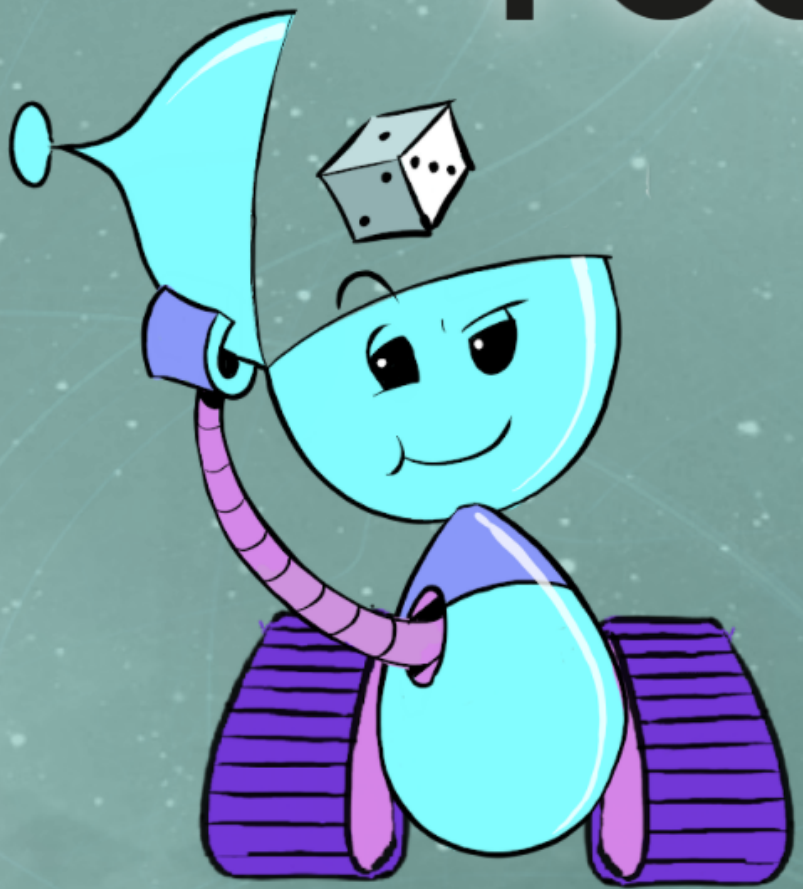
Pelinteko ja tuotekehitys

SUOMEN KOODIKOULU



Oppilaan kirja

Jussi Koivisto,
Jaakko Korpela

Pelinteko ja tuotekehitys

Oppilaan kirja - Versio 1.0

Kustantaja: Suomen Koodikoulu Oy
www.codeschool.fi

ISBN: 978-952-7403-26-6

SUOMEN KOODIKOULU

CODE SCHOOL FINLAND

Sisällys

Osa 1

Keskitason Scratch-ohjelmointia

Nykymaailmassa voit törmätä lähes missä tahansa elektronisiin laitteisiin, joita ohjaa **tietokone**. Tietokonetta puolestaan ohjaa **koodi**. Voidaksemme ymmärtää ja hallita ympäröivää maailmaamme, meidän täytyy ymmärtää, kuinka tietokoneet ja koodit toimivat.

Tämä kirja toimii oppaanasi tietokoneiden ohjelmoinnin maailmaan ja auttaa sinua kehittymään **ohjelmoijana**. Kun olet opiskellut sisällöt joita tämä kirja tarjoaa, voit jo melkein kutsua itseäsi huippuohjelmoijaksi!

Osassa 1 syvennät tietojasi Scratch-ohjelmointiympäristön käytöstä ja ymmärrystäsi ohjelmoinnillisesta ajattelusta. Eiköhän aloiteta!

Aloitetaan - Valmistautuminen osaan 1

1. Mitä erilaisia sovelluksia olet käyttänyt tänään? Mitä tulet vielä luultavasti käyttämään päivän aikana?

   _____

   _____

2. Ajattele jotain itsellesi mieluista ohjelmaa. Se voi olla sovellus, peli tai jotain ihan muuta. **Piirrä se**!

3. Ajattele suosikkiohjelmaasi, jonka juuri piirsit.

   a) Ymmärrätkö pääpiirteittäin, kuinka se on ohjelmoitu?

   _____

   b) Millaisia muutoksia haluaisit tehdä suosikkiohjelmaasi?

   _____

   _____

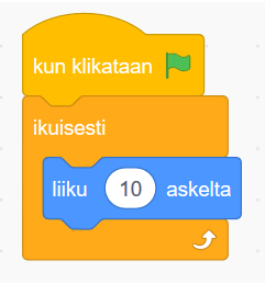
4. a) Lopuksi **– kuvittele unelmiesi ohjelma, jollaisen toivoisit olevan olemassa**

   b) **Keskustele** unelmiesi sovelluksesta **kavereiden kanssa**.

Luetaan – Scratchin kertaus

Scratch ohjelmointi on...

Visuaalista **lohko-ohjelmointia**. Scratchissa ohjelmoidaan rakentamalla **koodeja** liikuteltavien **lohkojen** avulla.



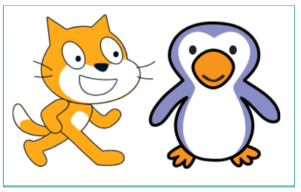
Scratch on **selainpohjainen visuaalinen ohjelmointiympäristö**, jolla voidaan ohjelmoida visuaalisia projekteja, kuten pelejä ja sovelluksia. Scratchissa koodit rakennetaan *liikuteltavien lohkojen* avulla. Lohkot löytyvät ohjelmointinäkymän vasemmasta reunasta, ja niitä käytetään **klikkaamalla ja raahaamalla**.

Ohjelmointi Scratchissa

Voit luoda **omia ohjelmia** kuten pelejä ja sovelluksia käyttämällä Scratchia! Tässä lista **tärkeistä sanoista** joihin sinun täytyy tutustua ennen kuin siirryt Scratchiin:

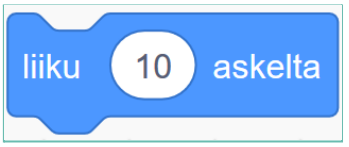
Hahmo on...

Digitaalinen kuva, joka **tottelee komentoja** sanasta sanaan. Hahmoja ohjelmoidaan erilaisiin toimintoihin Scratchissa.



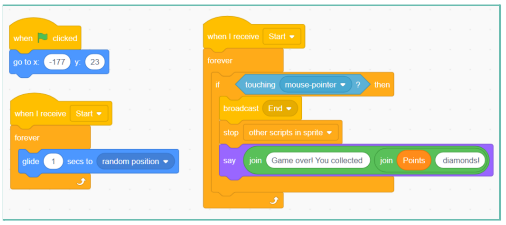
Komento on...

Yksittäinen ohje hahmolle. Esimerkiksi liiku (10) askelta. Scratchissa komentoja sanotaan **lohkoiksi**. **Useampaa lohkoa** kiinnitettynä toisiinsa kutsutaan **skriptiksi**.



Koodi on...

Usean komentoketjun yhdistelmä. Kun koodista syntyy jotain toimivaa, sitä kutsutaan **ohjelmaksi**. Pelit ja sovellukset ovat ohjelmia.



Scratchin yleiskatsaus

Scratch koostuu seitsemästä pääelementistä: ***koodeista***, ***asusteista***, ***äänistä,*** ***taustoista***, ***hahmoista***, ***ohjelmointialueesta*** and ***näyttämöstä***. Näiden avulla voit luoda omia ohjelmia, kuten pelejä, sovelluksia ja animaatioita.

Koodit koostuvat erivärisistä lohkoista:

Liike-lohkoja käytetään hahmojen **liikkeiden** ohjaamiseen.

Ulkonäkö-lohkojen avulla voidaan muokata hahmojen **ulkonäköä**.

Ääni-lohkoja käytetään projektien **äänien** ohjaamiseen.

Tapahtumat-lohkot aistivat tapahtumia ja **laukaisevat skriptit**.

Ohjaus-lohkoja käytetään **skriptien ohjaamiseen**.

Tuntoaisti-lohkoja käytetään tapahtumien **aistimiseen**.

Toiminnot-lohkoja käsitellään **numeroita** ja **merkkijonoja** (tekstiä)**.**

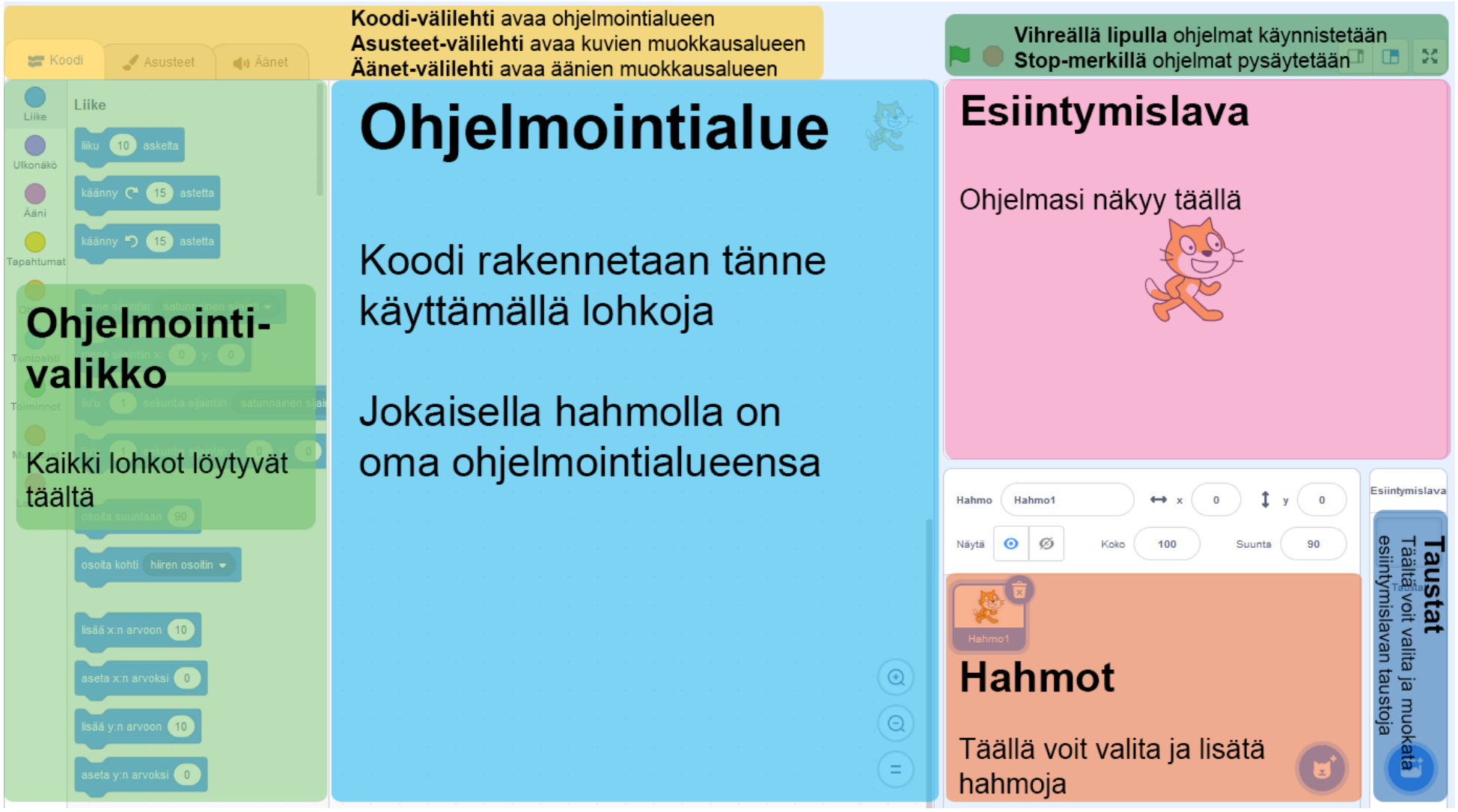
Muuttujat-lohkoja käytetään **tietojen tallettamiseen**.

Voit myös luoda **omia lohkoja** (***Lohkoni***) valikossa.

Scratchin päänäkymä esitellään seuraavalla sivulla →

Koodi-välilehti avaa ohjelmointialueen
Asusteet-välilehti avaa kuvien muokkausalueen
Äänet-välilehti avaa äänien muokkausalueen

Vihreällä lipulla ohjelmat käynnistetään
Stop-merkillä ohjelmat pysäytetään



Keskustele

↑ *Tämän **Keskustele**-kyltin ollessa tehtävien yläpuolella, keskustelkaa annetuista aiheista oman ryhmänne kanssa.*

Pohtikaa seuraavia kysymyksiä **ryhmissä**. **Kirjoittakaa** vastauksenne.

1. Mitä tiedät tällä hetkellä ohjelmoinnista (Scratchissa)?

   _____

   _____

   _____

2. Millaisia ohjelmointiprojekteja olet tähän mennessä tehnyt Scratchissa?

   _____

   _____

   _____

Ohjeet

↑ *Tämä **Ohjeet**-kyltti tarkoittaa, että kirja haluaa sinun seuraavan alla olevia ohjeita.*

1. **Avaa verkkoselain** tietokoneellasi.
2. **Mene** osoitteeseen **scratch.org** (tai scratch.mit.edu).
3. **Klikkaa Luo** sivun yläreunassa olevasta painikkeesta

4. **Selaile ohjelmointivalikkoa** sivun vasemmassa reunassa tutustuaksesi Scratch-ohjelmointiympäristöön. Käytä edellisellä sivulla olevaa kuvaa apunasi.

   ***Huomautus:** Jos sinulla on **Scratch-tunnus**, kirjaudu sisään. Jos opettaja haluaa että luotte tunnukset, odota ohjeita ennen kuin jatkat eteenpäin.*

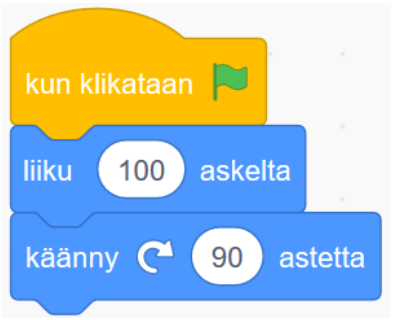
Ohjelmoidaan – Komennot, toistorakenteet ja ehtolauseet

Komennot ja tapahtumat

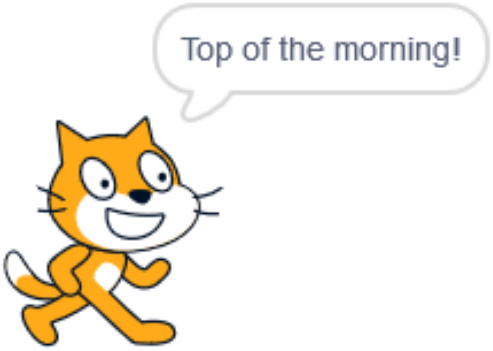
Scratchissa **hahmoille** annetaan **komentoja**. Komennot liitetään ***Tapahtumiin***. Tapahtuma, johon on kiinnitetty yksi tai useampi komento on nimeltään **skripti**. Luo seuraavaksi yksinkertainen skripti seuraamalla alla olevia ohjeita.

Jatka työskentelyä edellisellä sivulla aloitetun projektin parissa.

Ohjeet



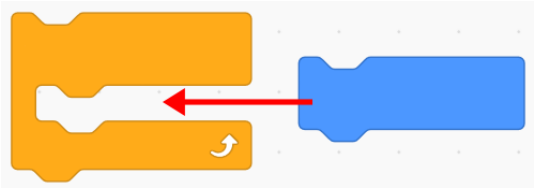
1. **Tuo** ohjelmointialueelle kun klikataan -lohko ja liitä siihen liiku ( ) askelta ja käänny ( ) astetta-lohkot
2. Aseta hahmo **liikkumaan 100 askelta** ja **kääntymään 90 astetta** ja **klikkaa sitten kuvaketta** käynnistääksesi ohjelman. Jatka klikkailua nähdäksesi kuinka hahmo liikkuu. Voit aktivoida skriptit myös klikkaamalla niitä.
3. **Muuntele lukuja** liiku ( ) askelta ja käänny ( ) astetta-lohkoissa nähdäksesi kuinka muutokset vaikuttavat koodin toimintaan.
4. Lisää odota (1) sekuntia-lohko Ohjaus-valikosta liiku ( ) askelta ja käänny ( ) astetta-lohkojen väliin nähdäksesi kuinka se vaikuttaa koodiin.



5. Laita hahmo **sanomaan jotain** lisäämällä sano (Hei!) (2) sekunnin ajan-lohko Ulkonäkö-valikosta.
6. **Laajenna skriptiä** lisäämällä siihen **vähintään 5 lohkoa** Liike ja Ulkonäkö-valikoista.

Toistorakenteet

Ohjaus-valikosta löytyvät lohkot mahdollistavat *skriptien ohjaamisen* koodeissa. Valikosta löytyy esimerkiksi **toistorakenteita** ja **ehtolauseita**. Näiden lohkojen avulla koodeista voidaan tehdä *tiiviimpiä*, *älykkäämpiä* ja *helpommin muokattavia*.



*Esimerkiksi toista ( ) kertaa ja jos ( ), niin-lohkoja kutsutaan myös **C-lohkoksi** niiden muodon vuoksi. Niitä käytetään lisäämällä muita lohkoja niiden **hakasten väliin**.*

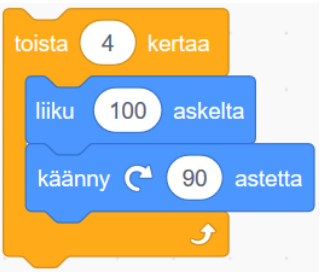
Toista-lohkoja käytetään ohjelmissa haluttujen komentojen **toistamiseen**. Ne tunnetaan yleisemmin nimellä **looppi** tai **silmukka**. Niiden avulla komentoja voidaan toistaa *ikuisesti* tai jonkin *määrityksen* tai *ehdon* mukaisesti.

Ohjeet

1. **Jatka** aiemmin aloitetun projektin parissa.

2. **Tuo** toista ( ) kertaa-lohko ohjelmointialueelle ja liitä liiku ( ) askelta ja käänny ( ) astetta-lohkot sen sisään.

3. **Näppäile** toista ( ) kertaa-lohkon tekstikenttään numero 4. **Käynnistä ohjelma** testataksesi kuinka toista ( ) kertaa-lohko vaikuttaa skriptin toimintaan.



4. **Korvaa** toista ( ) kertaa-lohko ikuisesti-lohkolla. Käynnistä ohjelma nähdäksesi kuinka se eroaa aiemmasta.

*Vasemmalla oleva **looppi** tai **silmukka** toistaa sen sisälle asetettua komentoa 10 kertaa. Hahmo liikkuu siis 10 askelta 10 kertaa, liikkuen yhteensä 100 askelta. Tämä eroaa kuitenkin komennosta "liiku 100 askelta". Osaatko selittää, miten?*

Testaa ja tutki!

↑ *Tämä **Testaa ja tutki!**-kyltti tarkoittaa, että saat rohkeasti tutkia, miten asiat toimivat.*

Työskentele parin kanssa ja **tee seuraavat tehtävät**:

1. a) **Avatkaa selaimessa tämä osoite:** codeschool.fi/pt1 ja
   b) Klikatkaa **Katso sisälle** -kuvaketta.

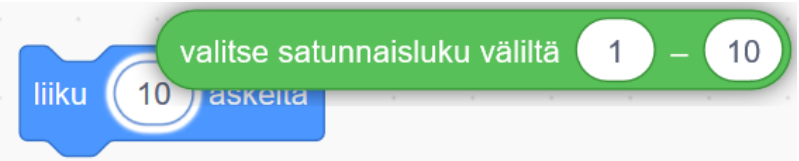
2. **Älkää vielä painako mitään näppäimistön näppäimiä! Katsokaa koodia** ja **keskustelkaa** parin kanssa, mitä luulette tapahtuvan jos painatte...
   a) **A-näppäintä**?
   b) **B-näppäintä**?
   c) **C-näppäintä**?

