Kapittel 23

Prosjekt: snake

Skrevet av **Inger Elisabeth Grenborgen**.

23.1 Planlegging

pssst

Husk nettstedet på Lokus.

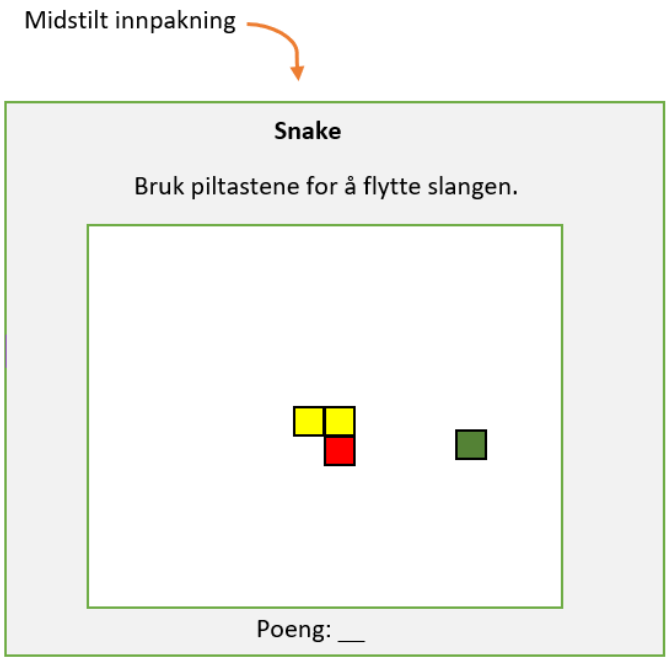
Dette prosjektet vil her bli presentert ganske kortfattet, men kan utvides med bedre grafikk og flere funksjoner hvis du setter deg godt inn i koden. Pass på å lese beskrivelsen mellom kodesnuttene så du vet hvor i koden du skal legge til hva. Koden blir ikke alltid presentert kronologisk, så du kan ikke skrive den inn fortløpende og forvente at det fungerer. Pass også på å teste koden underveis.

Kravspesifikasjon

En enkel kravspesifikasjon for snake-spillet kan se slik ut:

- Det skal være en slange som hele tiden beveger seg i samme retning hvis den ikke får beskjed om noe annet.
- Brukeren skal kunne endre retningen slangen beveger seg i ved å trykke på piltastene.
- Det skal dukke opp frukt på tilfeldige steder som slangen kan spise.
- Hvis slangen «spiser» en frukt, så skal den vokse og bli lengre, og poengsummen skal øke.
- Spillet er over hvis slangen treffer en vegg.
- Spillet er over hvis slangen treffer seg selv.

Skisse (wireframe)



Figur 23.1: Skisse av spillets utseende.

Pseudokode

I denne appen skal det skje mye, så det er viktig å ha en plan for hva som skal skje før du begynner å skrive koden. Siden vi ønsker å kunne legge til funksjonalitet for å restarte spillet med en knapp ved «game over», lønner det seg å legge all initiering av variabler i en startfunksjon. Gjør du dette, er det bare å kalle denne funksjonen hvis du vil restarte spillet. Det betyr igjen at vi må opprette alle variablene utenfor denne startfunksjonen slik at hele programmet har tilgang til disse variablene.

Det er flere måter å skrive dette programmet på. Her har vi valgt å bruke et canvas på 400 px × 400 px til å tegne opp spillbrett, slange og frukt. Slangen skal bestå av kvadrater på 20 px × 20 px og koordinatene til alle slangeleddene lagrer vi i en array. På den første plassen i arrayen skal koordinatet til hodet ligge. Vi har også en array for å holde orden på plasseringen til frukten, så det skal være lett å legge til mange frukter hvis du vil det.

