
            g.stroke();
            g.save();
            g.translate(keskx, kesky);
            g.rotate(nurk+(Math.PI/2));
            g.fillText("Rool", 0, 0);
            g.restore();
         }
      </script>
   </head>
   <body onload="setInterval('liigu()', 100);">
      <h1>Ruudud ringjoonel</h1>
      <canvas id="t1" width="400" height="300"
        style="background-color: yellow"
        onmousemove="hiirLiigub(event)"></canvas>
   </body>
</html>
```



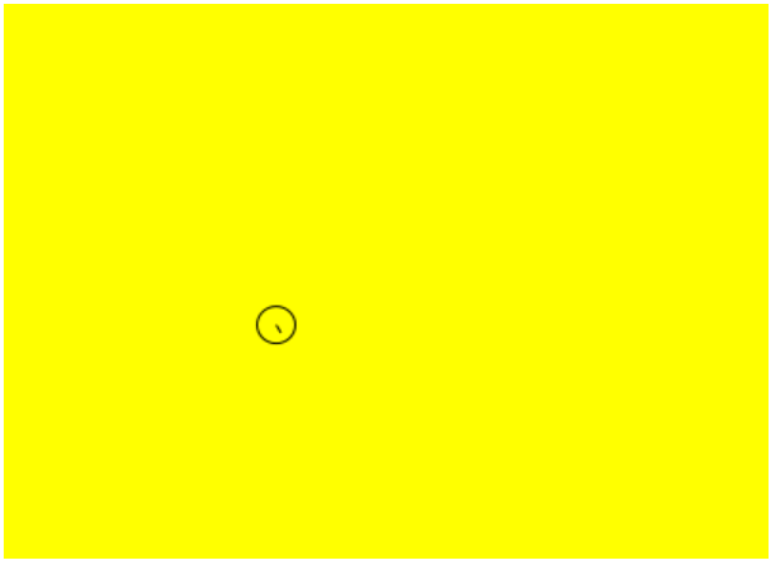
Ülesandeid

- Pane näide käima
- Asenda rooliratta suunda tähistav ringike sobiva nurga alla keeratud R-tähega

Liikur

Tavapäraselt kipume arvutigraafikas eraldi x- ja y-suunda arvestama ja meeles pidama. Samas nurga all liikumise ja pööramiste puhul on mõnigikord mugavam meeles pidada pigem asukohta ning liikumissuuna nurka. Edasised arvutused saab juba nende andmete põhjal teha. Praegusel juhul arvutatakse iga sammu puhul siinus ka koosiinus uuesti, mis üsnagi töömahukad arvutile. Koodi kiirust võimalik siitkaudu kasvatada, aga ka nii paistab liikume täiesti õnnestuma.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>

       function Liikur(x, y, kiirus, nurk){
         this.algus=function(){
          this.x=x;
          this.y=y;
          this.kiirus=kiirus;
          this.nurk=nurk;
         }
         this.liigu=function(){
           this.x+=this.kiirus*Math.cos(this.nurk);
           this.y+=this.kiirus*Math.sin(this.nurk);
         }
         this.muudaNurk=function(uusNurk){
            this.nurk=uusNurk;
         }
         this.joonista=function(g){
            g.beginPath();
            g.arc(this.x, this.y, 10, 0, 2*Math.PI, false);
            g.stroke();
            g.beginPath();
            g.moveTo(this.x,this.y);
            g.lineTo(this.x+5*this.kiirus*Math.cos(this.nurk),
                     this.y+5*this.kiirus*Math.sin(this.nurk));
            g.stroke();
         }
         this.algus();
       }

       var auto1=new Liikur(100, 100, 1, Math.PI/3);
       function liigu(){
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
          auto1.liigu();
          auto1.joonista(g);
       }
     </script>
  </head>
  <body onload="setInterval('liigu()', 50);">
     <h1>Ringjoone parameetriline võrrand</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```



Ülesandeid

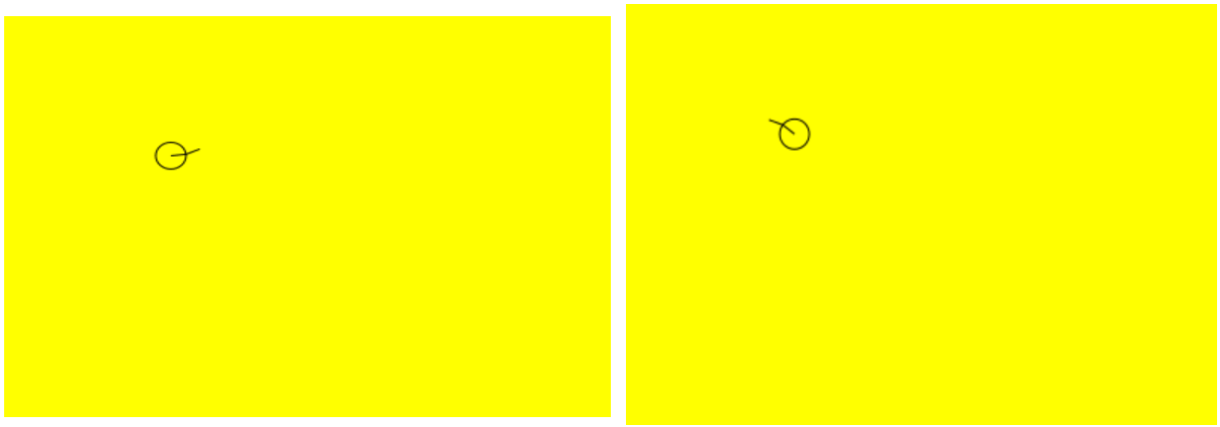
- Pane näide käima
- Lisa ekraanile nupud liikuri keeramiseks päri- ja vastupäeva
- Pane korraga liikuma mitu liikurit
- Joonista joone liikumissuunda näitavasse otsa väike ringike

Keerav liikur

