

▲ JS: Partikkel-fest

Skrevet av: Lars Klingenberg

Oversatt av: Stein Olav Romslo

Kurs: Web

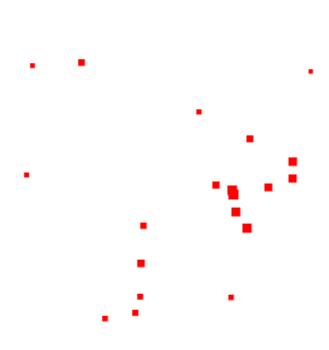
Tema: Tekstbasert, Nettside, Animasjon

Fag: Matematikk, Programmering, Kunst og håndverk Klassetrinn: 5.-7. klasse, 8.-10. klasse, Videregående skole

Introduksjon

Denne oppgåva byggjer på koden du skreiv i oppgåva Partikkel-animasjon (../partikkel_animasjon/partikkel_animasjon_nn.html). Så viss du ikkje har gjort den anbefalar me at du ser på den før du fortset med denne oppgåva.

Her skal me vidareutvikle partikkel -animasjonen vår slik at den ser slik ut:



Merk at i denne oppgåva kjem me berre til å gi hint for å beskrive kva du skal gjere. Du kjem ikkje til å få presentert den ferdige koden.

Steg 1: Kva må gjerast?

I denne oppgåva får du berre små døme på kode for å hjelpe deg til å kome fram til resultatet. Difor skal me gå jgennom tankemåten for å lage animasjonen over ved å presentere ei liste over ting som må gjerast:

La oss studere animasjonen og analysere kva den inneheldt:

| 0 | Ein partikkel på midten av skjermen som alltid er der. Kva kan vere grunnen til det? | |
|---|---|--|
| 0 | Partiklane som går ut frå midten og blir mindre og mindre di lengre ut dei går. | |
| 0 | Hastigheita til hvar partikkel varierer. | |
| 0 | Retninga varierer, men ein partikkel beveger seg alltid i ei rett linje. | |
| 0 | Det er mange partiklar som blir til kvart sekund. | |
| Me skal analysere punkta og sjå på kva me må programmere. Me startar på toppen. | | |
| 0 | Sidan partiklane går ut frå midten må alle starte der. Difor må me setje x - og y - posisjonen til å vere det same for kvar partikkel. | |
| 0 | Sidan partiklane blir mindre og mindre, men startar med same storleik, så må me endre på storleik -attributten til partikkelen på same måte som me gjer når me skal flytte på den. Tips: bruk ganging (*) for å få ein betre forminskingseffekt. | |
| 0 | Sidan hastigheita varierer kan me bruke Math.random til xSpeed og ySpeed. Her er eit døme på korleis det kan sjå ut: | |
| | | |

xSpeed: Math.floor(Math.random()*20 - Math.random()*20));

Dette gjer at du får eit positivt eller negativt tal med varierande hastigheit frå -20 til 20 i \times -retning. Gjer det same for y -retninga for å få partiklane til å bevege seg over alt på skjermen.

- For å få dei til å følgje ei rett linje brukar me berre endringar i x og y -retning frå førre oppgåve: particle.x = particle.x + particle.xSpeed;.
- Sidan det er mange partiklar som blir laga samstundes må me leggje kvar nye partikkel i ei liste for kvar gong draw() blir kalla, og bruke ei for -løkke til å endre kvar partikkel sine attributtar, og gjenta det for alle elementa i lista.

Prøv sjølv fyrst! Viss du ikkje får det til kan du nytte hinta under.

Hint

For-løkke

Ei for -løkke som skal gå gjennom ei liste ser slik ut:

Oppbygging av koden

For at du skal kunne byggje opp koden slik at partiklane oppfører seg som i animasjonen, så må me tenke over kor me set inn koden vår.

All endring på partikkel-objektet bør skje i for -løkka. På den måten skjer endringane gradvis, og animasjonen blir finare.

| 0 | Du bør eksperimentere med når elementa bør leggjast til i partikkel -lista. |
|---|---|
| 0 | Du bør òg eksperimentere med når du brukar clearRect(). Klarar du å sjå kva som er skilnaden på om den ligg i eller utanfor for -løkka? |

Lisens: CC BY-SA 4.0 (http://creativecommons.org/licenses/by-sa/4.0/deed)