Kode utenfor funksjonene

```
Opprett alle spillvariablene, men vent med å gi dem verdi
startSpill()
```

Funksjoner

```
FUNKSJON startSpill
   Initier alle spillvariablene (gi verdier til alle spillvariablene)
   Opprett en timer som kaller funksjonen flytt
   Opprett en lytter på tastaturet som kaller funksjonen byttRetning
   leggTilFrukt()  // Kalles like mange ganger som man ønsker det skal være frukter
   tegnOpp()


FUNKSJON leggTilFrukt
    Lag en tilfeldig x- og y- verdi
    WHILE (frukten er plassert på hodet eller på kroppen til slangen)
       Lag en ny tilfeldig x- og y- verdi
    Opprett og tegn frukt


FUNKSJON tegnOpp
    Tøm canvaset
    Tegn opp hodet og kroppen til slangen
    Tegn opp frukten på oppgitt x- og y-koordinat


FUNKSJON flytt
   (Kalles med et tidsbestemt mellomrom, men også når brukeren trykker på tastaturet.)
   Flytt slangen
   IF retningen er høyre
      IF det er plass til å gå til høyre
         Øk x-verdien
         tegnOpp()
   ELSE IF retningen er venstre
      IF det er plass til å gå til venstre
         Reduser x-verdien
         tegnOpp()
   ELSE IF retningen er opp
      IF det er plass til å gå opp
         Reduser y-verdien
         tegnOpp()
   ELSE IF retningen er ned
      IF det er plass til å gå ned
         Øk y-verdien
         tegnOpp()
   IF hodet treffer kroppen
      gameOver
```

```
   IF hodet treffer frukten
      Fjern frukt
      Øk poeng
      Plasser ut frukten på nytt


FUNKSJON byttRetning
   (Brukeren har trykket en tast på tastaturet og ønsker å endre retning på slangen.)
   Fjern timeren
   Opprett timeren på nytt
   flytt()


FUNKSJON slangeHit(x,y)  // Returnerer en boolsk variabel (true eller false)
   Sjekk om en gitt x- og y-verdi kolliderer med slangen


FUNKSJON gameOver
   Fjern timer
   Fjern lytter på tastaturet
   Vis en "game over"-melding til brukeren
   Lag en knapp som kaller startSpill
```

23.2 Koding

Aller først starter vi med å lage rammen til spillet, slik at HTML og CSS er på plass før vi starter med JavaScript-koden.

HTML

Nedenfor er hele HTML-dokumentet. Vi har et `<div>`-element med `id="innpakning"` som inneholder spillet. Deretter har vi en overskrift, et `<canvas>`-element og et `<p>`-element der vi skal skrive ut poengene brukeren får.

```
<!doctype html>
<html>
<head>
  <title>Snake</title>
  <meta charset="UTF-8">
  <style>
    /* Her skal vi skrive CSS-koden */
  </style>
</head>
<body>
  <div id="innpakning">
    <h1>Snake</h1>
    Bruk piltastene for å flytte slangen.
```

```
14      <canvas id="mittCanvas" width="400" height="400">
15        Nettleseren din støtter ikke canvas-elementet
16      </canvas>
17      <p id="poeng"> Poeng: </p>
18    </div>
19    <script>
20      /* Her skal vi skrive JavaScript-koden */
21    </script>
22  </body>
23  </html>
```

Kode 23.1: HTML-koden vi bruker i spillet.

CSS

I CSS-koden setter vi en svak farge på bakgrunnen slik at `<canvas>`-elementet kommer tydelig frem og vi midtstiller teksten.

```
body {
  background-color: #e0e0e0;
  text-align: center;
}
```

Innpakningen settes til å være 50 px bredere enn `<canvas>`-elementet og får posisjonen «relative» slik at elementer inni innpakningen posisjoneres i forhold til innpakningen. Med `margin: auto;` blir innpakningen midtstilt.

```
#innpakning {
  position: relative;
  width: 450px;
  margin: auto;
}
```

Vi lar `<canvas>`-elementet få en liten ramme og en hvit bakgrunnsfarge, så den skiller seg fra innpakningen. For å midtstille `<canvas>`-elementet i innpakningen bruker vi `margin: auto;`.

```
#mittCanvas {
  border: 2px solid #666666;
  background-color: #ffffff;
  display: block;
  margin: auto;
}
```