3. Kokeilkaa nyt **painaa A, B tai C yksi kerrallaan**. Ennen seuraavan näppäimien painamista palauttakaa ohjelma alkutilaansa painamalla **vihreää lippua**. 🚩

💡 *Täältä löytyy tärkeää tietoa **toiminnoista**, **ehdoista** ja **ehtolauseista.***

Toimintolohkot

Toimintolohkot voidaan jakaa **logiikka**- ja **operaattori**lohkoihin. Niitä ei käytetä toisten lohkojen tavoin lisäämällä niitä skriptien jatkoksi, vaan **asettamalla ne toisten lohkojen vastaavan muotoisiin aukkoihin** korvaamaan **teksti**- tai **numerokenttää**. Toimintolohkot ovat väriltään sinisiä ja vihreitä.



Jos viet toimintalohkon sopivan aukon päälle, aukon ympärille syttyy korostusvalo.

Logiikkalohkot ovat kuusikulmaisia ja ne voidaan asettaa samanmuotoiseen kuusikulmaiseen aukkoon toisessa lohkossa. Ne ilmoittavat *totuusarvoja*, kuten *tosi* tai *epätosi*, *1* tai *0*.

Esimerkiksi tämä koskettaako (reuna)?-lohko **tarkistaa ja ilmoittaa, koskettaako hahmo näyttämön reunaa**. Jos hahmo koskettaa reunaa, logiikkalohko ilmoittaa arvon **1**, joka tarkoittaa **tosi**. Jos hahmo puolestaan ei kosketa reunaa, ilmoittaa lohko arvon **0**, joka tarkoittaa **epätosi**. *Voit ajatella näiden logisten arvojen **1** tarkoittavan **kyllä** ja **0** tarkoittavan **ei**.*

Jos lisäät koskettaako (reuna)?-lohkon jos ( ), niin-lohkossa olevalle tekstialueelle, näistä muodostunut jos (koskettaako (reuna)?), niin-lohko suorittaa sen hakasten väliin rakennetun skriptin vain silloin, kun siihen asetettu ehto *koskettaako reunaa?* täyttyy.

Operaattorilohkot ovat pyöreäreunaisia ja ne voidaan asettaa samanmuotoiseen pyöräreunaiseen aukkoon toisessa lohkossa. Ne ilmoittavat *numeroita* tai *merkkijonoja (tekstiä).*

Ehtolauseet

Jos-lohkot tarkistavat niihin asetettua **loogista ehtoa**. Lohko tarkistaa onko sille asetettu ehto ***tosi*** vai ***epätosi***. Jos asetettu ehto *täyttyy* eli on *tosi*, jos-lohkon sisään rakennettu koodi suoritetaan. Jos ehto *ei täyty* eli on *epätosi*, ohjelma ohittaa lohkon ja liikkuu eteenpäin koodissa. Tämän vuoksi näitä lohkoja kutsutaan myös **ehtolauseksi**.

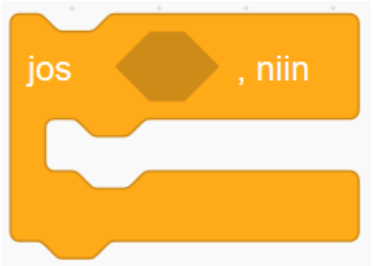
Loogiset arvot...

ilmaistaan luvuin ***1*** ja ***0***, jotka tarkoittavat ***tosi*** ja ***epätosi***. Niitä käytetään *ehtolauseissa* tarkistamaan *täyttyykö asetettu ehto (tosi)* vai *jääkö asetettu ehto täyttymättä (epätosi)*.

Tästä löydät lyhyet selitykset ja esimerkit erilaisista ehtolauseista:

Jos -ehto...

Tarkistaa onko ehto tosi ja suorittaa lohkon sisään rakennetun koodin **ainoastaan jos ehto täyttyy** eli on **tosi**.



Esimerkki:

Jos nykyinen **vuosi on 2020**, hahmo sanoo “Nyt on vuosi 2020!”.

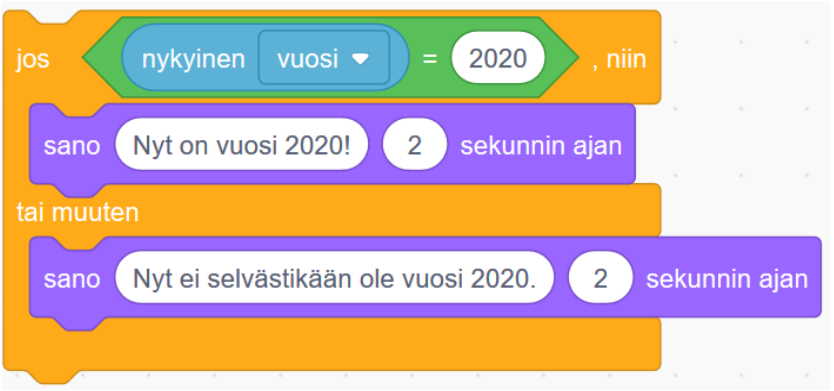
Jos **vuosi ei ole 2020**, ei tapahdu mitään.

Jos-tai muuten -ehto...

Tarkistaa onko ehto tosi.

Jos ehto **on tosi**, **ylempänä oleva komento suoritetaan**. Jos ehto on **epätosi**, **alempana oleva komento suoritetaan**.

Esimerkki:

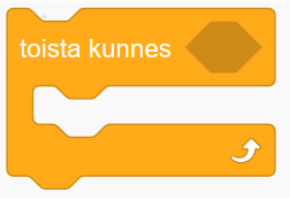


Jos nykyinen **vuosi on 2020**, hahmo sanoo “Nyt on vuosi 2020!”.

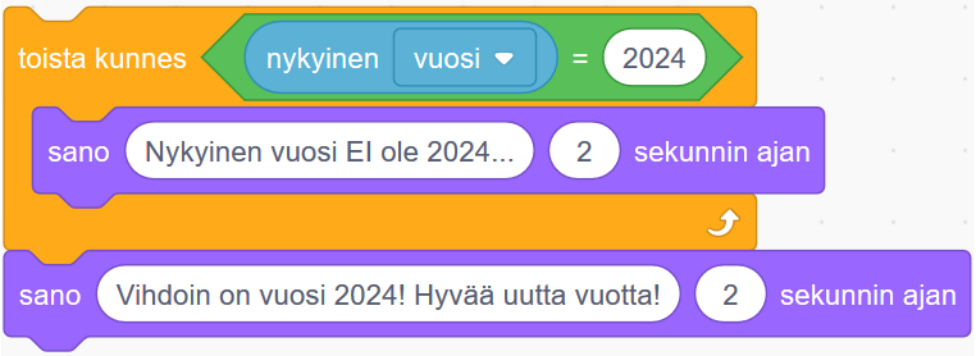
Jos **vuosi ei ole 2020**, hahmo sanoo “Nyt ei selvästikään ole vuosi 2020.”.

Toista kunnes -ehto...

Toistaa lohkon sisään asetettuja komentoja **kunnes ehto täyttyy.**



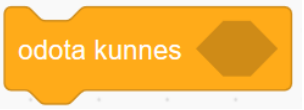
Esimerkki:



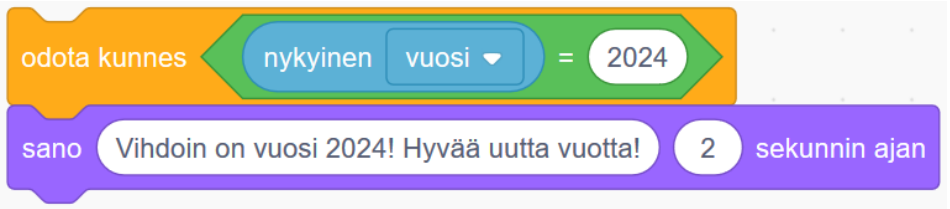
Hahmo sanoo **jatkuvasti** lausetta *“Nykyinen vuosi EI ole 2024...”*, siihen asti, että vuosi vaihtuu vuodeksi **2024** Kun vuosi on **2024**, asetettu ehto täyttyy jonka jälkeen **toisto loppuu** ja hahmo sanoo *“Vihdoinkin on vuosi 2024! Hyvää uutta vuotta!”*.

Odota kunnes -ehto...

Odottaa kunnes asetettu ehto täyttyy.



Esimerkki:



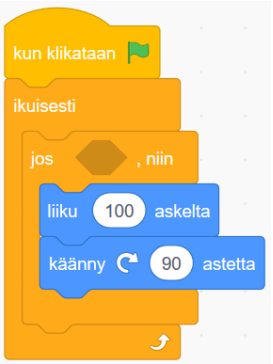
Lohko **odottaa** pysäyttäen ohjelman etenemisen, **kunnes vuosi on 2024**.

Sen jälkeen seuraava komento suoritetaan, ja hahmo sanoo *"Vihdoinkin on vuosi 2024! Hyvää uutta vuotta"*.

Ohjeet

1. **Jatka** aiemmin aloitetun projektin parissa.

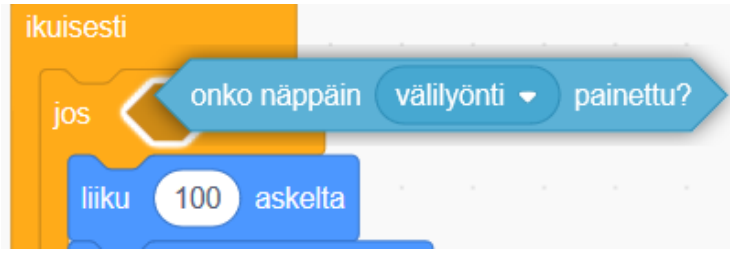
2. **Tuo** jos ( ), niin-lohko ohjelmointialueelle ja aseta se ikuisesti-lohkon sisään. Tämä kutsutaan *sisäkkäin asetteluksi*. Siirrä liiku ( ) askelta ja käänny ( ) astetta-lohkot sen sisään.



   ***Huomautus**: Sisäkkäisyyteen perehdytään kirjan **osassa 2**.*

3. Kun nyt **käynnistät ohjelman**, skripti tarkistaa *ikuisesti* täyttyykö *asetettu ehto*. Ohjelma ei pääty ennen kuin klikkaat **punaista stop-merkkiä**. Mitään ei kuitenkaan vielä tapahdu, sillä emme ole asettaneet vielä mitään *ehtoa*.

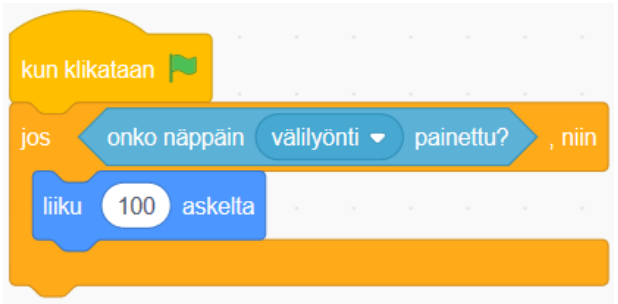
4. Mene Tuntoaisti-valikkoon ja **tuo** onko näppäin ( ) painettu?-lohko ohjelmointialueelle. Aseta se jos ( ), niin-lohkoon.



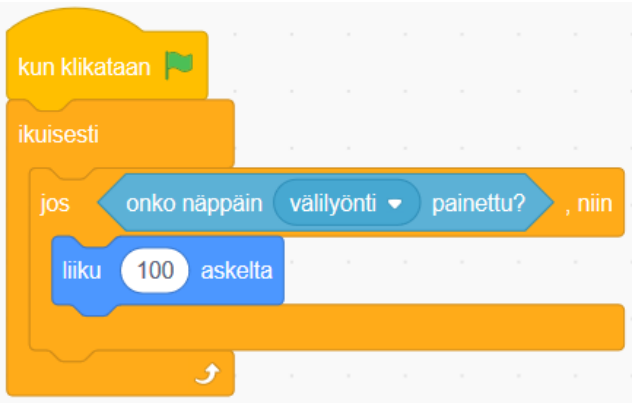
Onko näppäin ( ) painettu? on ***toimintalohko*** *ja siinä on* ***terävät reunat****. Se sopii tyhjään aukkoon jos ( ) , niin-lohkossa.*

5. Nyt voit **asettaa ehdon** skriptille. **Valitse jokin näppäin** ja käynnistä ohjelma. **Paina valitsemaasi näppäintä** nähdäksesi kuinka ohjelma toimii.

Huomautus: *Vasemmalla olevassa skriptissä* ***ei ole käytetty ikuisesti-lohkoa****. Se tarkistaa, onko välilyöntinäppäin painettuna* ***yhden kerran sen jälkeen kun ohjelma käynnistetään****, jonka jälkeen skripti päättyy. Käyttämällä ikuisesti-lohkoa* ***ehtolauseen ympärillä*** *saat skriptin pysymään päällä* ***ikuisesti****.*



Ei ikuisesti-lohkoa



Ehtolause asetettuna ikuisesti-lohkon sisälle

Testaa ja tutki!

1. a) **Mene selaimella tähän osoitteeseen:** codeschool.fi/pt2 ja
   b) Klikkaa **Katso sisälle** -nappia.

2. **Kokeile klikata näppäimiä A, B, C, D and E, yksi kerrallaan.**
   Ennen kuin kokeilet uutta näppäintä, resetoi ohjelma klikkaamalla 🚩 kuvaketta.

3. **Keskustele** parisi kanssa. Selittäkää eroavaisuudet...
   a) A ja B -näppäinten skriptien välillä
   b) C ja D -näppäinten skriptien välillä
   c) A ja E -näppäinten skriptien välillä

4. **Kuvaile**, mitä hahmo tekee kun **painetaan E -näppäintä**.

   _____

   _____

   _____

   _____

Haaste!

↑ *Tämä **Haaste**-kyltti tarkoittaa haasteita, jotka sinun tulee yrittää ratkaista itse.*

Tältä sivulta löydät **Scratchin kertaus -haasteita**. Suorittamalla seuraavat haasteet voit todistaa opettajallesi, että hallitset Scratch-ohjelmoinnin perusteet ja olet valmis siirtymään haastavampien tehtävien pariin.

Aloita luomalla uusi projekti. **Jaa projektisi** kun se on valmis.

Haaste 1: Tee ohjelma, jossa hahmo liikkuu, sanoo jotain ja vaihtaa asustetta. Koodissa pitää olla myös toista-lohko ja jos ( ), niin-lohko, johon on liitetty jokin Tuntoaisti-valikosta.

Haaste 2: Tee **piirustus** ohjelmoimalla. Käytä liiku ( ) askelta-lohkoja liikuttaaksesi hahmoa ympäri näyttämöä ja piirtääksesi. Käytä kynä-lohkoja piirtämiseen.

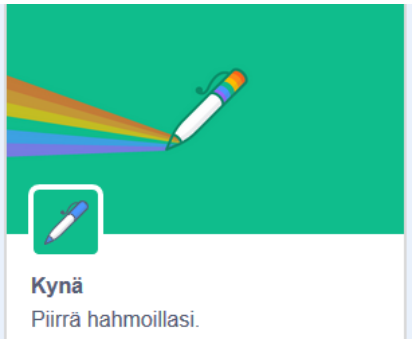
Vinkkejä haasteisiin

- Haasteessa 2 tarvitset **kynä-lohkoja**.

  Saadaksesi kynälohkot käyttöösi, sinun täytyy **lisätä laajennus**.

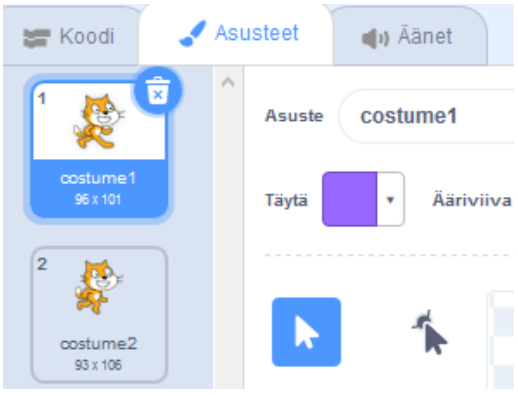
  Löydät valikon alareunasta **Lisää laajennus** kuvakkeen. Klikkaa sitä ja valitse **Kynä**.

  Nyt **kynä-lohkot** löytyvät valikosta.



- Voit hallita hahmojen **ulkonäköä ja animaatioita** **Asusteet**-välilehdellä. Tällä välilehdellä voit **muokata** ja **luoda uusia asusteita**.

  Luomalla **erilaisia asusteita** samalle hahmolle, voit **animoida** hahmoja. Lisäämällä seuraava asuste- tai vaihda asusteeksi ( )-lohkoja koodiin, voit laittaa hahmon vaihtamaan asusteesta toiseen ja saat hahmon näyttämään siltä, kuin se liikkuisi.



Testaa tietosi

Kun olet tehnyt ohjelmointitehtävät, vastaa seuraaviin kysymyksiin:

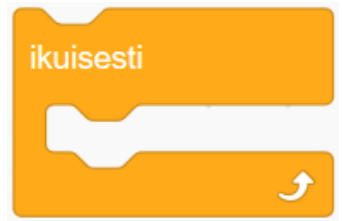
1. Selitä omin sanoin, miten **jos**-lohkot toimivat:

_____

_____

_____

2. Mitä **eroa** näillä kahdella eri **toistorakenteella** on?



_____

_____

_____

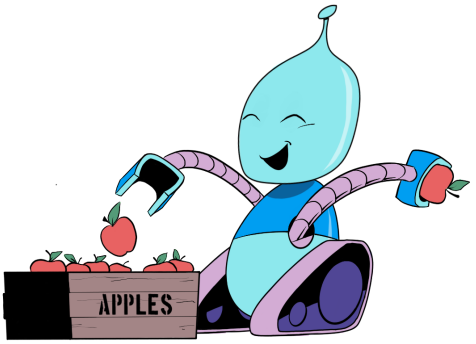
_____

_____

3. **Piirrä itsesi** ohjelmoijana!

Luetaan – Muuttujat

Ajattele **pahvilaatikoita**. Niitä käytetään **tavaroiden varastointiin**. Jos täyttäisit pahvilaatikon **omenoilla**, voisit kirjoittaa pahvilaatikon kylkeen tekstin **“Omenoita”**, antaen sille **luokituksen**. Kun lisäät pahvilaatikkoon omenoita, omenien lukumäärä, eli ohjelmointikielessä **arvo** kasvaa. Kun otat omenoita pois laatikosta, niiden arvo pienenee.



Muuttujat toimivat samalla tavalla. Ne ovat lohkoja, joilla on **luokitus** ja **arvo. Luokituksella** tarkoitetaan **muuttujan nimeä**, ja **arvolla** tarkoitetaan **muuttujaan tallennettua informaatiota**, kuten **numeroita** tai **sanoja.** Muuttujaan tallennettuja tietoja voidaan hakea ja käyttää myöhemmin niitä tarvittaessa.

Muuttujalla voi olla vain **yksi arvo kerrallaan**. Esimerkiksi, jos laitat laatikkoon **viisi omenaa**, laatikon *(eli muuttujan)* **arvoksi** muodostuu **viisi omenaa**. Jos **lisäät** laatikkoon **viisi omenaa**, laatikon arvo **kasvaa kymmeneen omenaan**. Jos sitten otat laatikosta **kaksi omenaa pois**, laatikon arvo **laskee kahdeksaan omenaan**.

Ohjelmointikielessä tämä ilmoitettaisiin “*Muuttujan ‘Omenoita’ arvo on 8”.*

Keskustele

1. **Mene selaimella tähän osoitteeseen: codeschool.fi/pt3**

2. a) **Tiputa omenoita** laatikon sisään klikkaamalla ympäriinsä ja **ota omenoita pois** laatikosta painamalla välilyöntiä.
   b) **Keskustele** parisi kanssa. Ymmärrättekö miten muuttujat toimivat?
   c) **Kuvaile omin sanoin kuinka muuttujat toimivat**.

   _____

   _____

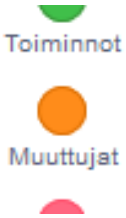
   _____

Ohjelmoidaan – Muuttujien määrittäminen ja muuttaminen

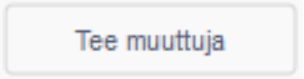
Tässä harjoituksessa **luomme muuttujan** joka muistaa **kuinka monta kertaa hahmoa klikataan**.

