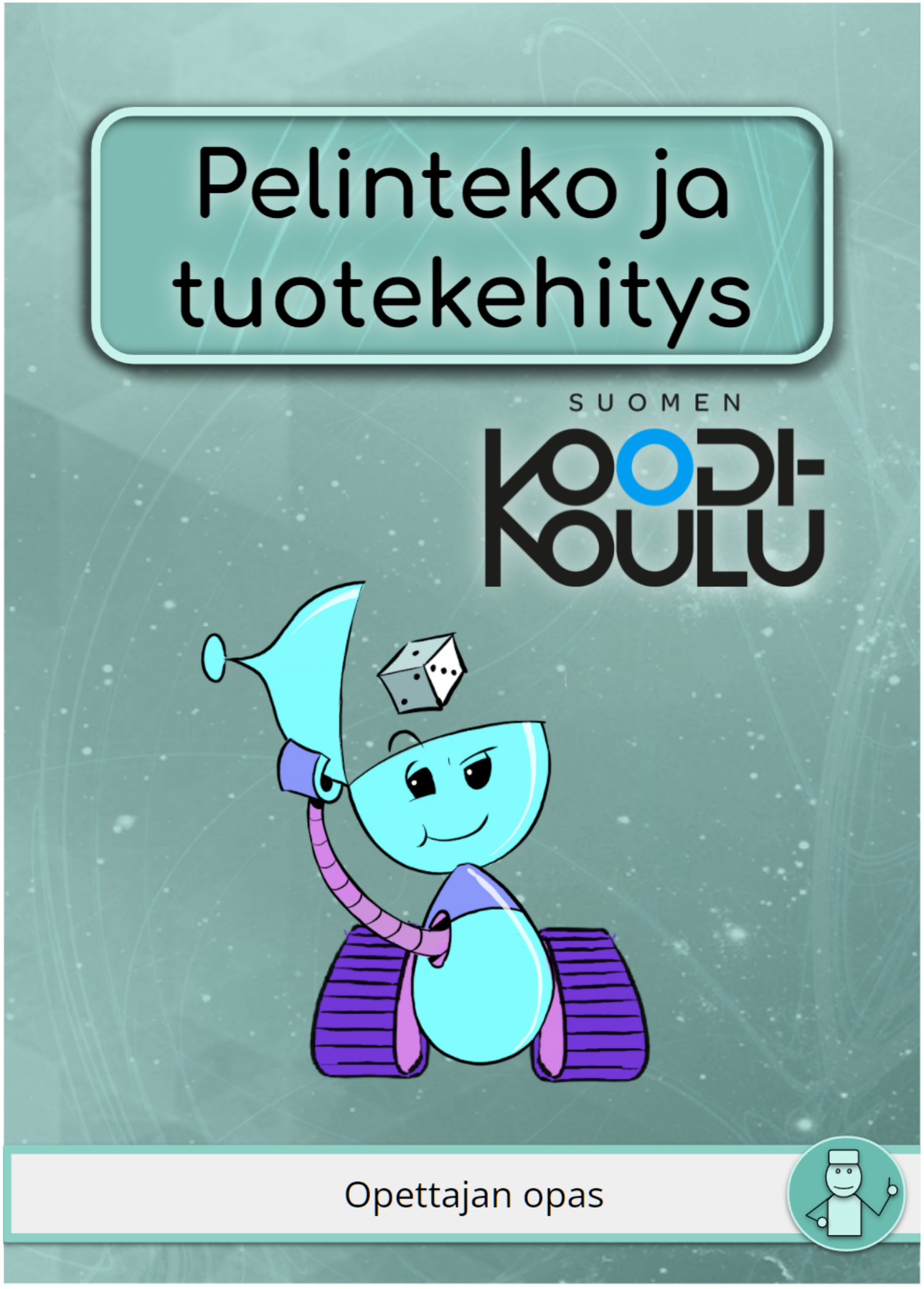



**Jussi Koivisto,**
**Jaakko Korpela**

# Pelinteko ja tuotekehitys

**Opettajan opas** - Versio 1.0

**Kustantaja: Suomen Koodikoulu Oy**
**www.codeschool.fi**



ISBN: 978-952-7403-30-3

## Opettajan oppaan rakenteesta

*Opettajan opas* sisältää kaikki *Oppilaan kirjan* sivut (vasen), sivulta 5 alkaen. Opettajalle suunnatut ohjeet, tiedot ja työkalut ovat sivun oikealla puolella. Opettajan opas sisältää esimerkkiratkaisut haasteisiin, esimerkkiprojekteja, työkaluja arviointiin ja sisällön laajentamiseen.

---

## Symbolien selitykset

Oppilaan kirjassa toistuu neljä toimintaa kuvaavaa symbolia: ***Ohjeet, Testaa ja tutki, Haaste!*** *ja* ***Keskustele***.