JavaScript

Da har vi både CSS og HTML på plass, så nå gjenstår JavaScript-koden. Her er det viktig å lese instruksjonen nøye før du skriver inn koden, den vil

ikke alltid presenteres i kronologisk rekkefølge. Det første vi skal gjøre er å få tegnet opp slangen og få den til å bevege seg ved hjelp av piltastene. For å få til det, må vi opprette alle variablene vi trenger, og vi må skrive funksjonene `startSpill()`, `flytt()`, `byttRetning()` og deler av funksjonen `tegnOpp()`. Vi starter med alle variablene som skal opprettes utenfor funksjonene.

```
var bodyEl = document.querySelector("body");
var poengEl = document.querySelector("#poeng");
var canvas = document.querySelector("#mittCanvas");
var ctx = canvas.getContext("2d");

var x;          // x-koordinaten til slangens hode
var y;          // y-koordinaten til slangens hode
var size = 20;  // Størrelse slange og frukte i piksler
var maksLengde; // Største lengden slangen kan ha
var vekst;      // Hvor fort slangen skal vokse
var slange;     // Array med alle slangeleddene
var frukt;      // Array med frukt
var poeng;      // Tar vare på poengene
var retning;    // Tar vare på retningen slangen beveger seg i
var hastighet;  // Hvor fort slangen skal bevege seg i ms
var timer;      // Lytter på tiden

startSpill();
```

Kode 23.2: Variablene vi bruker i spillet.

Når alle spillvariablene er opprettet, kaller vi på funksjonen `startSpill()` som skal sette startverdiene til alle variablene.

```
function startSpill() {
  x = 0;                // Startkoordinater til slangen
  y = 0;
  maksLengde = 2;       // Startlengden til slangen
  vekst = 2;            // Hvor mange ledd slangen skal vokse med
  slange = new Array(); // En array for slangeleddene
  frukt = new Array();  // En array for frukten
  poeng = 0;
  retning = 40;         // Retning nedover (keyCode = 40)
  hastighet = 300;      // Slangens hastighet (i ms).
  // Hastigheten styrer spillets vanskelighetsgrad.

  poengEl.innerHTML = "Poeng: " + poeng;

  //Legger til en keydown-hendelse på body-elementet
  bodyEl.addEventListener("keydown", byttRetning);

  // TO DO: Legg til en timer
  // TO DO: Legg til frukt

  tegnOpp();
}
```

Kode 23.3: Begynnelsen på funksjonen `startSpill()`.

I funksjonen `startSpill()` oppretter vi en lytter på `<body>`-elementet som lytter etter om brukeren trykker på tastaturet. I slutten av funksjonen kaller vi funksjonen `tegnOpp()` som skal tegne elementene i `<canvas>`-elementet. I denne funksjonen skal vi også legge til en timer som jevnlig passer på at slangen flytter seg. Vi må også lage frukter her, men det tar vi senere.

Koden til funksjonen `tegnOpp()` ser du nedenfor. Først blankes canvaset ut, deretter tegner vi en rød firkant på koordinatet $(x\ ,\ y)$. Dette representerer hodet til slangen (vi initierte variablene `x` og `y` i funksjonen `startSpill()` ovenfor). Etter å ha tegnet hodet har vi en `for`-løkke som går igjennom alle slangeleddene som ligger i arrayen `slange`, og tegner opp gule firkanter på disse koordinatene. Koden for å tegne frukten tar vi senere.

```
function tegnOpp() {
  ctx.clearRect(0, 0, 400, 400); // blanker ut canvaset
  ctx.fillStyle = "red";
  ctx.fillRect(x, y, size, size);
  ctx.fillStyle = "yellow";

  // tegner alle slangeleddene
  for(var i = 0; i < slange.length; i++) {
    ctx.fillRect(slange[i].xPos, slange[i].yPos, size, size);
  }

  // TO DO: Tegn opp frukten
}
```

Kode 23.4: Begynnelsen på funksjonen `tegnOpp()`.

Vi har en tastaturhendelse som kaller funksjonen `byttRetning()`, og her trenger vi å finne keyCoden til tasten brukeren har trykket. Kun piltastene er gyldige knapper, så derfor sjekker vi om keyCode ligger mellom 36 og 41 (piltastene har keyCodene 37, 38, 39 og 40). For å finne hvilken knapp brukeren har trykket på, bruker vi hendelsesobjektet, `e`.

```
function byttRetning(e) {
  if (e.keyCode > 36 && e.keyCode < 41) {
    retning = e.keyCode;
    flytt();
  }
}
```

Kode 23.5: Funksjonen `byttRetning()`.