Ohjeet

1. Luo **uusi Scratch projekti**. Anna sille nimi *"Muuttujaharjoitus"*.

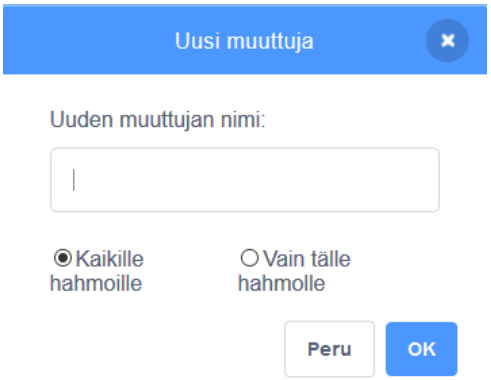


2. Mene Muuttujat-valikkoon ja tee uusi muuttuja klikkaamalla **Tee muuttuja** nappia.



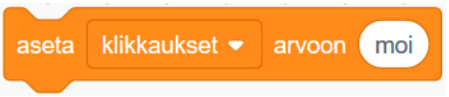
3. Anna muuttujalle nimi "klikkaukset".

Vinkki: *Kun nimeät muuttujia,* ***anna niille nimi joka kuvaa niiden käyttötarkoitusta****. Tämä auttaa sinua muistamaan, mihin käyttöön muuttuja on tarkoitettu. Jos tekisit esimerkiksi pistelaskuri muuttujan, voit nimetä sen nimellä "pisteet".* ***Tässä harjoituksessa muuttuja on nimeltään "klikkaukset", koska sitä käytetään klikkausten laskemiseen.***



4. Tuo aseta [klikkaukset] arvoon ( )-lohko ohjelmointialueelle.

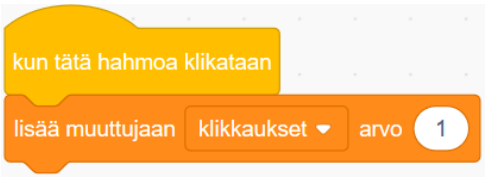
5. **Kirjoita jokin sana tekstikenttään**, ja **klikkaa** aseta [klikkaukset] arvoon ( )-lohkoa aktivoidaksesi sen.



6. Muuttujien arvot näkyvät näyttämön vasemmassa yläkulmassa. Huomaatko, että muuttujan arvoksi on nyt asetettu se sana, jonka kirjoitit tekstikenttään?

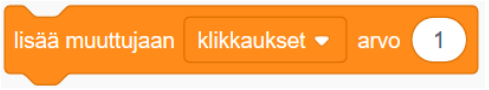
7. Kirjoita jokin **numero** sanan tilalle tekstikenttään ja klikkaa aseta [klikkaukset] arvoon ( )-lohkoa uudestaan. Muuttujan arvo on nyt muuttunut eriksi.

7. Tuo Tapahtumat-valikosta kun tätä hahmoa klikataan-lohko ja lisää se hahmosi ohjelmaan.



8. Liitä lisää muuttujaan [klikkaukset] arvo ( )-lohko siihen.

9. Joka kerta kun klikkaat hahmoa, muuttujan klikkaukset arvo kasvaa yhdellä. Voit **muuttaa klikkausten vaikutusta muuttujan arvoon muuttamalla lukua** lisää muuttujaan [klikkaukset] arvo ( )-lohkon tekstikentässä.



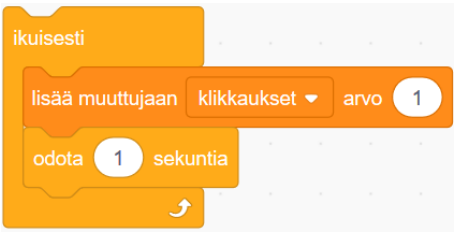
Testaa ja tutki!

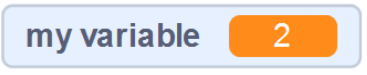
On aika testailla! **Jatka edellisellä sivulla aloittamasi projektin parissa**. Tässä osiossa **tutkimme muuttujien toimintaa**. Tästä on hyötyä myöhempää *Haaste!* -osiota varten.

1. **Luo erilaisia skriptejä**, jotka **määrittävät** tai **muuttavat** jonkin muuttujan arvoa. Keksitkö jotain jännittäviä tapoja käyttää muuttujia? Keskustele parin kanssa.

Tässä muutama **esimerkkiskripti**, joista voit ammentaa inspiraatiota:

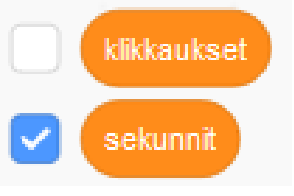
Muuttuja **muuttaa arvoaan yhdellä** aina sekunnin välein, toimien kuten ajastin!





Muuttujan **arvo määrittyy** joksikin luvuksi lukujen 1 ja 10 välillä joka kerta, kun skripti aktivoituu.

Vihje: *Voit **näyttää** tai **piilottaa** muuttujien arvot jotka näkyvät esiintymislavalla **poistamalla valinnan** muuttujien viereisestä valintaruudusta Muuttujat-valikossa.*



Muuttujien vertaaminen

Joskus koodatessa täytyy määrittää, onko muuttujan arvo **yhtä suuri**, **suurempi kuin** tai **pienempi kuin** jokin toinen arvo. Voit **verrata** muuttujan arvoa käyttämällä **( ) > ( ), ( ) < ( ) ja ( ) = ( ) toimintolohkoja**.

Jatka aiemmin aloitettua *muuttuja -projektia* seuraavassa tehtävässä.

Testaa ja tutki!

2. Luo **muuttuja** ja **skripti joka muuttaa muuttujan arvoa**.

3. Luo skripti, joka **tarkistaa** onko **muuttujan arvo yhtä suuri kuin jokin toinen arvo.**

4. **Ohjelmoi erilaisia asioita tapahtumaan** kun muuttujan saavuttaa jonkin tietyn arvon. Laita hahmo esimerkiksi **liikkumaan**, **sanomaan jotain** tai **vaihtamaan asustetta**.

Käytä jos ( ), niin-lohkoja käynnistämään eri tapahtumia.

Vinkki: *Laita jos ( ), niin-lohko ikuisesti-lohkon sisään että ohjelma tarkistaa muuttujan arvoa jatkuvasti!*

Voit käyttää myös odota kunnes < >-lohkoa jos ( ), niin-lohkon sijaan. Se toimii hieman eri tavalla. Osaatko selittää eron?

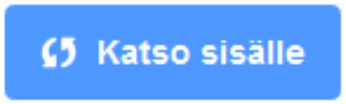
Pidä hauskaa muuttujien kanssa! Kun olet valmis, **vertaa luomuksiasi toisten oppilaiden kanssa** ja **jaa projektisi**.

Testaa ja tutki!

Työskentele parin kanssa ja **tee seuraavat tehtävät**:

1. a) **Mene selaimella tähän osoitteeseen: codeschool.fi/pt4**

   b) Klikatkaa **Katso sisälle** -kuvaketta.



2. **Katso Hahmo 1:n koodia**. **Keskustele** parisi kanssa ja kirjaa ylös mitä tapahtuu kun...
   a) **Klikkaat** 🚩 kuvaketta?
   b) **Klikkaat** hahmoa?
   c) Painat **välilyöntiä**?
   d) Mitä tapahtuu kun **muuttuja saavuttaa arvon 10**?

   _____

   _____

   _____

   e) Mitä tapahtuu kun **muuttujan arvo *ei ole* 10**?

   _____

   _____

   _____

3. Seuraavaksi **katso Hahmo 2:n koodia** ja **keskustele** parisi kanssa:
   a) Miten hahmojen **asusteenvaihtoskriptit eroavat toisistaan**?
   b) Miksi **asuste ei vaihdu takaisin** asuste1 **Hahmo 2:n** koodissa kun muuttujan arvo ei ole enää 10?

_____

_____

_____

4. **Muuta ohjelman koodia**. Ohjelmoi jotain tapahtumaan kun **muuttujan arvo on 15, 20 ja 25**. Käytä jos < >, niin; tai muuten-lohkoja **Hahmo 1:n** koodissa ja odota kunnes < >-lohkoja for **Hahmo 2:n** koodissa. Testaa, kuinka nämä kaksi erilaista skriptiä eroavat toisistaan!

💡 Vinkkejä useamman jos < >, niin; tai muuten-lohkon käyttöön

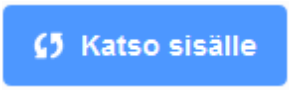
Kun lisäät koodin rakenteeseen useampia ehtolauseita eli jos < >, niin; tai muuten-lohkoja, ne täytyy asettaa **sisäkkäin**. Aseta uudet jos < >, niin; tai muuten-lohkot tai muuten osioon edellisen koodin jatkoksi. Muuten koodi ei toimi halutulla tavalla.

Sisäkkäin asettelusta löydät lisää kirjan **osasta 2**.

Klikkauspeli eli “klikkeri”

Remiksataan klikkauspeli! Seuraavassa tehtävässä pääset muokkaamaan valmiin klikkerin koodia ja ohjelmoimaan sen toimintaan hauskoja muutoksia.

Ohjeet

1. **Mene selaimella tähän osoitteeseen: codeschool.fi/pt5**
2. Klikkaa **Katso sisälle** -nappia. 

3. **Tarkastele koodia**. Tämä on klikkeri, jossa **muuttujia** käytetään **klikkausten laskemiseen**, **lisätehojen ostamiseen** ja **klikkausten tehostamiseen**.

Haaste!

Haasteissa 1-3, tehtäväsi on **remiksata** klikkeriä.

Haaste 1: Muokkaa pelin koodia siten, että klikkaukset-muuttujan arvo kasvaa **helpommin**.

Haaste 2: Luo **klikkauksen tehostaja** joka antaa **1000 klikkausta** yhdellä klikkauksella klikkausteho-muuttujan kautta.

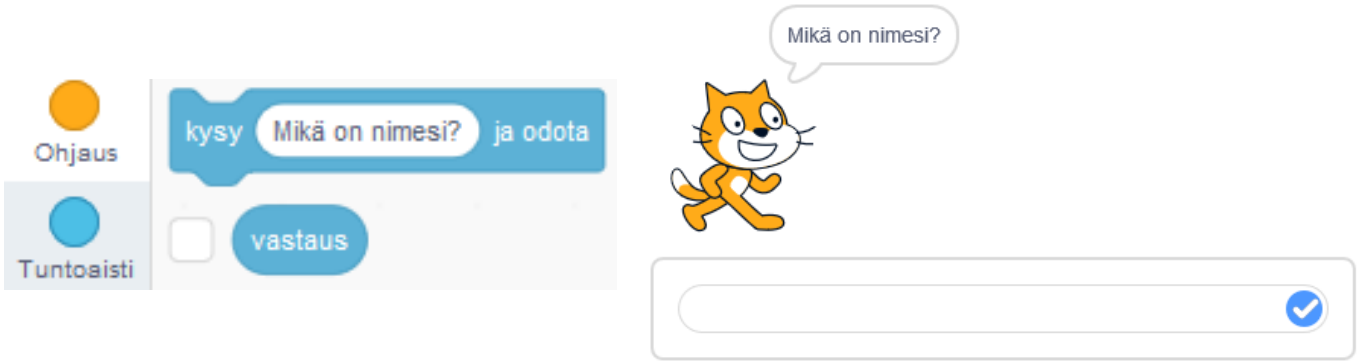
Haaste 3: Ohjelmoi **jotain tapahtumaan** kun klikkaukset-muuttuja saavuttaa jonkin haluamasi arvon.

💡 Vinkkejä haasteisiin

- **Haasteessa 2** sinun tulee luoda **uusi hahmo** tehdäksesi uuden **klikkauksen tehostaja** -napin.
- **Haasteessa 3** käytä odota kunnes < >-lohkoa saadaksesi ohjelman odottamaan kunnes muuttuja saavuttaa halutun arvon. Voit käyttää esimerkiksi Ulkonäkö-valikon lohkoja saadaksesi aikaan jotain jännittävää!

Kysyminen ja vastaaminen

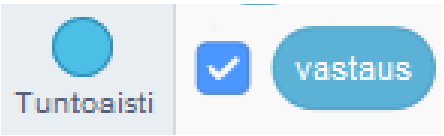
Scratchista löytyy valmiiksi tehty muuttuja nimeltään vastaus, jonka arvo määritetään **kysymällä**. Scratchissa peleistä on mahdollista tehdä **vuorovaikutteisia** lisäämällä niihin kysymyksiä ja joihin käyttäjä voi vastata.



Kun ohjelma **kysyy kysymyksen**, se **avaa tekstiruudun johon käyttäjä voi kirjoittaa**. Käyttäjän vastaus tallentuu vastaus-muuttujaan.

Ohjeet

1. Luo **uusi Scratch projekti**.
2. Tuo Tuntoaisti-valikosta kysy ( ) ja odota-lohko ohjelmointialueelle.
3. Klikkaa vastaus-muuttujan **vieressä olevaa valintaruutua** asettaaksesi muuttujan näkyviin näyttämöllä.

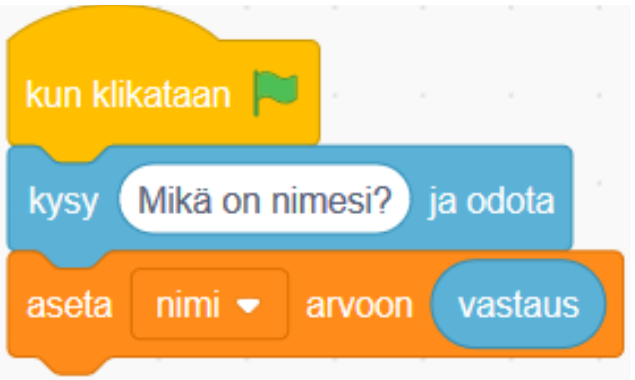


4. Klikkaa ohjelmointialueelle tuomaasi kysy ( ) ja odota-lohkoa. Hahmo kysyy sinulta kysymyksen, ja näyttämön alareunaan aukeaa vastauskenttä.
5. **Kirjoita jotain vastauskenttään ja klikkaa**  **kuvaketta**. Vastauksesi on nyt tallennettuna vastaus-muuttujaan.

Muistathan, että **muuttujalla voi olla kerrallaan vain yksi arvo**. Tämän vuoksi aina kun kysyt kysymyksen, **uusi vastaus korvaa edellisen vastauksen** vastaus-muuttujasta. Jos haluat kysyä useampia kysymyksiä, **sinun täytyy asettaa vastaus-muuttujaan tallennettu arvo johonkin toiseen-muuttujaan.**

Seuraavaksi harjoitellaan vastausten tallentamista toiseen muuttujaan!

Ohjeet



1. **Jatka** edellisellä sivulla aloittamasi projektin parissa.
2. **Ohjelmoi skripti**, joka asettaa nimi-muuttujan vastaamaan vastaus-muuttujan arvoa.

   Viereisessä esimerkissä (katso kuvaa) kysy ( ) ja odota-lohko ***kysyy*** *käyttäjältä kysymyksen, ja* ***asettaa annetun vastauksen vastaus-muuttujaan****.*

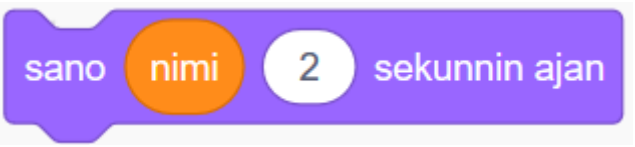
   Sen jälkeen ***aseta [nimi] arvoon (vastaus)-lohko asettaa nimi-muuttujan vastaamaan vastaus-muuttujan arvoa****.*

   Käyttäjän vastaus kysymykseen ”Mikä on nimesi?” on nyt tallennettuna nimi-muuttujaan, ja ohjelmassa voidaan nyt kysyä uusi kysymys ilman tämän vastauksen menettämistä.

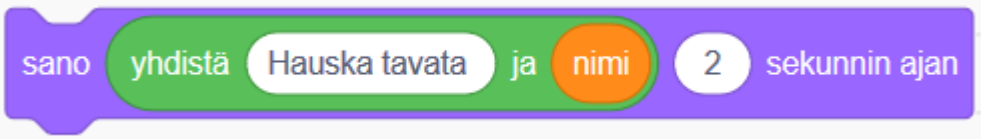
3. **Lisää seuraavaksi uusi kysymys** skriptiin ja **tallenna se johonkin toiseen muuttujaan**.

   Jos haluaisit ohjelmasi kysyvän käyttäjältä esimerkiksi tämän ***nimen*** ja ***iän***, koodisi voisi näyttää tältä. →

4. Laita hahmo **sanomaan muuttujalle tallennetun tieto** liittämällä nimi-muuttujalohko sano-lohkon tekstikenttään kuvan osoittamalla tavalla.



5. **Laita hahmosi sanomaan kokonaisia virkkeitä** joissa käytetään muuttujaa, kuten *”Hauska tavata nimi”*, käyttämällä yhdistä-toimintolohkoa. Sen avulla voit **liittää yhteen kaksi erillistä tekstikenttää**, joista toinen sisältää haluamasi tekstin ja toinen muuttuja-lohkon.



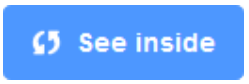
Vinkki: *Muista jättää yksi tyhjä välilyönti ensimmäisen lauseen jälkeen!*

6. **Tallenna** ja **jaa** projektisi.

Vinkki: *Voit yhdistellä* ***useita eri yhdistä-toimintolohkoja*** *asettelemalla ne* ***sisäkkäin****.*

Testaa ja tutki!

1. a) **Mene selaimella tähän osoitteeseen: codeschool.fi/pt6**

   b) Klikkaa **Katso sisälle** -nappia. 

2. **Katso koodia** ja **keskustele** parisi kanssa:
   a) Miten ohjelma tallentaa vastauksen toiseen muuttujaan?
   b) Miten ohjelmaa käyttää muuttujaan tallennettua vastausta?

3. Seuraavaksi **lisää kaksi uutta kysymystä ohjelmaan** ja **laita hahmo sanomaan uusiin kysymyksiin annetut vastaukset ääneen**. Sinun täytyy tehdä **kaksi uutta muuttujaa** uusia vastauksia varten.

Joissakin ohjelmissa, haluat ohjelmasi tunnistavat, onko käyttäjän antama vastaus **oikein**, eli **jotain mitä sinä olet määrittänyt**.

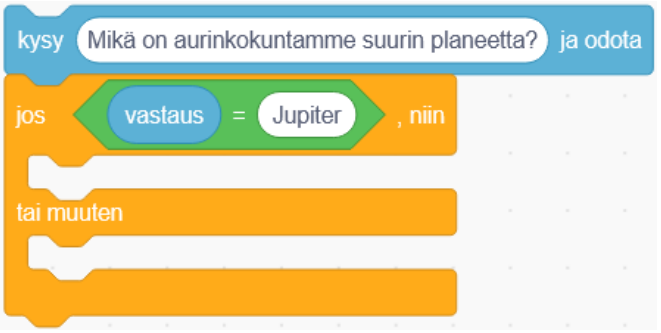
Voit ohjelmoida vastauksen tarkistuksen käyttämällä **( ) = ( ) logiikkatoimintoa**.

Ohjeet

Luo uusi projekti.

1. Käytä jos < >, niin; tai muuten-lohkoa **rakentaaksesi skriptin**, joka kysyy kysymyksen, jonka jälkeen se **tarkistaa onko vastaus jotain mitä sinä haluat sen olevan**.
2. **Jos vastaus on oikein**, laita hahmo kertomaan siitä pelaajalle ja anna hänelle **pisteitä**.
3. **Jos vastaus on väärin**, laita hahmo sanomaan jotain muuta, ja älä anna pelaajalle pisteitä.