Auto juhtmist jäljendades rooli pööramine ei tähenda mitte korraks sihi muutust, vaid pööratud rool paneb edasisõidu ajal auto pidevalt vastavas suunas pöörama. Sarnast tulemust püütakse ka siin saavutada. Ehk siis kui liikur sõitma pannakse, siis määratakse talle lisaks asukohale, kiirusele ja algnurgale ka nurgamuutus, et kui palju iga sammuga liikur uuesti pöörab. Selliselt tehtud liikur sõidab siis vähegi otsesihist kõrvale kallutatuna mitmesuguse suurusega ringe vastavalt etteantud nurgamuutusele.
Pööramissuuna ja -ulatuse paremaks jälgimiseks on sihtjoone otsas kümne ekraanipunkti suurune juhtlõik, mis selgema nägemise huvides kolmekordse võimendusega annab teada, et kui palju ja millises suunas pööratakse. Pööramisjoone arvutamiseks võetakse liikumisjoone ots ning sealt edasi siis arvutatakse vajaliku nurga ja pööramisjoone pikkuse järgi uued koordinaadid.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>

       function Liikur(x, y, kiirus, nurk, nurgamuutus){
         this.algus=function(){
          this.x=x;
          this.y=y;
          this.kiirus=kiirus;
          this.nurk=nurk;
          this.nurgamuutus=nurgamuutus;
         }
         this.liigu=function(){
           this.nurk+=this.nurgamuutus;
           this.x+=this.kiirus*Math.cos(this.nurk);
           this.y+=this.kiirus*Math.sin(this.nurk);
```

```
        }
        this.muudaNurk=function(uusNurk){
           this.nurk=uusNurk;
        }
        this.joonista=function(g){
           g.beginPath();
           g.arc(this.x, this.y, 10, 0, 2*Math.PI, false);
           g.stroke();
           g.beginPath();
           g.moveTo(this.x,this.y);
           //Joone ots
           var jotsx=this.x+10*this.kiirus*Math.cos(this.nurk);
           var jotsy=this.y+10*this.kiirus*Math.sin(this.nurk);
           g.lineTo(jotsx, jotsy);

           var rattapikkus=10;
           var rattanurk=this.nurk+3*this.nurgamuutus;
           var rattaotsx=jotsx+rattapikkus*Math.cos(rattanurk);
           var rattaotsy=jotsy+rattapikkus*Math.sin(rattanurk);
           g.lineTo(rattaotsx, rattaotsy)
           g.stroke();
        }
        this.algus();
      }

      var auto1=new Liikur(100, 100, 1, Math.PI/3, -0.1);
      function liigu(){
         var g=document.getElementById("t1").getContext("2d");
         g.clearRect(0, 0, 400, 300);
         auto1.liigu();
         auto1.joonista(g);
      }
    </script>
  </head>
  <body onload="setInterval('liigu()', 50);">
    <h1>Ringjoone parameetriline võrrand</h1>
    <canvas id="t1" width="400" height="300"
      style="background-color: yellow" ></canvas>
  </body>
</html>
```