Funksjonen `flytt()` skal flytte hodet til slangen hver gang brukeren trykker på en piltast, eller tidsintervallet tilsier at nå må slangen flytte seg. En første utgave av funksjonen flytt vil se slik ut:

```
function flytt() {
  // oppretter et slangeledd der hodet var
  var slangeLedd = { xPos: x, yPos: y };
```

```
  // legg til slangeLedd først i arrayen
  slange.unshift(slangeLedd);

  // hvis slangen er lengre enn maks lengde
  if (slange.length > maksLengde) {
    // fjern det siste slangeleddet i arrayen
    slange.pop();
  }

  if (retning == 37) { // retningen er venstre
      x -= size; // reduser x-koordinaten med 20px
      tegnOpp();
  } else if (retning == 39) { // retningen er høyre
      x += size; // øk x-koordinaten med 20px
      tegnOpp();
  } else if (retning == 38) { // retningen er oppover
      y -= size; // reduser y-koordinaten med 20px
      tegnOpp();
  } else if (retning == 40) { // retningen er nedover
      y += size; // øk y-koordinaten med 20px
      tegnOpp();
  }
}
```

Kode 23.6: Funksjonen `flytt()`.

Nå har vi et kjørbart program. Pass på at du tester at det fungerer før du går videre med koden.

Slangen kan nå styres med piltastene, men den vil forsvinne ut av spillbrettet hvis vi ikke endrer retning med piltastene ofte. For å hindre dette må vi legge til noen begrensninger. Når vi går mot venstre, må vi sjekke om det er plass til å gå mot venstre. Dette kan vi gjøre ved å teste om x-koordinaten til hodet minus størrelsen til slangeleddet er større eller lik 0, altså `if (x - size >= 0)`. Hvis du har gjort det riktig så langt, skal slangen ikke kunne forsvinne ut på den venstre siden av spillbrettet. Nå må vi gjøre det samme med de andre sidene. På høyre side må vi teste om x-koordinaten pluss størrelsen til slangeleddet er mindre enn 400, altså `if (x + size < 400)`. Når vi går oppover må vi teste om y-koordinaten minus størrelsen til slangeleddet er større eller lik 0, altså `if (y - size >= 0)`. Når vi går nedover må vi teste om y-koordinaten pluss slangeleddet er mindre enn 400, altså `if (y + size < 400)`. Disse testene må legges til etter at vi har sjekket retningen til slangen. Igjen er det viktig å teste spillet for å se om vi har fått det til.

```
function flytt() {
  // oppretter et slangeledd der hodet var
  var slangeLedd = { xPos: x, yPos: y };
  // legg til slangeLedd først i arrayen
  slange.unshift(slangeLedd);

  // hvis slangen er lengre enn maks lengde
  if (slange.length > maksLengde) {
```

```
    // fjern det siste slangeleddet i arrayen
    slange.pop();
  }

  if (retning == 37) {  // retningen er venstre
    // hvis det er plass til å gå til venstre
    if (x - size >= 0) {
      x -= size; // reduser x-koordinaten med 20px
      tegnOpp();
    }
  } else if (retning == 39) { // retningen er høyre
    // hvis det er plass til å gå til høyre
    if (x + size < 400) {
      x += size; // øk x-koordinaten med 20px
      tegnOpp();
    }
  } else if (retning == 38) { // retningen er oppover
    // hvis det er plass til å gå oppover
    if (y - size >= 0) {
      y -= size; // reduser y-koordinaten med 20px
      tegnOpp();
    }
  } else if (retning == 40) { // retningen er nedover
    // hvis det er plass til å gå nedover
    if (y + size < 400) {
      y += size; // øk y-koordinaten med 20px
      tegnOpp();
    }
  }
}
```

Kode 23.7: Utvidet versjon av funksjonen `flytt()`.