   ***Huomautus**: Näitä vastauksia ei tarvitse tallentaa erilliselle muuttujalle, koska ohjelman ei tarvitse muistaa annettuja vastauksia enää myöhemmin.*
4. **Keksi muutama lisäkysymys ja tee projektista pieni tietovisa!**

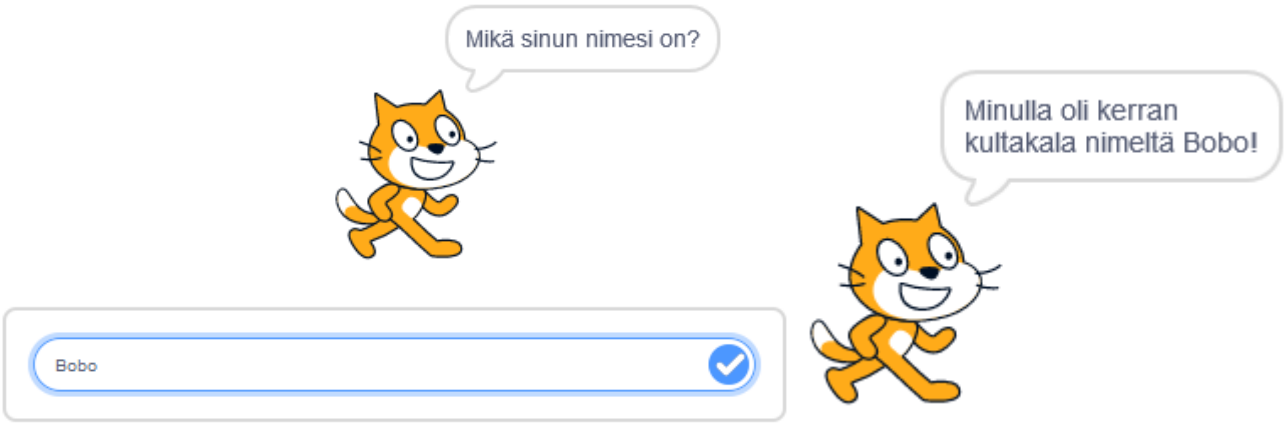


Haaste!

Haasteissa 1-3 sinun tulee käyttää taitojasi oppimiasi taitoja ja luoda erilaisia ohjelmia.

Haaste 1: Ohjelmoi **vuorovaikutteinen keskustelu**!

Tee **hassu keskustelu** jossa hahmo **kysyy kysymyksiä**, tallentaa vastaukset muuttujaan ja käyttää niitä myöhemmin jollain huvittavalla tavalla!



Haaste 2: Tee **tietovisa**!

Vaatimukset tietovisalle:

- Ohjelma **kysyy** ja **muistaa** pelaajan nimen.
- **Vähintään kolme kysymystä** joihin on määritetty **oikea vastaus**.
- Jokaisesta **oikeasta vastauksesta** pelaajalle **annetaan pisteitä**.
- *Katso vinkkejä ohjelmointiin sivuilta 26-29.*

Haaste 3: Tee **matikkapeli**!

Kuten Haastessa 2, sinun tulee tehdä tietovisa, mutta tällä kertaa **käyttämällä numeroita**. Ohjelmoi siis siis matikka-tietovisa! *Voit katsoa vinkkejä tähän haasteeseen seuraavalta sivulta.*

Vinkkejä haasteisiin

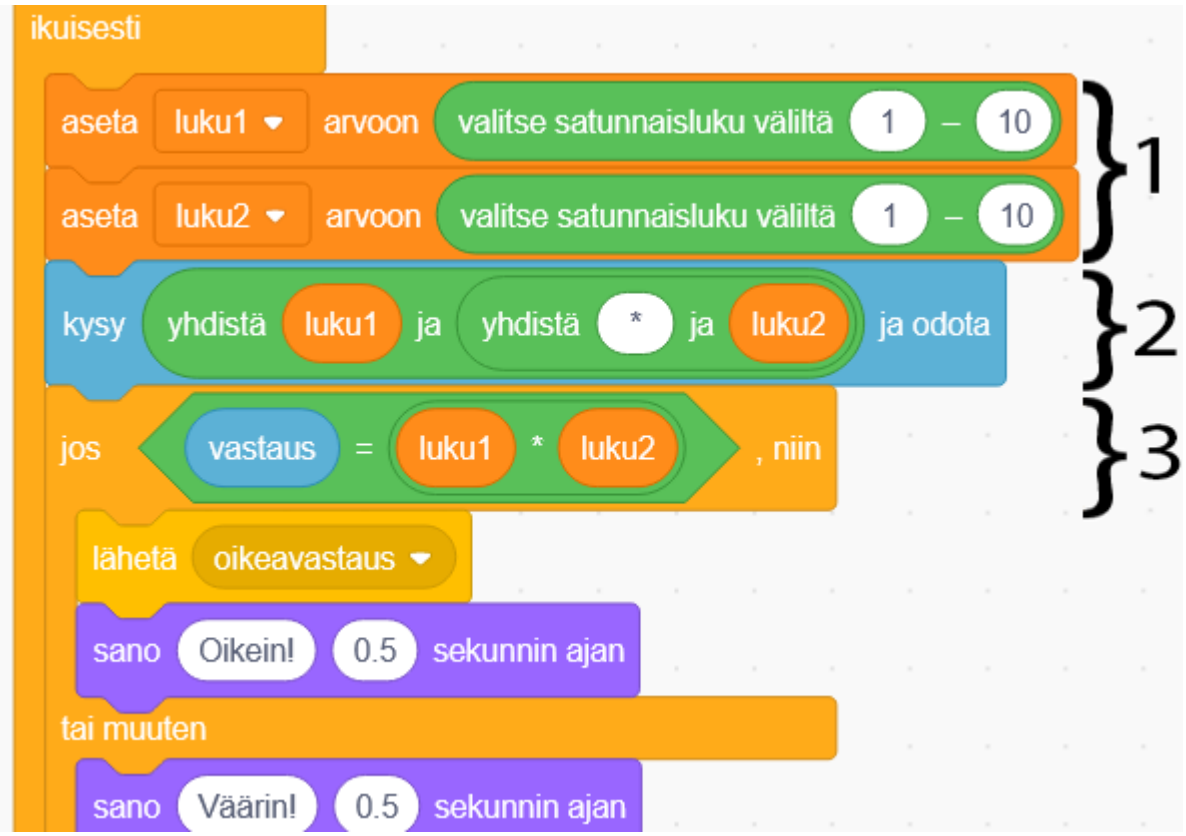
- Haasteessa 2 sinun tulee käyttää **muuttujia**, että **ohjelma muistaa pelaajan nimen**.
- Haasteessa 3 voit tehdä **matikkapelistäsi älykkään**! ***(vaativa)***

Alla oleva esimerkkikoodi toimii seuraavalla tavalla:

1. **Skripti valitsee satunnaisen numeron** valitse satunnaisluku väliltä (1) - (10)-toimintolohkon avulla **muuttujille** luku1 ja luku2.

2. Skripti **kysyy kysymyksen** "*luku1* * *luku2*". Kummassakin muuttujassa on nyt asetettuna satunnainen numero väliltä 1-10, jonka skripti on arponut.

3. Skripti tarkistaa onko vastaus yhtä suuri kuin muuttujien luku1 ja luku2 välinen **tulo** käyttämällä ( ) = ( ) logiikkatoimintoa. Muuttujien välinen **kertolasku** tehdään käyttämällä ( ) * ( ) logiikkatoimintoa.

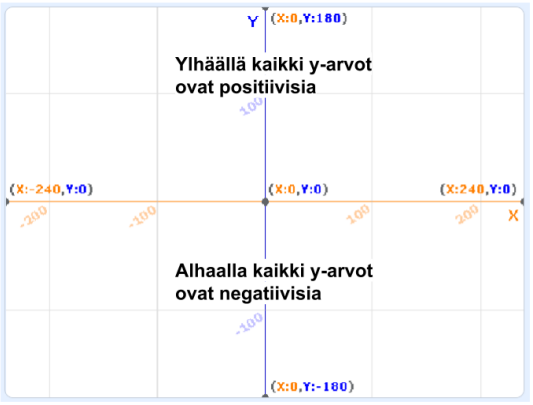


Luetaan – Koordinaatit

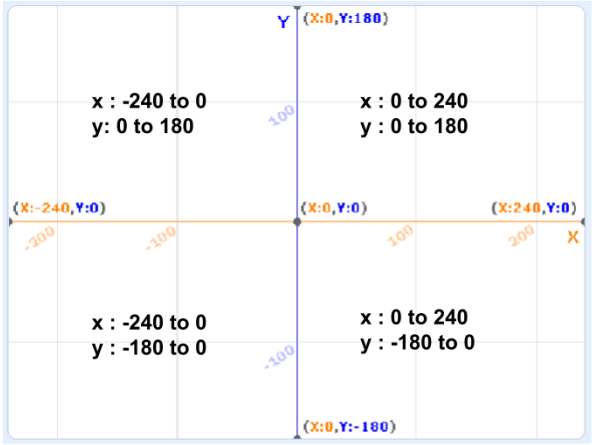
Scratchissa tehdyt ohjelmat näkyvät **esiintymislavalla**. Esiintymislava on kaksiulotteinen alusta, jonka toiminta perustuu **X-Y -koordinaatistoon**. Koordinaatisto koostuu ***X-akselista*** ja ***Y-akselista***.

X-akseli kulkee vasemmalta oikealle, ja se määrittää **esiintymislavan leveyden**, joka on **480 pikseliä**. Se on jaettu keskeltä negatiiviseen ja positiiviseen puoliskoon, ja sen vasen ääripiste on ***-240*** ja oikea ääripiste on ***240***.

Y-akseli kulkee alhaalta ylös, ja se määrittää **esiintymislavan korkeuden**, joka on **360 pikseliä**. Se on jaettu keskeltä negatiiviseen ja positiiviseen puoliskoon, ja sen alimmainen ääripiste on ***-180*** ja ylimmäinen ääripiste on ***180***. **Esiintymislavan keskipiste on *0* kummallakin akselilla.**



Koordinaatit ilmoitetaan kahdella luvulla *(x, y)*. **Ensimmäinen luku määrittää x:n**, ja **toinen luku määrittää y:n**. Esimerkiksi *(100, -120)* tarkoittaa, että **x:n arvo on 100** ja **y:n arvo on -120**.



Koordinaatteja käyttämällä voit **määrittää hahmojen tarkat sijainnit esiintymislavalla**.

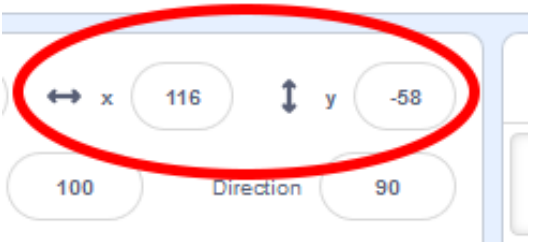
Harjoitellaan niiden käyttöä!

Esiintymislava voidaan jakaa neljään osioon.
X-akseli kulkee vasemmalta oikealle, -240 -> 240
Y-akseli kulkee alhaalta ylös, -180 -> 180

Ohjelmoidaan – Koordinoimaan opettelu

Ohjeet

1. **Aloita uusi Scratch projekti**. Lisää projektiin hahmo. Voit nähdä hahmon koordinaatit esiintymislavan alapuolelta.

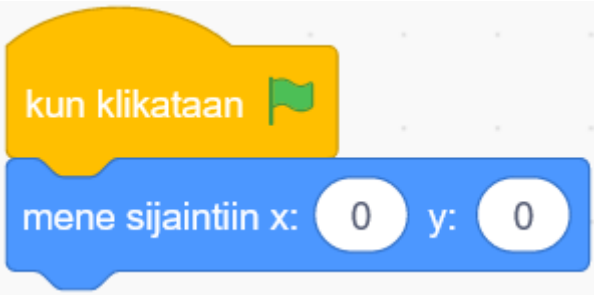


2. Liikuttele hahmoa ympäri lavaa. **X** ilmaisee sijainnin **vaakasuunnassa** ja **y** ilmaisee sijainnin **pystysuunnassa**.

Testaa ja tutki!

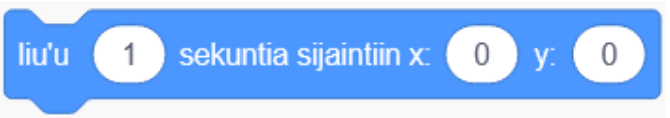
1. Tuo mene sijaintiin x: ( ) y: ( )-lohko ohjelmointialueelle. Sen avulla voit määrittää hahmon **x:n** ja **y:n** sijainnin.

Esimerkki →



2. **Muuttele x:n ja y:n arvoja** saadaksesi käsityksen siitä, miten koordinaatisto toimii.

3. Kokeile eri lohkoja Liike-valikosta jotka mahdollistavat **x:n** ja **y:n** arvojen muokkaamisen! Kokeile liu'u ( ) sekuntia sijaintiin x: ( ) y: ( ), lisää x:n arvoon ( ), lisää y:n arvoon ( ), aseta x: arvoksi ( ) ja aseta y:n arvoksi ( )-lohkoja.



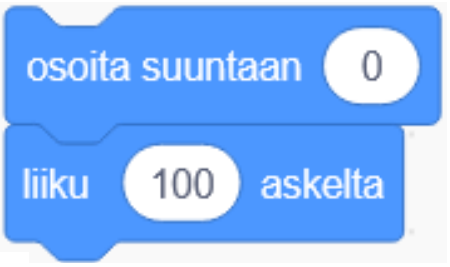
Testaa ja tutki!

Mene selaimella osoitteeseen codeschool.fi/pt7 ja pelaa Koordinaattipeliä!
Yritä ratkaista sokkelo käyttäen koordinaatteja!

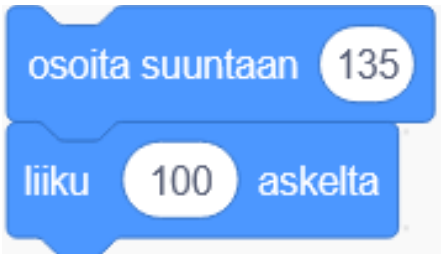
Huomautus: *Kun olet pelannut peliä, klikkaa* ***Katso sisälle*** *kuvaketta ja pohdi, kuinka peli toimii.*

X:n ja y:n arvojen muuttaminen

Olet varmasti jo liikuttanut hahmoasi esiintymislavalla käyttäen **askeleita**. Kun liikutat **hahmoa askeleilla**, **hahmo liikkuu** määrittämäsi luvun verran **pikseleitä sen osoittamaan suuntaan**. Hahmo voi liikkua ainoastaan eteenpäin.

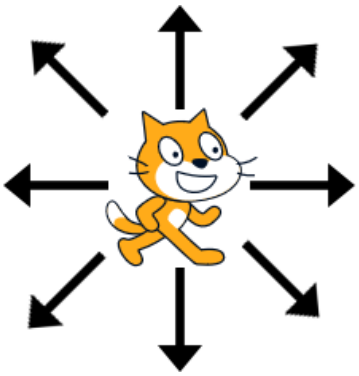


Liikuttaaksesi hahmoa **ylös**, hahmon täytyy osoittaa **ylöspäin**.



Että voisit liikkua **vinosuunnassa**, hahmokin täytyy **asettaa vinoon**.

Jos liikutat hahmoa muuttamalla x:n ja y:n arvoja, hahmon ei tarvi osoittaa siihen suuntaan, johon haluat sen liikkuvan. Voit liikuttaa hahmoa ympäri esiintymislavaa ilman, että hahmon täytyy kääntyä osoittamaan kulkusuuntaansa.



Ohjeet

1. **Mene selaimella tähän osoitteeseen: codeschool.fi/pt8**
2. Klikkaa **Katso sisälle** -nappia.
3. Tarkastele koodia. **Keskustele** parin kanssa **miten Hahmo1:n ja Hahmo2:n liikkumistyylit eroavat toisistaan**.
4. a) **Käytä nuolinäppäimiä** liikuttaaksesi Hahmo1:ä esiintymislavalla.

   b) **Käytä WASD näppäimiä** liikuttaaksesi Hahmo2:a esiintymislavalla.

   c) **Paina Z ja X näppäimiä**. Hahmot liikkuvat vinottain. Miten hahmojen liikkeet eroavat toisistaan?

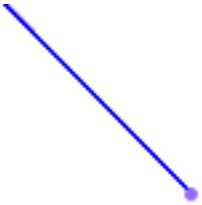
Testaa ja tutki!

Liiku esiintymislavalla **vaihtamalla X:n ja Y:n arvoja**!

1. **Aloita uusi Scratch projekti**. Tee skripti joka liikuttaa hahmoa:

   a) **sivuttaissuunnassa**

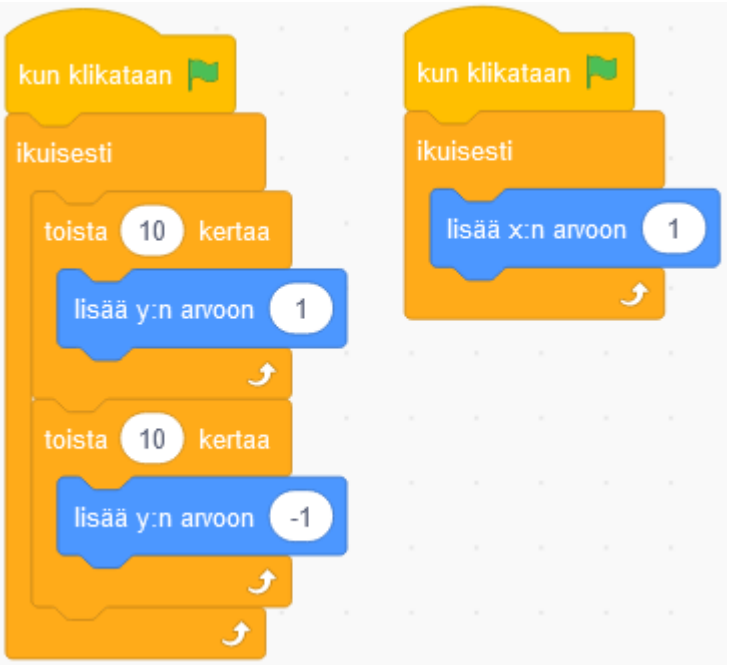
   b) **vinottain**



   c) **aaltoliikkeessä**

Vinkkejä hahmon liikuttamiseen x:n ja y:n avulla

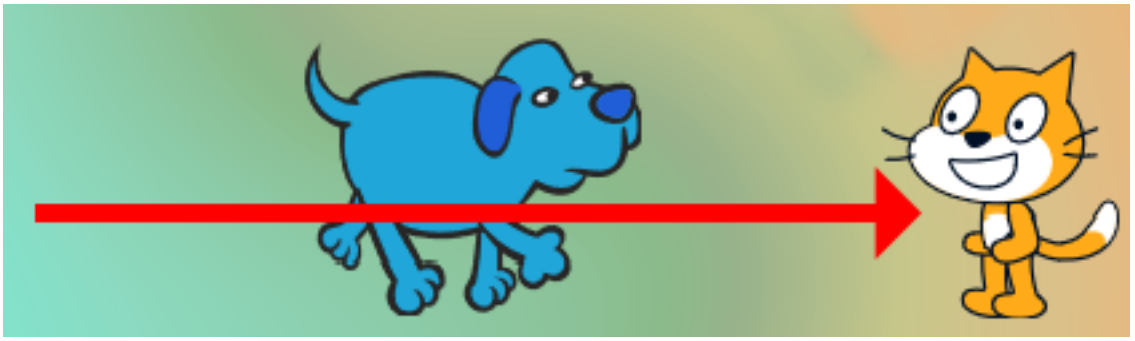
- **Saadaksesi aikaan aaltoliikkeen**, voit luoda kaksi eri skriptiä joista toisella muutat hahmon x-sijaintia ja toisella y-sijaintia. Kokeile itse!



Haaste!

Luo uusi Scratch projekti jokaista haastetta varten!

Haaste 1: Ohjelmoi **hahmo**, joka liikkuu esiintymislavan läpi **kohti toista hahmoa**. Kun se **koskettaa** toista hahmoa, sen liike **pysähtyy**.



Haaste 2: Ohjelmoi **hahmo, joka kysyy mihin sen tulisi liikkua**, ja liikkuu sitten vastauksen mukaisesti.