Ülesandeid

- Pane näide tööle
- Lisa pööramisnurga suurendamiseks ja vähendamiseks nupud

Rool ja liikur

Autosõidu puhul saab pidevalt sõiduki pööramist muuta. Nii ka siin püütakse nüüd kaks eraldi loodud lahendust kokku ühendada. Rooli juurest saab küsida, et kui palju peaks keerama ning liikuri ülesandeks on siis need andmed vastu võtta ning nende põhjal sujuvalt mööda ekraani liikuda. Suuremalt jaolt saab Rooli ja Liikuri tüübid kokku kopeerida. Ning lihtsalt liikumise juures siis iga sammu puhul küsida rooli käest selle pööratuse nurk ning määrata see autole uueks rooli asendiks ehk nurgamuutuseks iga sammu juures.

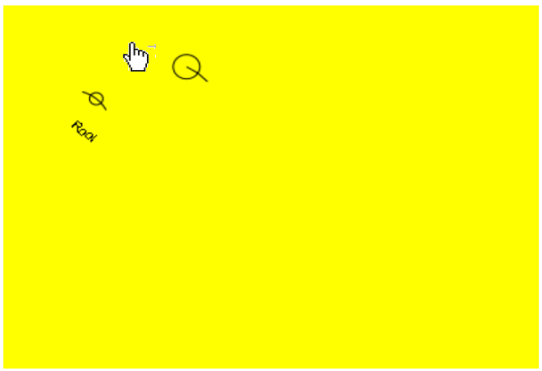
```
auto1.uusRooliAsend(rool1.kysiNurk()/10);
```

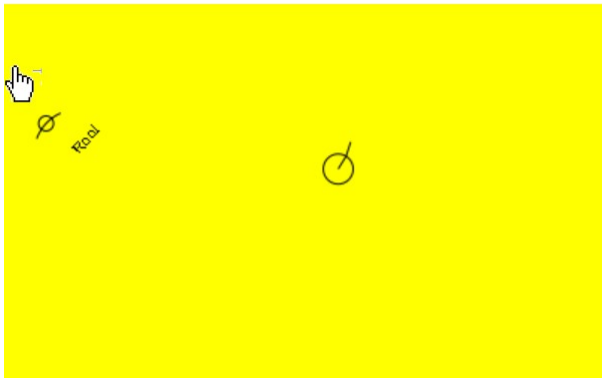
Katseliselt selgus, et suhteliselt mugav oli autot juhtida praeguste parameetrite järgi siis, kui keeramine sammu juures oli kümnendik rooli tegelikust pöördenurgast. Aga eks selliste suhete paika sättimine olegi rakenduse loomise juures mugavuse tekitamise ja katsetamise küsimus.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>