Når vi nå har fått til å begrense slangen til å være på spillbrettet, kan vi gå løs på å sjekke om slangen treffer seg selv. For å teste dette skikkelig bør slangen vår være lengre enn det den er nå. Da kan vi midlertidig endre slangens `maksLengde` i `startSpill()`-funksjonen til 12, slik at vi kan få testet ut dette. Nederst i funksjonen `flytt()` legger vi til følgende kode for å sjekke om hodet på slangen treffer kroppen:

```
// sjekk om koordinatene til hodet treffer koordinatene til
   kroppen
if (slangeHit(x, y)) {
  gameOver(); // hodet traff kroppen - avslutt spillet
}
```

Kode 23.8: Bruk av funksjonen `slangeHit()`.

Det neste blir så å skrive kodene for funksjonene `slangeHit(xPos, yPos)` og `gameOver()`. Funksjonen `gameOver()` har vi valgt å gjøre veldig enkel her, men den kan og bør utvides når du har fått alt det andre til å fungere. I `gameOver()`-funksjonen er det viktig å fjerne lytteren på tastaturet.

```
function gameOver() {
  // fjerner keydown-hendelse på body-elementet
  bodyEl.removeEventListener("keydown", byttRetning);

  console.log("Game over!");

  // TO DO: Lag en gameover beskjed til brukeren
  // TO DO: Restart spillet
}
```

Kode 23.9: Funksjonen `gameOver()`.

I funksjonen `slangeHit(xPos, yPos)` tar vi koordinatene `xPos` og `yPos` som vi får tilsendt, og sjekker om de er det samme som koordinatene til et av slangeleddene. Hvis de er det har vi en kollisjon og funksjonen returnerer `true`.

```
function slangeHit(xPos, yPos) {
  // går gjennom alle slangeleddene
  for (var i = 0; i < slange.length; i++) {
    // sjekk om hodet (xPos, yPos) treffer en del av slangen
    if (xPos == slange[i].xPos && yPos == slange[i].yPos) {
      return true; // koordinatene kolliderer med slangen
    }
  }
  return false;
}
```

Kode 23.10: Funksjonen `slangeHit()`.

Igjen er det på tide å teste koden. Nå skal du få opp beskjeden «Game over!» i konsollen hvis du krasjer i veggene eller slangehodet treffer kroppen. Grunnen til at du får denne beskjeden hvis du krasjer i veggene er fordi det da er blitt opprettet et slangeledd der hodet er. Siden hodet ikke får lov til å flytte ut av spillbrettet, blir dette sett på som en kollisjon.

Det neste vi skal gjøre nå er å legge ut frukt som slangen kan spise. Da skal vi opprette en funksjon som vi kaller `leggTilFrukt()` som skal legge ut en frukt på et tilfeldig sted på spillbrettet. Dette stedet kan ikke være oppå slangen, så vi må sjekke at fruktens plassering ikke kolliderer med slangehodet eller slangekroppen. For å få en tilfeldig plassering bruker vi `Math.random()`. Fordi det er plass til 400 px/20 px = 20 slangeledd bortover spillbrettet, så ganger vi verdien fra `Math.random()` med 20 og stryker desimalene med `Math.floor()`, slik at vi får tall fra og med 0 til og med 19. Dette tallet ganger vi så med 20 slik at vi står igjen med de mulige tallene 0, 20, 40, 60, 80, ..., 360, 380. Dette er mulige koordinat-plasseringer for frukten. Når en tilfeldig plassering er funnet, må vi sjekke om denne plasseringen er ledig. Det vil si at den ikke kolliderer med kroppen eller hodet til slangen. Hvis den kolliderer, må vi få en ny tilfeldig plassering av frukten. Derfor er koden omsluttet av en `while`-løkke som kjøres helt til vi har et sett med koordinater som ikke kolliderer med hodet eller kroppen. Når vi har det, kan ny frukt opprettes og legges til i `frukt`-arrayen.

```
function leggTilFrukt() {
  var ikkePlassert = true;

  while (ikkePlassert) {
    var fruktX = Math.floor(Math.random() * 20);
    var fruktY = Math.floor(Math.random() * 20);

    fruktX = fruktX * size;
    fruktY = fruktY * size;

    // hvis frukten ligger oppå hodet eller frukten treffer
       slangekroppen
    if (fruktX == x && fruktY == y || slangeHit(fruktX, fruktY))
       {
      // løkken må kjøres igjen
    } else {
      // opprett en ny frukt
      var f = { xPos: fruktX, yPos: fruktY };
      frukt.push(f); // legg frukten til arrayen
      ikkePlassert = false;
    }
  }
}
```

Kode 23.11: Funksjonen `leggTilFrukt()`.