Käytä kysy ( ) ja odota-lohkoa:

Jos käyttäjä kirjoittaa "**ylös**", hahmo liikkuu ylös.
Jos käyttäjä kirjoittaa "**alas**", hahmo liikkuu alas.
Jos käyttäjä kirjoittaa "**vasen**", hahmo liikkuu vasemmalle.
Jos käyttäjä kirjoittaa "**oikea**", hahmo liikkuu oikealle.

Vinkkejä haasteisiin

- **Haastessa 1** käytä ikuisesti, jos < >, niin; tai muuten and lisää x:n arvoon ( )-lohkoja ja koskettaako ( )?-toimintolohkoa.

- **Haastessa 2** käytä **kysymistä** ja **vastauksen vertaamista**.

Projekti – Oppimispelin prototyyppi

Tehtäväsi on luoda **oppimispeli nuoremmille oppilaille**.

Opettajasi on keskustellut nuorempien oppilaiden opettajan kanssa ja sopinut teeman **oppimispelille**. Se saattaa olla esimerkiksi **kertolaskupeli** matikkaan, **lippuvisa** maantietoon, tai ehkäpä vain **tietovisa** johonkin muuhun oppiaineeseen. Käytä ohjelmointitaitoja, jotka olet oppinut kirjan **osan 1** aikana. Kun ohjelmoit peliäsi, käytä **kysymistä** ja **vastaamista**, **muuttujia**, **koordinaatteja** ja niin edelleen!

Tässä **osan 1 projektissa** luot oppimispelin **prototyypin**. Myöhemmin edetessäsi kirjan **osaan 2**, tulet **testaamaan peliäsi nuorempien oppilaiden kanssa**, keräämään **käyttäjäpalautetta** ja **jatkokehittämään peliäsi**.

Projektin vaatimukset:

- ❑ Pelissä on **vähintään kymmenen kysymystä**
- ❑ Käytä vähintään **kahta hahmoa**
- ❑ Liikuta **vähintään yhtä hahmoa koordinaattien avulla**
- ❑ Käytä **vähintään kahta eri muuttujaa** vastausmuuttujan lisäksi
- ❑ Ohjelmoi **pistelaskuri**
- ❑ Tee peliin **intro** tai **lopetus**

Tässä muutama esimerkkiprojekti, joita voit tutkia ja joista voit etsiä innostusta:

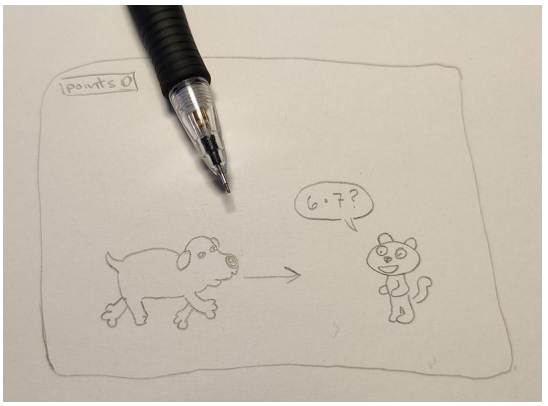
- Tietovisa: codeschool.fi/pt9
- Kertolaskupeli: codeschool.fi/pt10
- Lippuvisa: codeschool.fi/pt11

Ennen kuin aloitat ideoimaan ja tekemään peliä, lue pelin suunnittelusta ja luo oma suunnitelmasi! Jatka seuraavalle sivulle →

Suunnittelu

Suunnittelu on yhtä olennainen osa pelin tekemistä kuin itse **koodaaminen**. Suunnittelemalla huolellisesti, saat tehtyä pelistä **helppokäyttöisen** ja **houkuttelevan näköisen**. Lisäksi suunnittelu auttaa **itseäsi ymmärtämään paremmin, kuinka pelin tulisi toimia**. Hyvin toteutettu suunnittelu edellyttää pelin tarkastelua käyttäjän näkökulmasta. Tätä toimintaa kutsutaan **suunnitteluajatteluksi**.

Suunnitteluajattelu sisältää muutamia erilaisia prosesseja, kuten **kohderyhmän haastattelua**, **ongelmien etsimistä ja ratkaisemista**, **luonnostelua ja piirtämistä**, **mallintamista ja prototyyppien tekemistä** sekä **testaamista ja arvioimista**.



Luonnostele ja piirrä paperille, miltä peli voisi näyttää. On paljon kätevämpää tehdä luonnokset ensin paperille, kuin alkaa suoraa piirtämään Scratchissa. Pohdi valmiiksi esimerkiksi mihin eri napit tulevat, miltä hahmot näyttävät, miten hahmot liikkuvat, mihin asiat on sijoiteltu ja niin edelleen.

Mallinna ja tee prototyyppi. *Sanallista* tai *luo kaavio* kuvaamaan mitä pelissä tapahtuu. Tämä auttaa sinua hahmottamaan pelisi toimintaa kokonaisuudessa. Esimerkiksi kertolaskupelin **sanallistaminen** voisi olla jotain tämän tyyppistä:

> *Peli alkaa. Vihollishahmo liikkuu kohti toista hahmoa. Toinen hahmo kysyy kertolaskun. Jos käyttäjä vastaa kysymykseen oikein, vihollishahmo liikkuu hieman taaksepäin. Pisteet kasvavat, mitä kauemmin peli kestää. Kun vihollishahmo koskee toista hahmoa, peli päättyy.*

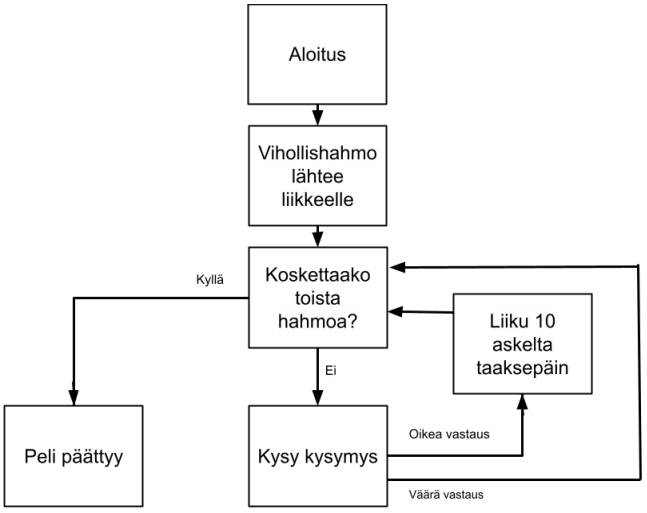
Ollaksesi vielä tarkempi, voisit **kuvailla vihollishahmon toimintoja** seuraavalla tavalla:

> *Kun peli alkaa, liikuta vihollishahmo aloituspisteeseen, ja aloita sen liikuttaminen toista hahmoa kohti sekä käynnistä kävelyanimaatio. Jos käyttäjä syöttää oikean vastauksen, liikuta vihollishahmoa taaksepäin 10 askelta. Kun vihollishahmo koskettaa toista hahmoa, lopeta peli.*

Ollaksesi vielä yksityiskohtaisempi, voit kuvailla toimintoja jopa näin tarkasti:

> *Vihollishahmo on animoitu vaihtamaan asustetta 0.66 sekunnin välein. Se liikkuu kohti toista hahmoa 0.5 askelta kerrallaan, ja toistaa tätä komentoa kunnes se koskettaa toista hahmoa. Se myös vaihtaa sen y-koordinaatin asemaa ylös ja alas tuoden kävelyanimaation aidomman näköiseksi.*

Voit myös luoda **kaavion** koodisi toiminnasta. Käytä laatikoita ja nuolia luodaksesi kaavion josta ilmenee, kuinka koodi toimii.



Näitä toimia kutsutaan **mallintamiseksi**. Aloita pelisi tekeminen näistä, ja siirry vasta sen jälkeen muuttamaan kirjoittamasi teksti tai piirtämäsi kaavio oikeaksi ohjelmaksi. Sitä kutsutaan puolestaan **prototyypin tekemiseksi**.

Anna toisten oppilaiden kokeilla peliäsi sen ollessa vielä kesken. Kysy sitten heidän mielipiteitään. Mitä he muuttaisivat pelissä? Mitä mieltä he olivat pelin ulkonäöstä? Tee **kysely** ottaaksesi selville pelin kehitystarpeet. Kysy opettajalta valmista kyselypohjaa tai vinkkejä oman kyselyn tekemiseksi.

Voit nyt aloittaa oppimispelin suunnittelun!
Suunnittelupaperi löytyy seuraavalta sivulta →

Tee suunnitelma

Ryhmän jäsenet:

Kuvaile peliä:

_____

_____

_____

_____

Tarvitsemme apua ohjelmoinnissa	Kyllä	Ei	Ehkä
	○	○	○

Luonnostele ja piirrä miltä peli näyttää:

Valmistaudu esittelemään suunnitelmasi opettajalle ja muille oppilaille!

Sanallista tai **piirrä kaavio** pelisi toiminnasta:

Valmistaudu esittelemään suunnitelmasi opettajalle ja muille oppilaille!

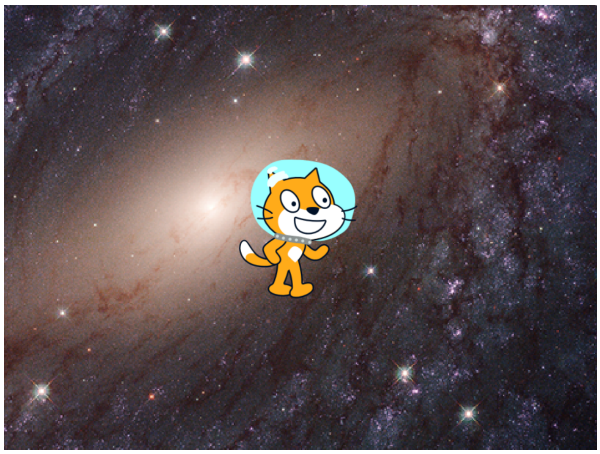
Voit nyt alkaa luomaan ja ohjelmoimaan!

💡 Vinkkejä oppimispelin tekemiseen

Seuraavilta sivuilta löydät **vinkkejä** jotka auttavat sinua **lisäämään mielenkiintoisia elementtejä peliin**. Käytä niitä vapaasti!

Teema

Mieti pelillesi teema. **Voit pyytää ideoita teemaa varten nuoremmilta oppilaita**, joille peli tehdään.



Esimerkiksi **avaruusteemaisessa pelissä** voit käyttää **avaruustaustaa** ja **astronauttihahmoa!**

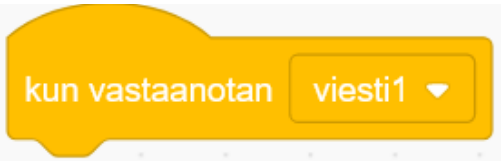
Sinun ei kuitenkaan kannata vielä tässä vaiheessa tehdä lopullista päätöstä hahmojen ulkonäöstä. **Odota kirjan osaan 2, jossa opetellaan hahmojen animointia**.

Lähettäminen ja vastaanottaminen

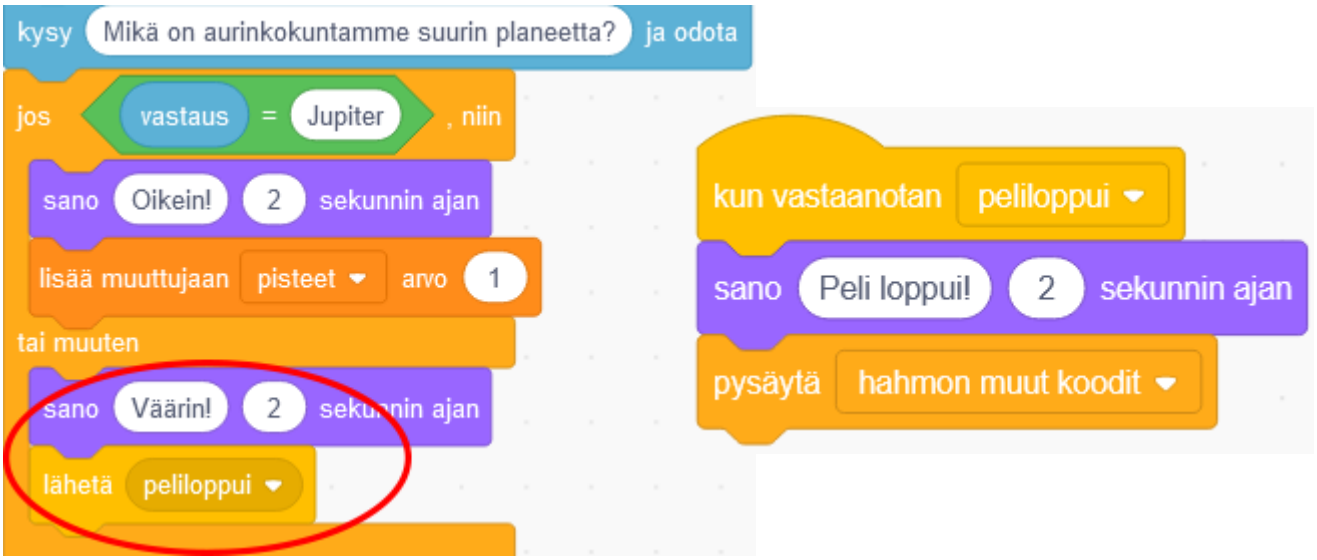
Lähetykset ovat viestejä, joiden avulla hahmojen eri skriptit voivat keskustella keskenään. Niiden avulla voit **käynnistää toisia skriptejä**, joissa on **lähetystä** vastaava **ensimmäinen lohko**. Niitä käytetään erilaisten pelitapahtumien käynnistämiseen, kuten silloin kun peli päättyy.

Lähetä (viesti1)-lohko **lähettää viestin**.

Kun vastaanotan (viesti1)-lohko **vastaanottaa viestin** ja **käynnistää sen alle rakennetun skriptin**.



Jos vastaus on **väärä** alla olevassa esimerkissä, ohjelma sanoo “Väärin!” ja **lähettää viestin “loppu”**.



Kun kun vastaanotan (loppu)-lohko viereisessä skriptissä **vastaanottaa viestin “loppu”, se käynnistää sen alla olevan skriptin**.

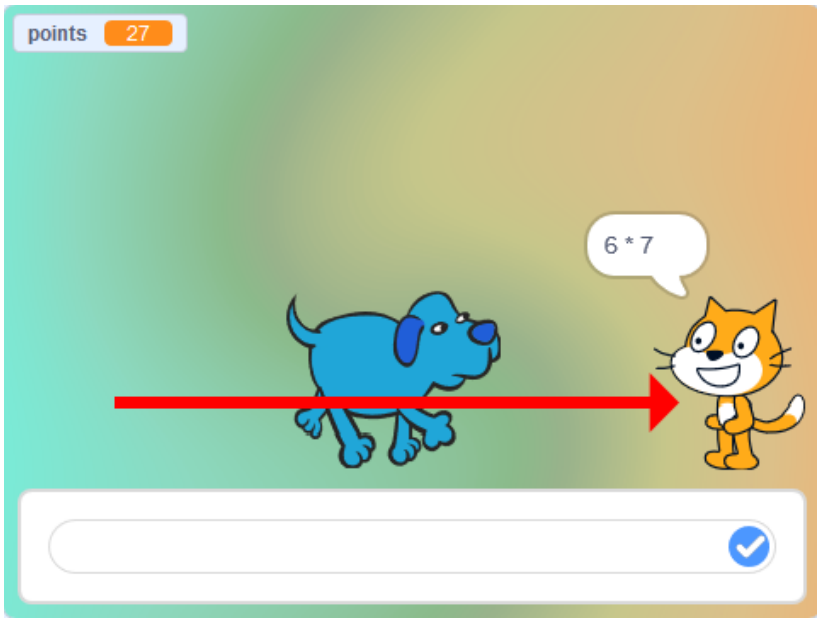
Vihje: *pysäytä [hahmon muut koodit]-lohkolla voidaan pysäyttää kaikki muut hahmon ohjelmassa olevat koodit. Sitä kannattaa hyödyntää esimerkiksi silloin, kun haluat pelin loppuvan.*

Jännitys

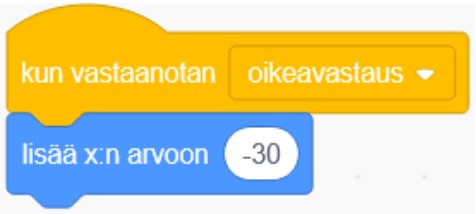
Tehdäksesi pelistä jännittävän, siihen kannattaa lisätä elementtejä jotka aiheuttavat **paineen tunnetta** tai **antavat mahdollisuuden saavuttaa huippupisteet**.

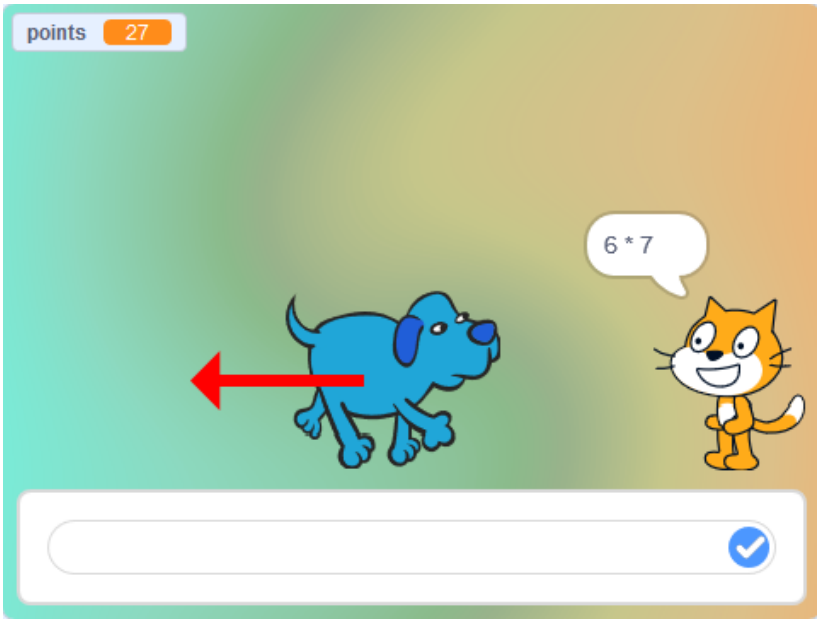
Voit lisätä peliin esimerkiksi **ajastimen**. Se voi olla vain alaspäin laskeva aika...

...tai esimerkiksi vihollishahmo, joka liikkuu toista hahmoa kohti. Jos pelissä on mahdollista **hävitä**, se tekee pelaamisesta jännittävämpää, koska silloin myös **voittaminen** on paljon hienompi saavutus.



Voit myös **palkita** pelaajaa oikeista vastauksista. Joka kerta kun pelaaja vastaa oikein, voisi peli antaa hieman lisäaikaa ajastimeen, tai siirtää uhkaavaa vihollishahmoa hieman taaksepäin.

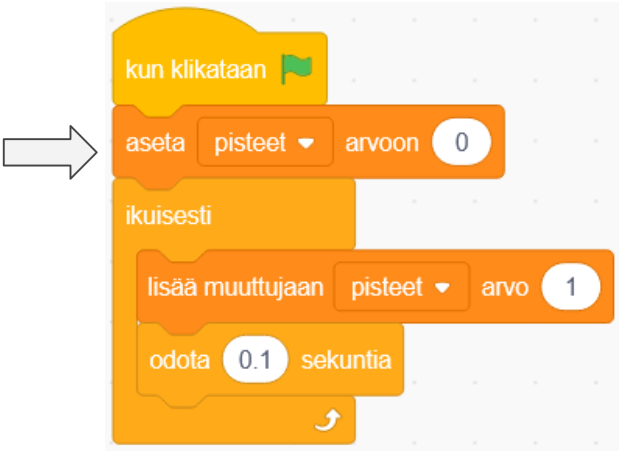




Pistelaskuri

Kilpailulliset pelit ovat myös jännittävämpiä! Lisää peliin **pistelaskuri**. Pisteitä voisi kertyä esimerkiksi **oikeista vastauksista**, **kuluneesta ajasta** tai **jostain ihan muusta**! Luo **muuttuja**, joka laskee pisteet.