       function Liikur(x, y, kiirus, nurk, nurgamuutus){
         this.algus=function(){
          this.x=x;
          this.y=y;
          this.kiirus=kiirus;
          this.nurk=nurk;
          this.nurgamuutus=nurgamuutus;
         }
         this.liigu=function(){
           this.nurk+=this.nurgamuutus;
           this.x+=this.kiirus*Math.cos(this.nurk);
           this.y+=this.kiirus*Math.sin(this.nurk);
         }
         this.muudaNurk=function(uusNurk){
            this.nurk=uusNurk;
         }
         this.uusRooliAsend=function(uusAsend){
            this.nurgamuutus=uusAsend;
         }
         this.joonista=function(g){
            g.beginPath();
            g.arc(this.x, this.y, 10, 0, 2*Math.PI, false);
            g.stroke();
            g.beginPath();
            g.moveTo(this.x,this.y);
            //Joone ots
            var jotsx=this.x+10*this.kiirus*Math.cos(this.nurk);
            var jotsy=this.y+10*this.kiirus*Math.sin(this.nurk);
            g.lineTo(jotsx, jotsy);

            var rattapikkus=10;
            var rattanurk=this.nurk+3*this.nurgamuutus;
            var rattaotsx=jotsx+rattapikkus*Math.cos(rattanurk);
            var rattaotsy=jotsy+rattapikkus*Math.sin(rattanurk);
            g.lineTo(rattaotsx, rattaotsy)
            g.stroke();
         }
```

```
        this.algus();
      }

      function Rool(keskx, kesky, raadius, nurk){
         this.algus=function(){
            this.keskx=keskx;
            this.kesky=kesky;
            this.raadius=raadius;
            this.nurk=nurk;
         }
         this.kysiNurk=function(){
            return this.nurk+Math.PI/2;
         }
         this.joonista=function(g){
             g.beginPath();
             g.arc(this.keskx, this.kesky, this.raadius,
                  this.nurk-0.4, this.nurk+0.4, false);
             g.stroke();
             g.beginPath();
             g.arc(this.keskx+this.raadius*Math.cos(this.nurk),
                   this.kesky+this.raadius*Math.sin(this.nurk),
                     5, 0, 2*Math.PI, false);
             g.stroke();
            g.save();
            g.translate(this.keskx, this.kesky);
            g.rotate(this.nurk+(Math.PI/2));
            g.fillText("Rool", 0, 0);
            g.restore();
         }
         this.uusNurk=function(hx, hy){
            this.nurk=Math.atan2(hy-kesky, hx-keskx);
         }
         this.algus();
      }

      var auto1=new Liikur(100, 100, 1, Math.PI/3, -0.1);
      var rool1=new Rool(50, 100, 30, -Math.PI/2);

      function hiirLiigub(e){
         var tahvlikoht=document.getElementById("t1").getBoundingClientRect();
         var hx=e.clientX-tahvlikoht.left;
         var hy=e.clientY-tahvlikoht.top;
         rool1.uusNurk(hx, hy);
      }

      function liigu(){
         var g=document.getElementById("t1").getContext("2d");
         g.clearRect(0, 0, 400, 300);
         auto1.uusRooliAsend(rool1.kysiNurk()/10);
         auto1.liigu();
         auto1.joonista(g);
         rool1.joonista(g);
      }
    </script>
  </head>
  <body onload="setInterval('liigu()', 50);">
    <h1>Ringjoone parameetriline võrrand</h1>
    <canvas id="t1" width="400" height="300"
      style="background-color: yellow"
      onmousemove="hiirLiigub(event)"></canvas>
  </body>
</html>
```

Jooniselt võib näha, kuidas siis saab rooli abil liikurit mitut moodi sättida.





Ülesandeid

- Pane näide käima
- Muuda pööramise suurust vastavalt rooli asendile.

Ringrajasõit

- Pane auto sõitma ümber täpi, loe mitu ringi on läbitud
- Ringi kraadide järgi saab määrata "väravad", kui kaugelt mingi nurk tuleb läbida. Väravad on näha joonisel ning nende õiget läbimist kontrollitakse ringide kohta punktide arvutamisel.