Når `leggTilFrukt()`-funksjonen er ferdig, må vi kalle den én gang per frukt vi ønsker på spillbrettet. Det gjøres i `startSpill()`-funksjonen før funksjonskallet på `tegnOpp()`. Erstatt kodelinjen:

```
// TO DO: legg til frukt
```

Med:

```
leggTilFrukt();
leggTilFrukt();
```

Prøver du å kjøre koden nå, skjer det ikke noe nytt, for vi mangler selve opptegningen av frukten. Vi må føye til en løkke som tegner opp all frukten som ligger i arrayen `frukt` i slutten av `tegnOpp()`-funksjonen. Se koden nedenfor.

```
function tegnOpp() {
  ctx.clearRect(0, 0, 400, 400); // blanker ut canvaset
  ctx.fillStyle = "red";
  ctx.fillRect(x, y, size, size);
  ctx.fillStyle = "yellow";

  // tegner alle slangeleddene
  for(var i = 0; i < slange.length; i++) {
    ctx.fillRect(slange[i].xPos, slange[i].yPos, size, size);
  }

  // tegner opp all frukten
  for(var i = 0; i < frukt.length; i++) {
```

```
    ctx.fillStyle = "green";
    ctx.fillRect(frukt[i].xPos,frukt[i].yPos, size, size);
  }
}
```

Kode 23.12: Den ferdige versjonen av funksjonen `tegnOpp()`.

Kjører vi koden nå, så ligger frukten på spillbrettet, men slangen kan ikke spise den, så nå trenger vi å sjekke om hodet kolliderer med frukten. Det gjør vi i slutten av `flytt()`- funksjonen etter at vi har sjekket om hodet kolliderer med slangekroppen (nedenfor vises hele `flytt()`-funksjonen, det er bare den nederste biten som er ny).

```
function flytt() {
  // oppretter et slangeledd der hodet var
  var slangeLedd = { xPos: x, yPos: y };
  // legg til slangeLedd først i arrayen
  slange.unshift(slangeLedd);

  // hvis slangen er lengre enn maks lengde
  if (slange.length > maksLengde) {
    // fjern det siste slangeleddet i arrayen
    slange.pop();
  }

  if (retning == 37) {  // retningen er venstre
    // hvis det er plass til å gå til venstre
    if (x - size >= 0) {
      x -= size; // reduser x-koordinaten med 20px
      tegnOpp();
    }
  } else if (retning == 39) { // retningen er høyre
    // hvis det er plass til å gå til høyre
    if (x + size < 400) {
      x += size; // øk x-koordinaten med 20px
      tegnOpp();
    }
  } else if (retning == 38) { // retningen er oppover
    // hvis det er plass til å gå oppover
    if (y - size >= 0) {
      y -= size; // reduser y-koordinaten med 20px
      tegnOpp();
    }
  } else if (retning == 40) { // retningen er nedover
    // hvis det er plass til å gå nedover
    if (y + size < 400) {
      y += size; // øk y-koordinaten med 20px
      tegnOpp();
    }
  }

  // sjekk om koordinatene til hodet treffer koordinatene til
     kroppen
  if (slangeHit(x, y)) {
```

```
    gameOver(); // hodet traff kroppen - avslutt spillet
  }

  // sjekk om hodet til slangen treffer frukten.
  // Hodet er alltid på punktet (x, y)
  for (var i = 0; i < frukt.length; i++) {
    if (x == frukt[i].xPos && y == frukt[i].yPos) {

      poeng++; // øk poengene

      // oppdater poengvisningen
      poengEl.innerHTML = "Poeng: " + poeng;

      // øk lengden til slangen
      maksLengde += vekst;

      frukt.splice(i, 1); // fjern frukten
      leggTilFrukt(); // legg ut en ny frukt
    }
  }
}
```

Kode 23.13: Den ferdige versjonen av funksjonen `flytt()`.