Tässä esimerkissä skripti muuttaa pisteet-muuttujan arvoa **yhdellä** aina 0.1 sekunnin välein. Pisteet alkavat rullaamaan heti kun kuvaketta klikataan.



Voit myös **antaa pisteitä oikeista vastauksista** ja **ottaa niitä pois vääristä vastauksista**.

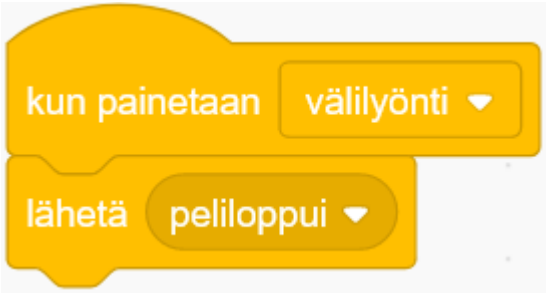
Ohjelmoi peli ilmoittamaan pelaajan pisteet pelin päätyttyä. Käytä yhdistä-toimintolohkoa sano-lohkon sisällä yhdistääksesi **pistemuuttujan arvon** jonkinlaiseen tekstiin!

*Keksitkö miten voisit liittää myös **pelaajan nimen** tähän?*

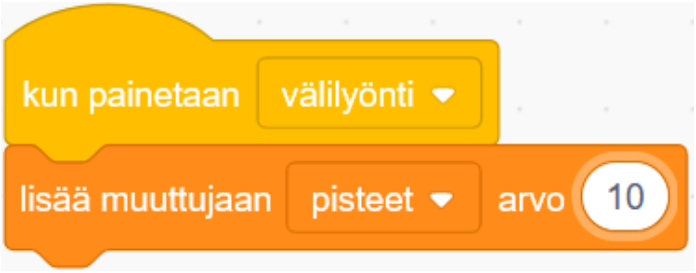
Pikanäppäimet ja oikotiet

Kun ohjelmoit erilaisia **tapahtumia** peliin, jotka **laukaisee** jokin tietty toiminto, **lisää oikoteitä** voidaksesi testailla tapahtumien toimivuutta helpommin. Oikotiet voivat olla juttuja, jotka mahdollistavat **pisteiden saamisen** nopeammin tai helpottavat muuten vain pääsyä **tapahtumien alkuun**.

Jos pelissäsi on **lopputapahtuma** jonka toimintaa haluat testata, on äärimmäisen turhauttavaa pelata koko peli alusta alkaen joka kerta, kun haluat testata sitä.



Lisää oikotie lopputapahtumaan käyttäen esimerkiksi **pikanäppäintä**, jolla tapahtuma käynnistyy välittömästi.



Jos pelissä tapahtuu jotain, kun **muuttuja** saavuttaa tietyn arvon, **lisää oikotie jolla saavutat halutun arvon välittömästi**.

Luomalla erilaisia **oikoteitä** peliisi, on sen testaaminen paljon kätevämpää!

Klikkaamalla esiintymislavalla näkyvää muuttuja-kuvaketta, voit muuttaa sen esitystapaa.

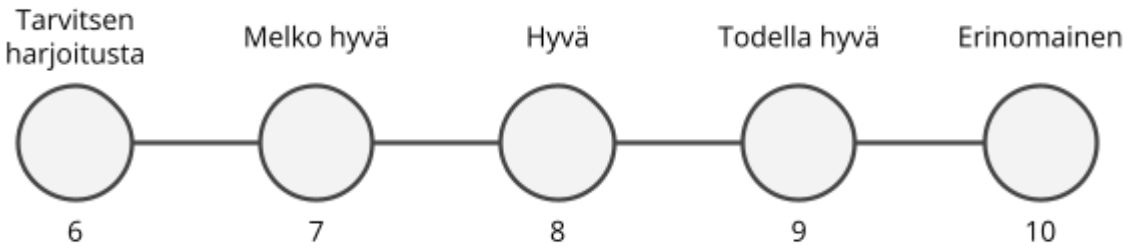
Oikotiet ovat vain testaamista varten! Muista siivota tai deaktivoida oikotiet valmista peliä varten!

Itsearviointi - Osa 1

Vastaa seuraaviin kysymyksiin tehtyäsi kaikki osan 1 harjoitukset:

1. Arvioi oma osaamisesi arviointiperusteiden mukaan.

   Keskustele arvioinnista parin kanssa!



2. Mieti opiskeluasi. Mistä olet erityisen ylpeä?

   _____

   _____

3. Mieti opiskeluasi. Mitä tekisit eri tavalla ensi kerralla?

   _____

   _____

4. a) Lopuksi – Alussa sait kuvitella **unelmiesi ohjelman**. Oletko työskennellyt niin kovasti, että **jonain päivänä voisit jopa itse ohjelmoida tuon unelmiesi ohjelman**? Keskustele parisi kanssa.

   b) Mieti miltä **omat ohjelmointitaitosi nyt tuntuvat** ja **ota selfie** jossa tämä tunne näkyy!

Osa 2

Oppimispelin kehittäminen

Pelejä, ohjelmistoja ja sovelluksia tehdään ensisijaisesti ihmisten käyttöön. Tämän vuoksi onkin tärkeää pyytää **käyttäjiltä palautetta**. Ovathan käyttäjät suurin syy, miksi edellämainittuja alun alkaenkaan tehdään.
Kysymällä käyttäjiltä mistä he pitävät tai eivät pidä, saat arvokasta tietoa siitä, miten sovellusta kannattaa **kehittää**. Yleisesti eri ohjelmistojen työstäjiä kutsutaankin **developereiksi** tai **devaajiksi**, joka tarkoittaa suomennettuna **kehittäjää**.

Osassa 2 opit toistorakenteiden ja ehtolauseiden sisäkkäin asettelusta, hahmojen piirtämisestä ja animoinnista, käyttäjäpalautteen keräämisestä ja paljon muuta!

Aloitetaan - Valmistautuminen osaan 2

1. Ajattele taas itsellesi mieluisia sovelluksia ja pelejä. Mitä niissä esiintyviä elementtejä haluaisit **tuoda omaan peliisi**?

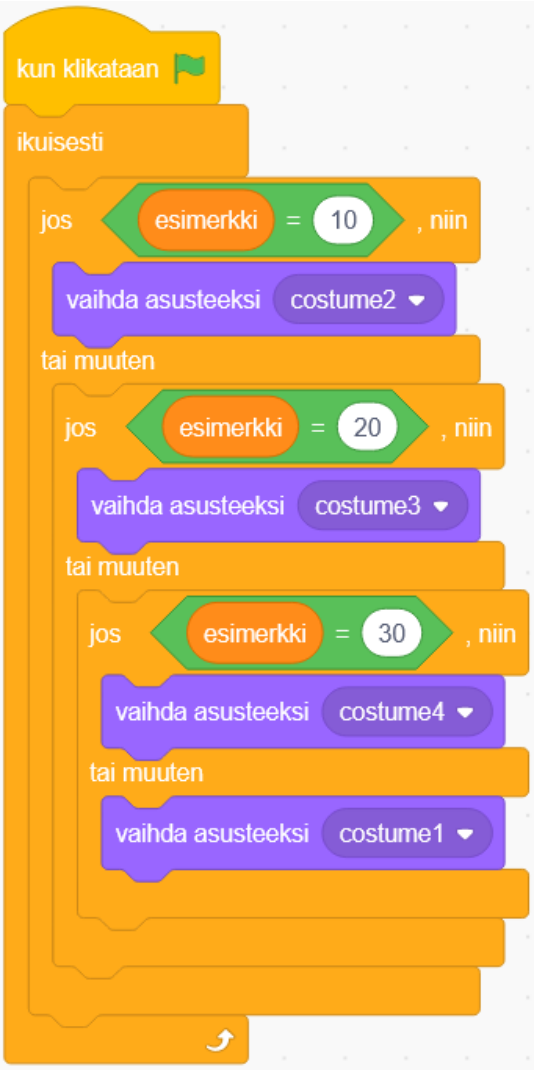
   Kirjoita ja **piirrä** alle:

2. Tee **itsellesi yksi lupaus** opiskeluun liittyen. Esimerkiksi: “Kysyn apua aina, kun tarvitsen sitä”.

   → **Tallenna tämä lupaus puhelimeesi** tai **kirjoita** se tähän! Yritä **muistaa** ja **pysyä** itsellesi tekemässä lupauksessa.

Luetaan – Sisäkkäiset toistot & ehdot

Toistot ja ehdot voidaan asetella **sisäkkäin**. Se tarkoittaa, että **toiston sisällä voi olla toinen toisto, jonka sisällä on ehto jonka sisällä on toisto**... ja niin edelleen. Yksi yleisimmin käytetty **sisäkkäin asettelun tyyli** joita käytämme tämän kirjan projekteissa on asettaa **ehtoja ja toistoja** yhden **ikuisesti-toiston sisälle**. Koska ikuisesti-lohkoilla tehtävät toistot mahdollistavat *ikuisesti* kestävän toiston skripteille, niiden käyttö on yleistä.



Oikealla olevassa esimerkissä jos < >, niin; tai muuten-lohkot on aseteltu ikuisesti-lohkon sisään. Tällä tavalla tehty skripti **tarkistaa ikuisesti esimerkki-muuttujan arvoa**.

Jos skriptissä **ei** käytettäisi ikuisesti-lohkoa, ohjelma tarkistaisi esimerkki-muuttujan arvon **yhdesti** ohjelman käynnistämisen jälkeen, jonka jälkeen skripti loppuisi.

Viereisessä esimerkissä myös **ehdot on aseteltu sisäkkäin**. Asettamalla ehdot näin, skripti saadaan toimimaan siten, että se **tarkistaa onko esimerkki-muuttujan arvo 10**. Jos se ei ole, skripti tarkistaa onko sen **arvo 20**. Jos tämäkään ei pidä paikkaansa, skripti jatkaa eteenpäin ja tarkistaa, onko **arvo 30**. Jos mikään näistä ehdoista ei ole **tosi**, skripti jatkaa lopussa olevaan **tai muuten**-osioon.

Sisäkkäin asettelu...

tarkoittaa ohjauslohkojen, kuten silmukoiden tai ehtolauseiden sijoittaminen **toisten lohkojen kanssa sisäkkäin**.

Ohjelmoidaan – Silmukoiden ja ehtolauseiden sisäkkäin asettelu

Ohjeet

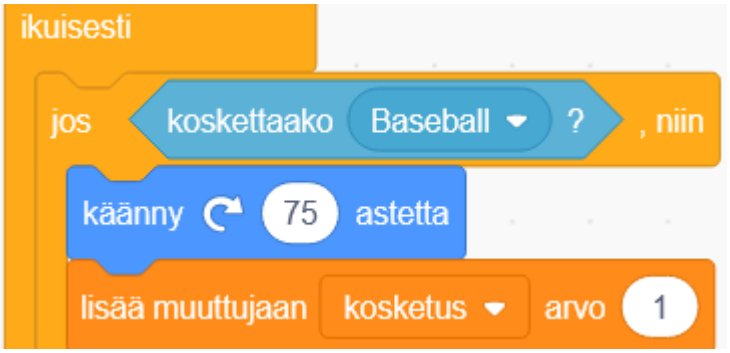
1. **Mene selaimella tähän osoitteeseen: codeschool.fi/pt12**
2. Klikkaa **Katso sisälle** -nappia.
3. **Sano hei "Mr. Nosey":lle!** Sinun tehtäväsi on ohjelmoida hänet **pysymään ympyrän seinämien sisäpuolella** ja **reagoimaan ympäriinsä lentelevään pesäpalloon** käyttäen **sisäkkäin aseteltuja ehtoja**.

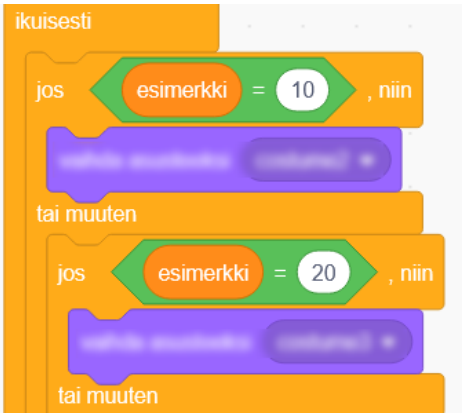
Haaste!

Haaste 1: Ohjelmoi Mr. Nosey **kääntymään 75 astetta osuessaan pesäpalloon** tai **koskettaessaan mustan ympyrän reunoja**. Keksi myös itse **yksi ehto lisää** ja aseta se **sisäkkäin aiempien ehtojen kanssa**. Jos mikään ehto **ei täyty**, laita Mr. Nosey kulkemaan suoraa eteenpäin.

Lohkot on aseteltu valmiiksi ohjelmointialueelle. Tehtäväsi on **asetella ne oikeaan järjestykseen**.

Haaste 2: Ohjelmoi **muuttuja** joka **laskee kosketukset**. Aina kun Mr. Nosey koskettaa mustaa ympyrän reunaa tai pesäpalloa, ohjelman tulee **korottaa kosketus-muuttujan arvoa yhdellä**.





Seuraavaksi luo **sisäkkäin aseteltuja ehtolauseita** käyttäen skripti, joka tarkistaa **onko kosketus-muuttujan arvon yhtä suuri kuin 10, 20 tai 30**. Ohjelmoi **jotain tapahtumaan**, jos ehto täyttyy!

Ohjeet

1. **Mene selaimella tähän osoitteeseen: codeschool.fi/pt13**
2. Klikkaa **Katso sisälle** -nappia.

   Tehtäväsi on **ohjelmoida Mr. Nosey piirtämään erilaisia kuvioita käyttäen sisäkkäin aseteltuja toistoja**.

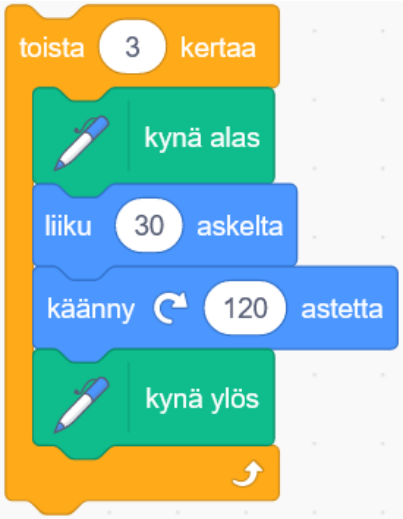
Haaste!

Haaste 3: Ohjelmoi Mr. Nosey piirtämään **kolmioita**, **neliöitä**, **viisikulmioita** ja **kuusikulmioita** käyttäen **sisäkkäisiä toistoja**.

Lohkot on aseteltu valmiiksi ohjelmointialueelle. Tehtäväsi on **asetella ne oikeaan järjestykseen**.

Vinkkejä haasteisiin

- **Haasteessa 1** Mr. Nosey saattaa jäädä jumiin. Muuta käänny ( ) astetta-lohkon arvoja jos niin tapahtuu, tai käynnistä ohjelma uusiksi!
- **Haasteessa 2** voit käyttää lohkoja Ulkonäkö-valikosta.
- **Haasteessa 3** tehtäväsi on **piirtää kuvioita käyttäen toistoja**. Piirtääksesi **kolmion**, käytä liiku ( ) askelta ja käänny ( ) astetta-lohkoja toista ( )-lohkon sisällä.



Piirtäminen tehdään **yksi sivu kerrallaan**.

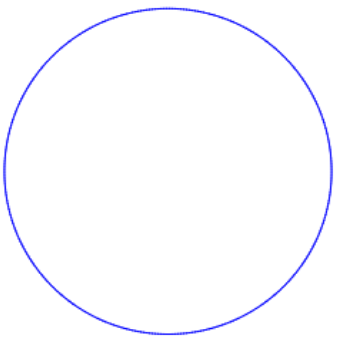
Kolmiossa on **kolme sivua**, joten aseta arvo **3** toista ( )-lohkoon.

Valitse haluamasi määrän askelia liiku ( ) askelta-lohkoon. Sivun **pituudella** ei ole vaikutusta lopputulokseen, kunhan pituus ei ylitä esityslavan kokoa. Esimerkkikuvassa sivun pituudessa käytetään **30** askelta.

Lopuksi täytyy vielä tietää **kuinka monta astetta** meidän tulee kääntyä. **Jaa luku 360 muodossa olevien kulmien määrällä**. Kolmiossa on **kolme kulmaa**, joten **luku 360 jaetaan luvulla 3**, jolloin osamääräksi jää luku **120**. Tämä luku syötetään käänny ( ) astetta-lohkoon. *(360 / 3 = 120)*

Miksi jaamme luvun 360?

Kun kuljetaan kokonainen kierros, käännytään aina yhteensä **360** astetta. Jokaisen muodon, joka kulkee kokonaisen kierroksen ympäri, **kulmien asteiden yhteenlaskettu summa on aina 360**. Esimerkiksi **suorakulmaisessa neliössä** on **neljä 90 asteen kulmaa**, jotka yhteenlaskettuna muodostavat luvun 360. *(4 x 90 = 360)*



Ohjelmoidaan – Asusteet & animointi

Kun olet päättänyt pelisi teeman, **on vuorossa hahmojen luonti ja animointi**. Animaatioiden avulla saat pelistäsi näyttävän ja valmiin tuntuisen.

Animointi Scratchissa tehdään **vaihtamalla hahmojen asusteita**. Animaation jokaisen tapahtuman täytyy siis olla **oma asusteensa**. Voit luoda animaatioita **pelin eri tapahtumia** varten. Yksinkertainen kävelyanimaatio voidaan tehdä käyttämällä **kahta eri asustetta**.

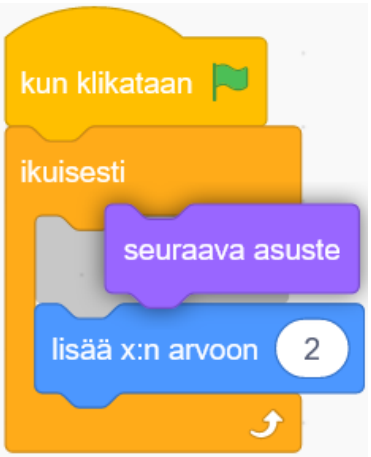
Seuraavaksi harjoitellaan animointia. **Luo uusi Scratch projekti** ja anna sille nimeksi *"animointiharjoitus"*.

Ohjeet

1. Tuo **Dog2** hahmo ohjelmaasi

2. Dog2 hahmolla on valmiiksi tehtyjä asusteita; **kaksi jotka esittävät kävelevää koiraa** ja yksi jossa se näyttää miettivän jotain. **Poista** viimeksi mainittu asuste.

3. **Luo skripti**, joka laittaa hahmon liikkumaan ja **vaihtamaan** asustetta.



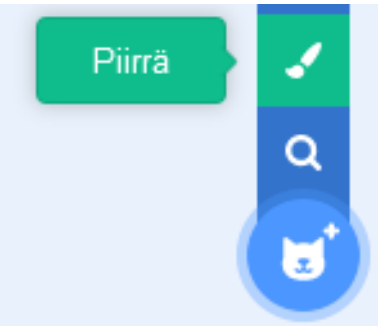
4. Kun käynnistät skriptin, hahmo liikkuu ja näyttää kuin se kävelisi. Sen jalat liikkuvat hurjaa vauhtia!

5. Voit hallita animaation toistonopeutta tekemällä sille **erillisen skriptin** kuten viereisessä esimerkissä. Lisää odota ( ) sekuntia-lohko ja muutan odotuksen sekuntimäärää hallitaksesi toiston nopeutta.

Nyt voit aloittaa **luomaan omia pelihahmoja**!

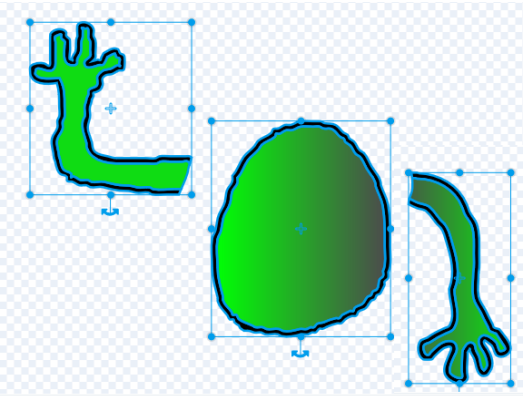
Testaa ja tutki!

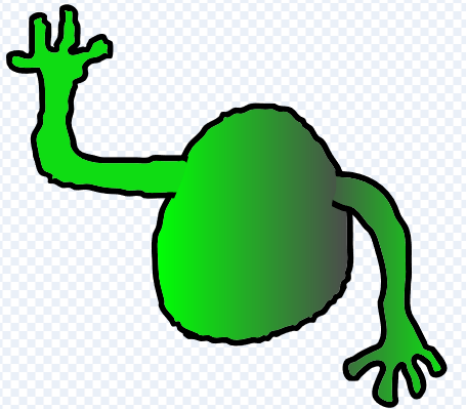
1. Avaa **peliprojektisi** Scratchissa.
2. Siirrä hiiri hahmonluontikuvakkeen päälle ja valitse **Piirrä**.



Vinkkejä hahmojen piirtämiseen

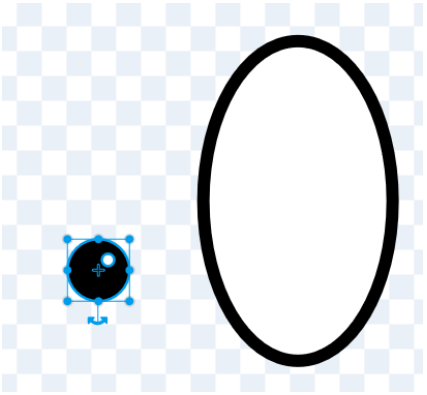
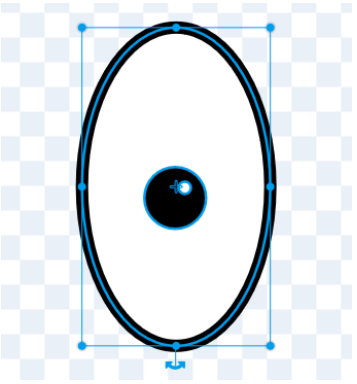
➪ Kun luot hahmoja, piirrä osat joita haluat liikutella animaatioissa **erillisinä osina**. Tällä tavoin helpotat osien liikuttelua ja muokkaamista kun aloitat animoinnin.





- Voit **ryhmittää** eri osia, jolloin ne tarttuvat toisiinsa ja pysyvät yhdessä.

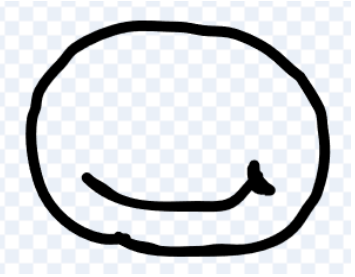
  Voit aina halutessasi myös **purkaa ryhmityksen**.

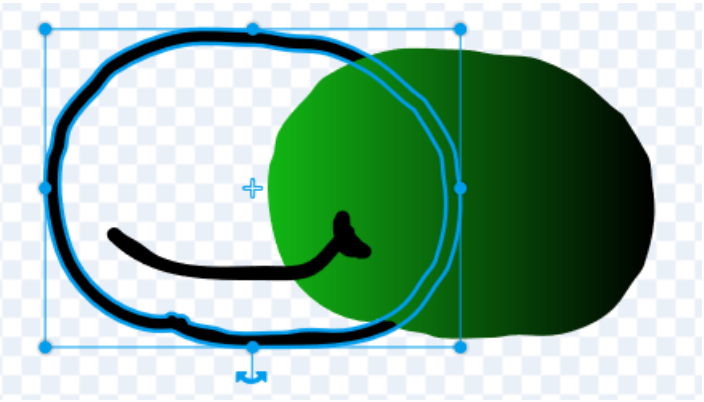
- Voit poistaa ryppyiset viivat **muotoilutyökalulla**. Klikkaa ylimääräisiä muotoilupisteitä ja poista ne. Voit myös muotoilla hahmoa erilaiseksi muotoilutyökalun avulla.


- Piirrä aina ensin hahmon ääriviivat **siveltimellä**. Sen jälkeen **täytä** muoto käyttäen **täyttötyökalua**.

  ***Huomautus**: Kun käytät täyttötyökalua, kokeile eri täyttöjä valikosta!*



Kun olet ensin piirtänyt ääriviivat ja sen jälkeen täyttänyt muodon, **ryhmitä** ne yhteen. Muuten ääriviivat ja täyttö jäävät kahdeksi erilliseksi osaksi.



Käytä mielikuvitustasi ja pidä hauskaa!

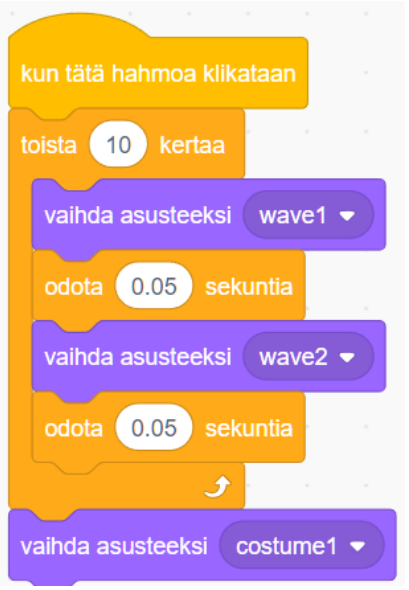
Hahmojen animointi

Scratchissa animaatiot tehdään **luomalla useita hieman erilaisia asusteita** hahmoille ja sitten **vaihtamalla asusteita skriptin avulla**. Animoinnissa pätee nyrkkisääntönä **mitä useampi asuste**, **sitä sulavampi animaatio**. Joskus sillä on väliä, joskus taas ei.

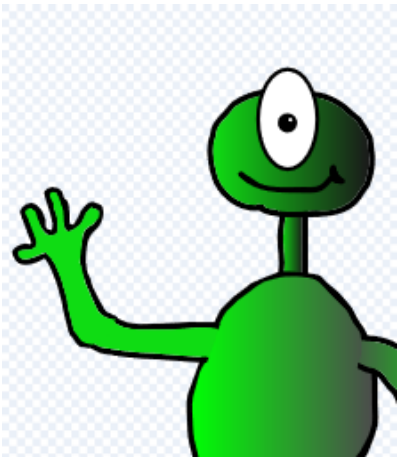
Voit animoida **liikuttelemalla hahmon osia** tai **muotoilemalla niitä**. Harjoitellaanpa!

Ohjeet

1. **Mene selaimella tähän osoitteeseen: codeschool.fi/pt14**
2. Klikkaa **Katso sisälle** -nappia.
3. Valitse Alien-hahmo ja avaa sen **asusteet**-välilehti. Vilkuile hahmolle tehtyjä eri asusteita.



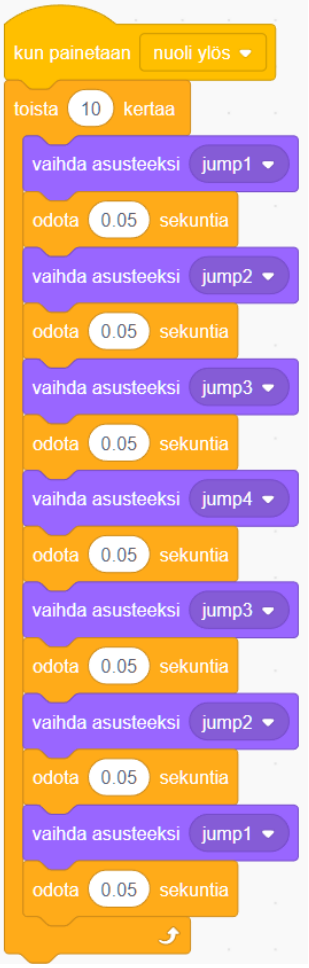
4. Voit käynnistää **vilkutusanimaation** klikkaamalla hahmoa esiintymislavalla. Se on tehty käyttämällä **kahta eri asustetta** joita toistetaan skriptin avulla. Odota ( ) sekuntia-lohkojen avulla hallitaan animaation nopeutta.



5. Paina välilyöntiä. Hahmo alkaa heiluttaa raajojaan kuin se tekisi **X-hyppyjä**. Katso asusteista, kuinka animaatio on tehty käyttäen **seitsemää eri asustetta**.

Asusteet **hyppy1** - **hyppy 4** muodostavat animaation puoliväliin asti, kun taas asusteet **hyppy5** - **hyppy7** ovat kopioita aiemmista asusteista, jotka on aseteltu päinvastaiseen järjestykseen. Näin animaatiosta **kiertävä** ja skriptistä lyhyempi.

Viereisessä skriptissä on käytetty **sisäkkäisiä toistoja**. Skripti alkaa vaihda asusteeksi (hyppy1) ja sitten toistaa **kuusi kertaa** seuraava asuste-lohkon ja lyhyen tauon. Tällä tavalla **hahmon asuste vaihtuu hyppy1:stä 6 kertaa päätyen lopulta asusteeseen hyppy7**. Sen jälkeen skripti alkaa taas alusta vaihtaen asusteeksi **hyppy1**. Skripti toistaa animaation 10 kertaa toista (10) kertaa-lohkon avulla.



Jos käytettäisiin pelkästään asusteita **hyppy1** - **hyppy 4**, skripti täytyisi rakentaa hieman erilaiseksi. Viereisestä esimerkistä huomaat, kuinka skriptistä tulisi tällöin paljon pidempi. Joskus on siis **käytännöllisempää kopioida asusteita ja järjestellä ne uusiksi**.

6. Piirrä Alienille **silmänräpäytys animaatio**. Tee kopio **asuste1:stä** ja laita Alien räpäyttämään silmäänsä käyttämällä **muotoilutyökalua**.

7. **Rakenna skripti** silmänräpäytys animaatiolle.

Testaa ja tutki!

Nyt on aika aloittaa **omien hahmojesi animointi**!

1. Käytä oppimiasi taitoja animoidaksesi pelissäsi olevia hahmoja.
2. Voit luoda animaatioita esimerkiksi **liikkumista**, **loppua** ja **muita pelin tapahtumia** varten.
3. Käy läpi alla oleva lista vinkeistä animointiin

Vinkkejä animointiin

➪ **Luo varmuuskopioita** hahmoista. Aina silloin tällöin asusteita muokatessa jokin menee pieleen. Jos näin käy, voit palata alkuperäiseen hahmoon varmuuskopion avulla. Tee ainakin yksi varmuuskopio!

➪ Jos sinulla on **kiertävä animaatio**, kuten aiempi esimerkki X-hyppäävästä Alienista, **voit luoda kopioita aiemmasta asusteista ja vaihtaa sitten asusteiden järjestystä listalla**. Näin sinun ei tarvitse luoda jokaista asustetta uusiksi.

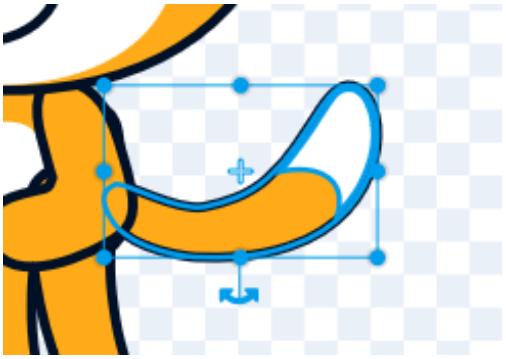
➪ Kaikkea liikettä ruudulla ei tarvitse tehdä asusteiden avulla! Esimerkiksi X-hyppäävä Alien **kellui ympäri avaruutta** yksinkertaisen skriptin avulla.

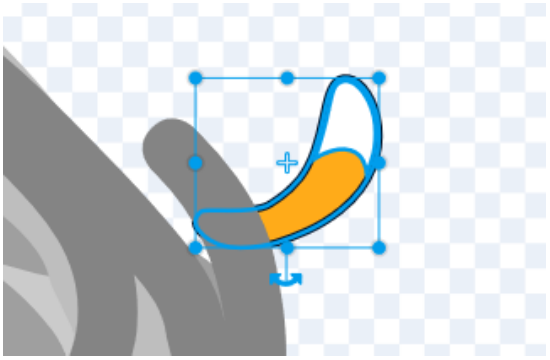
- Voit käyttää animaatioissa myös ihan jotain muuta, kuin hahmon alkuperäisiä asusteita!

  Mene selaimella osoitteeseen codeschool.fi/pt15 ja **katso mitä käy kun hahmot koskettavat toisiaan**.

Katso koodia. Kun hahmo **Dog2** saavuttaa hahmon **Cat** se laukaisee skriptissä animaatiom. Koko animaatio on toteutettu **Dog2:n** asusteissa. **Cat** hahmo piilotetaan skriptin avulla, mutta näyttää silti siltä, kuin molemmat hahmot olisivat yhä esiintymislavalla.

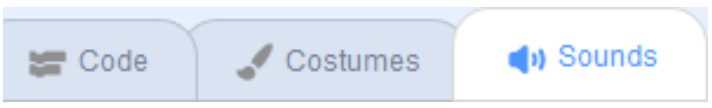
***Huomautus**: Jos piilotat hahmos, muista tuoda se takaisin näkyviin käyttäen näytä-lohkoa aina ohjelman alussa!*





Jotkin **Cat** hahmon osat on kopioitu **Dog2** asusteisiin, joten animaatio näyttää siltä kuin molemmat hahmot olisivat yhä näkyvissä. Siistiä, eikö?

- **Lisää äänitehosteita peliisi**! Kuten aiemmassa esimerkissä kissan ja koiran tappelusta, ääniefektit tekevät animaatiosta eläväisemmän. Voit luoda ja muokata ääniä **Äänet**-välilehdestä.

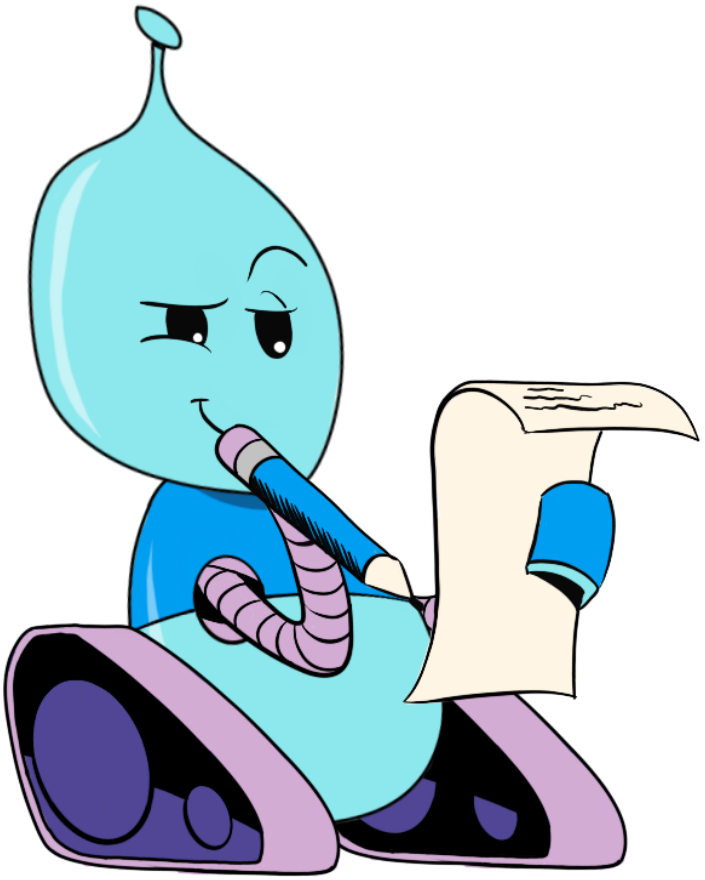


Testaaminen ja palautteen kerääminen

Nyt sinulla on pelistäsi **prototyyppi**, joka on valmis käyttäjien **testattavaksi** ja **arvioitavaksi**. Aika kerätä siis **palautetta**!

Anna kohderyhmän oppilaiden pelata peliäsi. Sen jälkeen **kerää heiltä palautetta** pelistä. Että palautteen antaminen olisi heille helpompaa, tee heille **palautelomake** täytettäväksi.

Kun suunnittelet palautelomaketta, **mieti, mitä haluaisit tietää pelistäsi pelaajan näkökulmasta**. Voit kysyä esimerkiksi **oliko peli liian vaikea tai helppo**, **oliko teema mieluisa**, **mistä käyttäjät pitivät tai eivät pitäneet**. Jätä tilaa myös **vapaalle sanalle** että pelin arvioija voi antaa palautetta myös niistä osista peliä, jota et hoksannut kysyä!

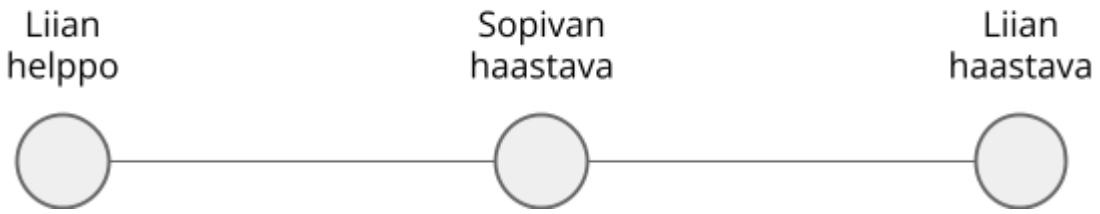


Esimerkki palautelomakkeesta seuraavalla sivulla →

Palautelomake

Nimet:

Millainen pelin vaikeusaste oli mielestäsi?



Mistä pidit pelissä? Mistä et pitänyt?

_____

_____

Mitä lisäisit peliin? Entä mitä ottaisit siitä pois?

_____

_____

Vapaa sana

_____

_____

Miltä haluaisit pelin näyttävän? Kirjoita tai piirrä! (teema)

Projekti – Oppimispelin versio 1.0

On aika saatella peliprojekti loppuun. Käytä kirjan **osan 2** aikana oppimiasi uusia taitoja, käy läpi nuoremmilta oppilailta saamasi **palaute** ja **viimeistele pelisi**!

Lisävaatimuksia projektille:

Kehitä peliä **saamasi palautteen pohjalta**

Tee **omia hahmoja** peliisi

Lisää **animaatioita peliin**

Tee pelillesi **intro**

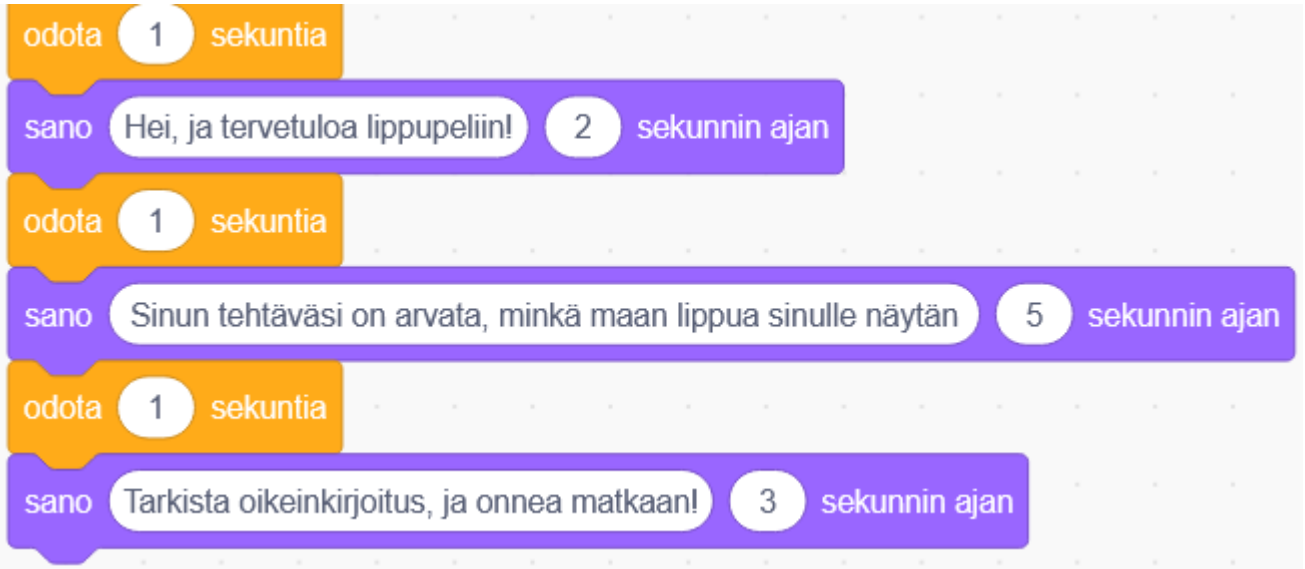
Voit nyt aloittaa pelisi viimeistelyn!