Test koden igjen, og vi ser at nå forsvinner frukten når slangen spiser den. Frukten blir deretter tegnet opp på nytt et annet sted. Poengsummen øker også når slangen spiser en frukt. Det siste vi skal gjøre nå, er å få slangen til å bevege seg framover av seg selv. I `startSpill()`-funksjonen må vi sette et intervall på `timer` slik at funksjonen `flytt()` kalles med jevne mellomrom.

```
timer = setInterval(flytt, hastighet);
```

Hvis du tester koden igjen nå, vil du merke at innimellom er det som om slangen hopper to plasser når du bytter retning. Dette skjer fordi `flytt()` alltid blir kalt etter et angitt tidsintervall, uavhengig av om brukeren bytter retning. For å unngå dette, må vi i funksjonen `byttRetning()` fjerne tidsintervallet og legge det til på nytt. Se koden nedenfor for den fullstendige koden til funksjonen `byttRetning()`.

```
function byttRetning(e) {
  if (e.keyCode > 36 && e.keyCode < 41) {
    clearInterval(timer);                  // fjerner timeren
    timer = setInterval(flytt, hastighet); // setter en ny timer
    retning = e.keyCode;
    flytt();
  }
}
```

Kode 23.14: Den ferdige versjonen av funksjonen `byttRetning()`.

Det siste vi mangler er å slette timeren når spillet er over, altså i `gameOver()`-funksjonen. Denne koden skal ligge øverst i funksjonen:

```
function gameOver() {
  // fjerner keydown-hendelse på body-elementet
  bodyEl.removeEventListener("keydown", byttRetning);
  clearInterval(timer); //Fjerner timeren

  console.log("Game over!");

  //TODO: Lag en gameover beskjed til brukeren
  //TODO: Restart spillet
}
```

Kode 23.15: Den ferdige versjonen av funksjonen `gameOver()`.

Nå er spillet ferdig, men det er muligheter for individuelle tilpasninger og utvidelser hvis du ønsker det. Se forslag til utvidelser nedenfor.

Forslag til utvidelser

Som ekstra utfordringer kan du prøve å forbedre snake-spillet med de 6 forslagene nedenfor.

Game over

Legg til en «game over»-beskjed til brukeren med en knapp som starter spillet på nytt (se figur 22.3).

Hastighet

Øk hastigheten på slangen etter hvert som man får flere poeng.

Grafikk

Legg inn bilder/pixelart for frukten.

Ulike frukter

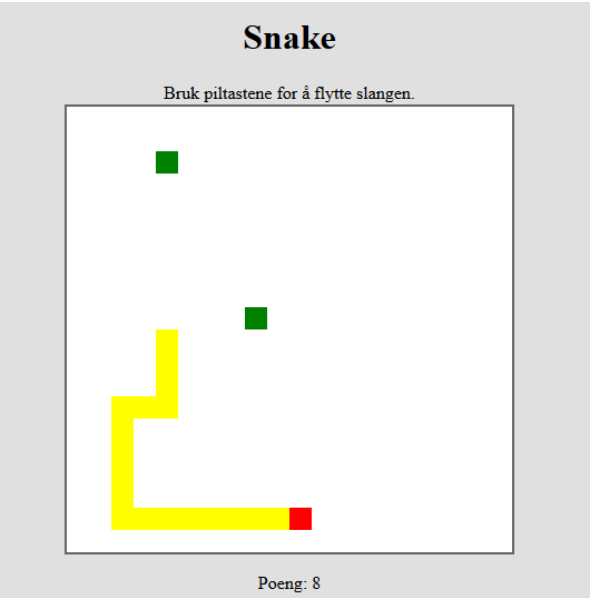
Lag ulike typer frukt som gir forskjellige poeng.

Krympemat

Lag «krympemat» slik at slangen kan bli kortere og la den for eksempel forsvinne hvis ikke slangen har spist den innen kort tid.

Fordeling av frukt

I denne versjonen kan flere frukt havne oppå hverandre, hvordan kan man unngå det?



Figur 23.2: Skjermbilde av det ferdige spillet.

Figur 23.3: Eksempel på «Game over»-beskjed med «Spill igjen»-knapp.