💡 Vinkkejä pelin kehittämiseen

Seuraavilta sivuilta löydät **vinkkejä** jotka auttavat sinua **kehittämään peliäsi**. Käytä niitä vapaasti!

Intro

Voit lisätä peliisi **intron**, jossa kerrot pelaajalle **miten peliä pelataan**, **mistä aiheesta pelin on** ja **antaa muitakin käytännöllisiä vinkkejä pelaamiseen**!

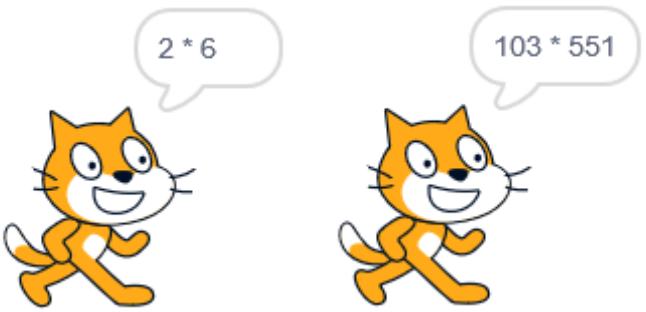


Anna pelaajalle tarpeeksi aikaa lukea tekstit! Lisää odota ( ) sekuntia-lohkoja hallitaksesi tekstien nopeutta.

Vaikeustason säätäminen

Jos palautteen mukaan pelisi oli joko liian **helppo** tai **vaikea**, voit vaikuttaa pelin vaikeustasoon muutamilla eri tavoilla:

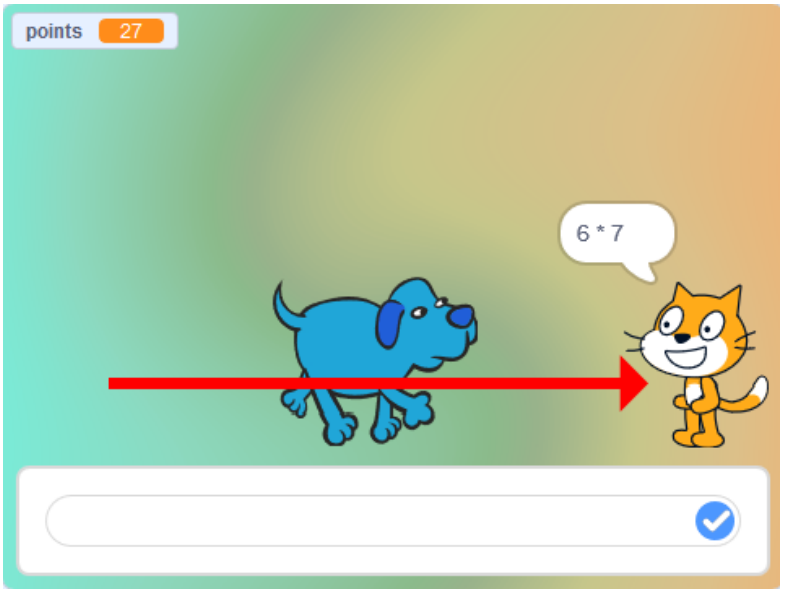
a) **Tee kysymyksistä joko helpompia tai vaikeampia**

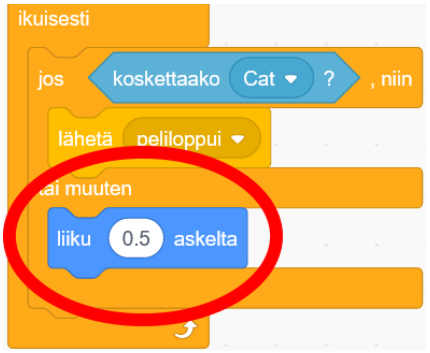


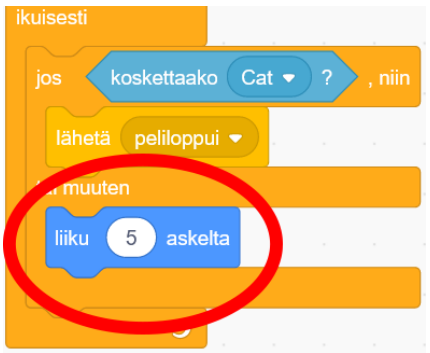
b) **Voit antaa pelaajalle vaihtoehtoja, joista valita tehdäksesi pelistä helpomman**

c) **Ota pelaajalta pisteitä, jos hänen vastauksensa on väärä**

d) **Voit laittaa vihollishahmon tai ajastimen liikkumaan nopeammin tai hitaammin**





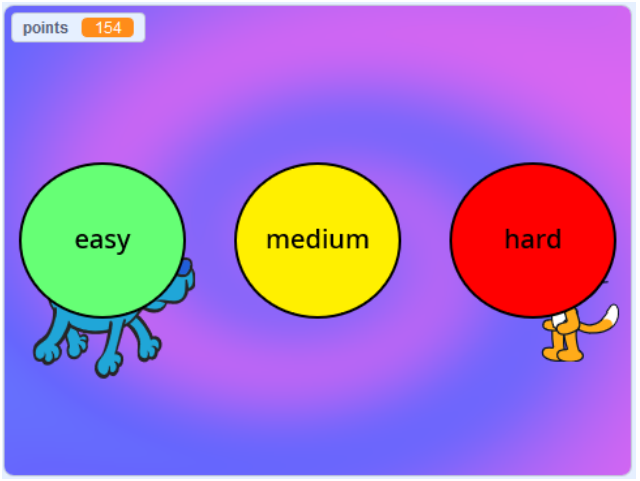


Vaikeustason valintapainikkeet ***(vaativa)***

Käyttämällä **lähetyksiä**, voit lisätä peliisi vaikeustason valintanäppäimet joiden avulla pelaaja saa itse valita pelin vaikeustason.

Mene osoitteeseen codeschool.fi/pt16

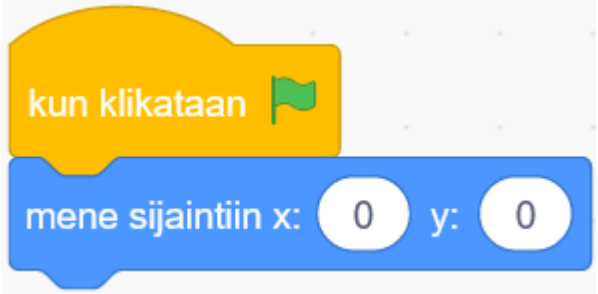
Klikkaa **Katso sisälle** -nappia.

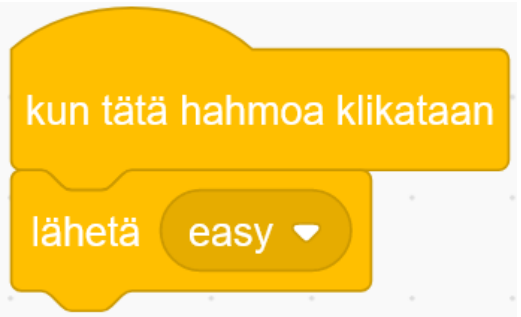


Tässä esimerkissä vaikeustason valinnalla **vaikutetaan Dog2 hahmon nopeuteen**. Mitä vaikeampi vaikeustaso valitaan, sitä nopeammin hahmo liikkuu kohti **Cat** hahmoa.

Vaikeustason valintanäppäimet ovat **omia hahmojaan**.

Jokaisella nappihahmolla on oma skriptinsä **sijainnille esiintymislavalla** ja jokainen niistä lähettää **eri lähetyksen** kun niitä klikataan.

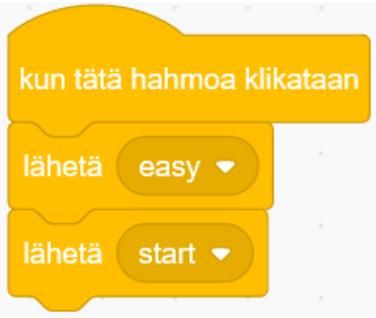




Kun **Dog2** hahmo vastaanottaa **lähetyksen vaikeustason valinnasta**, lähetystä vastaava skripti käynnistyy kun vastaanotan ( )-lohkon alapuolelta.

Vaikeustason vaikuttaa hahmon nopeuteen

Että **peli ei alkaisi heti kun** 🚩 **kuvaketta klikataan**, vaan odottaisi että pelaaja valitsee vaikeustason, täytyy myös pelin alkamiselle tehdä oma **lähetys**.



Kuvien nimien käyttö vastauksina

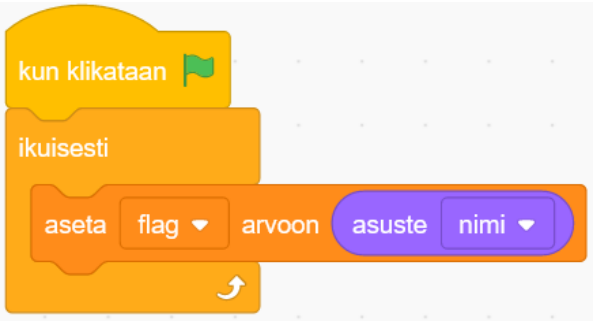
Esimerkkipelissä **lippupeli** (codeschool.fi/pt11) oikeat vastaukset on ohjelmoitu vastaamaan **lipun asusteen nimeä**. Voit vapaasti käyttää tätä tekniikkaa, jos pelissäsi käytetään kuvia kysymysten tukena.

Avaa lippupeli, klikkaa **katso sisälle** kuvaketta ja katso koodia.

Liput-hahmolla on useita eri asusteita jotka on kaikki nimetty kyseessä olevan lipun mukaisesti.

Aina kun pelaaja vastaa oikein, skripti **lähettää** viestin seuraavalippu. Toinen skripti **Liput**-hahmon ohjelmoinnissa vastaanottaa viestin, ja vaihtaa hahmolle seuraavan asusteen.

Lippu-muuttuja on asetettu vastaamaan asusteen nimeä. Käyttämällä ikuisesti-toistoa ohjelma tarkistaa ikuisesti, mitä asustetta hahmo kulloinkin käyttää. Tällä tavalla, jos pelaaja kirjoittaa vastaukseksi saman tekstin kuin millä asuste on nimetty, on vastaus silloin **oikein**.



Saman voi tehdä myös ilman toistoa. Keksitkö miten?

Taustat, erikoistehosteet & äänet

Lisäämällä peliisi **taustoja**, **erikoistehosteita** & **ääniä** saat siitä näyttävämmän. Voit katsoa mallia tästä projektista: codeschool.fi/pt15

Alta löydät **vinkkejä taustojen**, **erikoistehosteiden** ja **äänien** käyttöön.

Below you'll find **tips about backdrops, effects and sounds**.

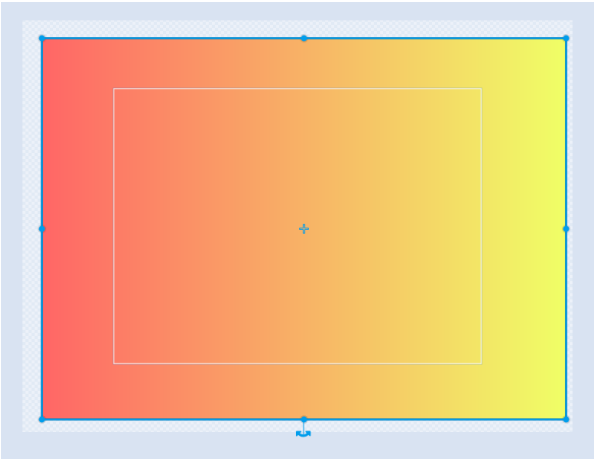
Taustat:

Taustojen luominen on hyvin samanlaista kuin asusteiden luominen. Klikkaa tyhjää taustaa esiintymislavan alapuolelta.

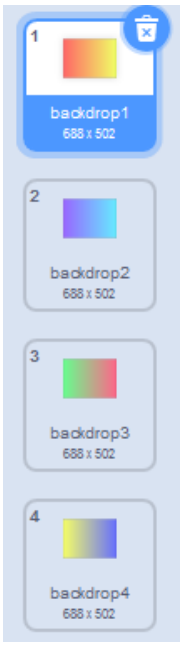
Valitse **Taustat**-välilehti.

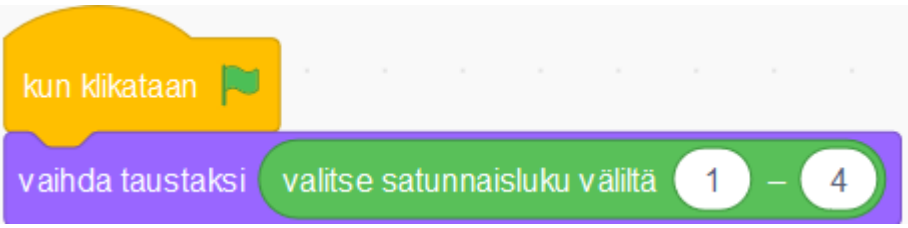
Käytä **suorakuolmiotyökalua** luodaksesi ison suorakulmion joka on **suurempi kuin esiintymislava**.

Käytä seuraavaksi **täyttötyökalua** valitaksesi taustallesi värin. Käytä eri **täyttövalintoja** jos haluat taustasi olevan jotain muuta kuin yksivärinen!



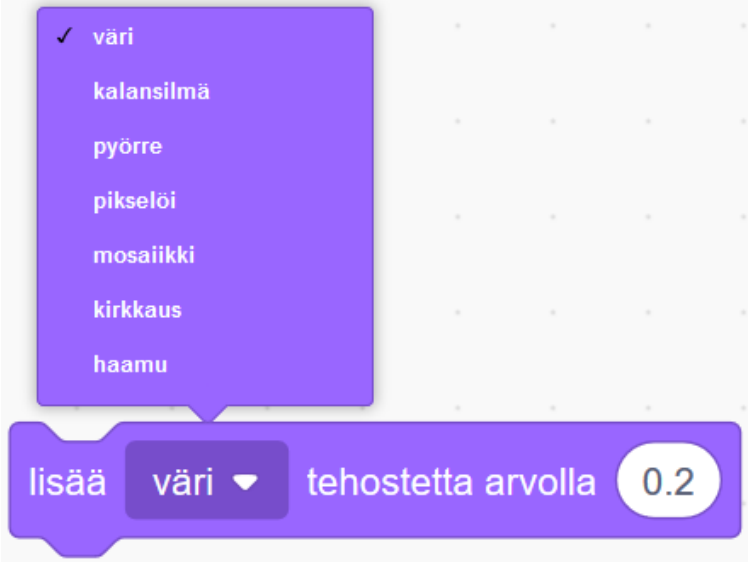
Voit lisätä peliisi useamman erilaisen taustan ja tehdä skriptin, joka valitsee pelin alussa satunnaisesti yhden taustoista pelin taustaksi.

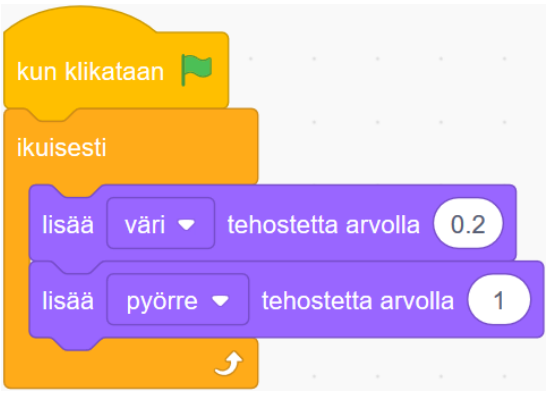




Erikoistehosteet:

Käyttämällä **tehosteita** voit lisätä peliisi jännittäviä yksityiskohtia. Lisäämällä lisää [ ] tehostetta arvolla ( )-lohkoja **taustan ohjelmaan** voit tehdä pelisi taustasta eläväisen!

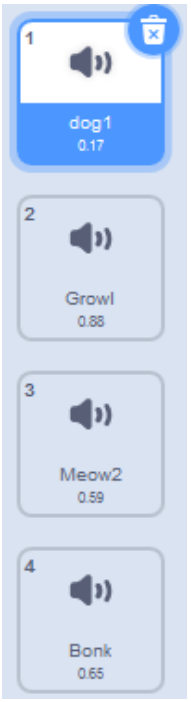




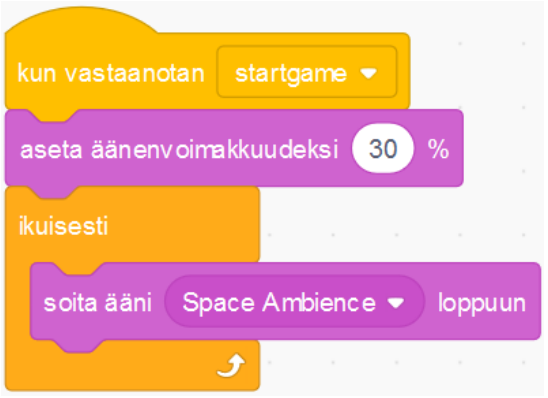
Äänet:

Käytä **ääniä** pelissäsi **korostaaksesi** animaatioita.
Kun aiemman esimerkin (endgame) kissan ja koiran tappeluanimaatio alkaa, ohjelma aloittaa samanaikaisesti myös **ääniskriptin**.

Skripti suorittaa soita ääni (Growl) loppuun-lohkon kerran, jonka jälkeen se aloittaa animaation. Animaatioskriptiin on lisätty myös soita ääni (valitse satunnaisluku väliltä (1) - (4))-lohko, joka valitsee satunnaisen äänen 1-4 **Dog2** hahmon **äänivalikoimasta**.
Äänivalikoimaan voit vaikuttaa **Äänet**-välilehden kautta.



Ääniä voi käyttää myös taustamusiikkina.

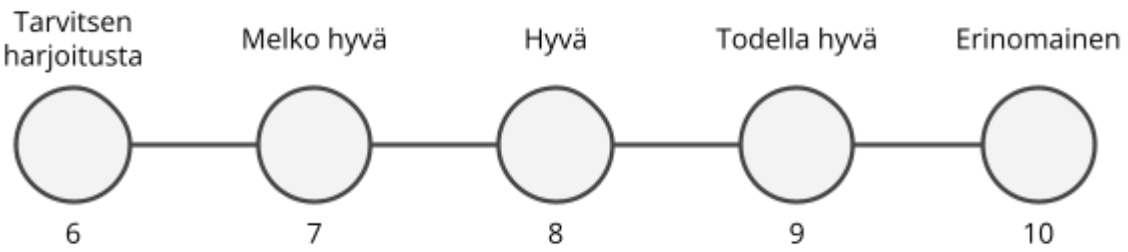


Itsearviointi - Osa 2

Vastaa seuraaviin kysymyksiin tehtyäsi kaikki osan 2 harjoitukset:

1. Arvioi oma osaamisesi arviointiperusteiden mukaan.

   Keskustele arvioinnista parin kanssa!



2. Mieti omaa opiskeluasi osaan 2 liittyen. Kirjoita itsellesi kolme kehua, jotka liittyvät toimintaasi ja onnistumisiisi.

   _____

   _____

   _____

3. a) Lue lupaus, jonka annoit itsellesi tämän osan alussa. Pystyitkö pitämään lupauksesi?

   b) Miksi, miksi et?

   _____

   _____

   _____

SUOMEN KOODIKOULU

ISBN: 978-952-7403-26-6

9 789527 403266

www.suomenkoodikoulu.fi