***Ohjeet***

→ ***Oppilas seuraa kirjan ohjeita tai opettaja ohjaa osion.***

***Testaa ja tutki!***

→ ***Oppilaat tutkivat ja kokeilevat eri työkaluja ilman tarkkoja ohjeita.***


***Haaste!***

→ ***Oppilaat ratkaisevat rajattuja tai avoimia ongelmanratkaisutehtäviä.***

***Keskustele***

→ ***Oppilaat keskustelevat aiheista parin, ryhmän tai koko luokan kesken.***

---


***Arviointi***

→ ***Tämä symboli esiintyy ainoastaan opettajan oppaassa. Sen yhteydessä esitellään työkaluja ja vinkkejä arviointiin.***

# *Pelinteko ja tuotekehitys* - Oppimiskokonaisuuden esittely

Tässä oppimiskokonaisuudessa oppilaat tutustuvat ohjelmoinnin alkeisiin ja projektilähtöiseen ohjelmoinnin oppimiseen tarkoituksenmukaisella tavalla. Oppimiskokonaisuus on tarkoin suunniteltu polku, joka mukailee pelillistä oppimista: Oppilaalle esitellään uusia työkaluja, niiden käyttöä harjoitellaan, sovelletaan ja yhdistetään aikaisemmin opittujen työkalujen kanssa. Harjoituksia ja projekteja suositellaan tekemään parityönä. Oppimiskokonaisuuden oppimistavoitteet on sovellettu *Uudet lukutaidot* -hankkeessa tarkennetuista, POPS:iin pohjautuvista ohjelmointitaitojen tavoitteista*****.

Ohjelmointiharjoitukset tehdään Scratch-ohjelmointiympäristössä. Scratch on ilmainen verkkoselaimessa toimiva ohjelmoinnin opettamiseen kehitetty ohjelma, jota oppilaat voivat halutessaan käyttää myös kotona.

***Pelinteko ja tuotekehitys*** rakentuu kahdesta osiosta. Ensimmäisessä osiossa kerrataan Scratchin käyttö, tutustutaan muuttujiin ja koordinaatiston hyödyntämiseen ja tehdään prototyyppi oppimispelistä. Osiossa 2 opitaan edellytykset pelin valmiiksi saattamiseen ja hiomiseen sekä viimeistellään peli versioksi 1.0.

***** *Uudet lukutaidot, Ohjelmointiosaaminen: https://uudetlukutaidot.fi/ohjelmointiosaaminen/*

---

## Oppitunnit

Oppimiskokonaisuus koostuu 24 oppitunnista. Alla on ehdotus tuntien käytöstä.

| **Lista oppitunneista** | | |
|---|---|---|
| **Oppitunti #** | **Sisältö** | **Oppilaan kirjan sivut** |
| **1** | Aloitetaan; Scratchin kertaus | 7-11 |
| **2** | Komennot, toistorakenteet ja ehtolauseet | 12-20 |
| **3** | Haaste-tehtävät; testaa tietosi | 21-22 |
| **4** | Muuttujat; Muuttujien määrittäminen ja muuttaminen | 23-28 |
| **5** | Haaste-tehtävät | 29 |
| **6** | Kysyminen ja vastaaminen; Haaste-tehtävät | 30-34 |
| **7** | Haaste-tehtävät (jatko) | 34-35 |
| **8** | Koordinaatit; Haaste-tehtävät | 36-40 |
| **9** | Projektin aloitus | 41-45 |
| **10** | Projektityö | 46-50 |
| **11** | Projektityö | 46-50 |
| **12** | Projektien esittely, purku | - |
| **13** | Itsearviointi; Välikoe | 51 + välikoe |
| **14** | Välikokeen purku; Osan 2 aloitus; Sisäkkäiset toistot & ehdot | 52-54 |
| **15** | Silmukoiden ja ehtolauseiden sisäkkäinen asettelu | 55-57 |
| **16** | Asusteet & animointi | 58-63 |
| **17** | Animointi (oma peliprojekti) | 64-65 |
| **18** | Testaaminen ja palautteen kerääminen | 66-67 |
| **19** | Projektityön jatkaminen | 68-76 |
| **20** | Projektityö | 68-76 |
| **21** | Projektityö | 68-76 |
| **22** | Projektityö | 68-76 |
| **23** | Itsearviointi; Loppukoe | 77 + loppukoe |
| **24** | Pelien esittely kohderyhmälle; Loppukokeen purku | - |

*Tämä ajankäyttösuunnitelma pohjautuu pääasiassa 6lk oppilaiden kanssa tehtyihin testauksiin. Mikäli oppilaat ovat vanhempia, voi eteneminen olla nopeampaa.*

# Sisällys

**Osa 1 - Keskitason Scratch-ohjelmointia** **6**
Aloitetaan - Valmistautuminen osaan 1 7
Luetaan – Scratchin kertaus 8
Scratchin yleiskatsaus 9
Ohjelmoidaan – Komennot, toistorakenteet ja ehtolauseet 12
Komennot ja tapahtumat 12
Toistorakenteet 13
Toimintolohkot 15
Ehtolauseet 16
Testaa tietosi 22
Luetaan – Muuttujat 23
Ohjelmoidaan – Muuttujien määrittäminen ja muuttaminen 24
Muuttujien vertaaminen 26
Klikkauspeli eli “klikkeri 29
Kysyminen ja vastaaminen 30
Luetaan – Koordinaatit 36
Ohjelmoidaan – Koordinoimaan opettelu 37
X:n ja y:n arvojen muuttaminen 38
**Projekti – Oppimispelin prototyyppi** **41**
Suunnittelu 42
Itsearviointi - Osa 1 51

**Osa 2 - Oppimispelin kehittäminen** **52**
Aloitetaan - Valmistautuminen osaan 2 53
Luetaan – Sisäkkäiset toistot & ehdot 54
Ohjelmoidaan – Silmukoiden ja ehtolauseiden sisäkkäin asettelu 55
Ohjelmoidaan – Asusteet & animointi 58
Hahmojen animointi 62
Testaaminen ja palautteen kerääminen 66
Palautelomake 67
**Projekti – Oppimispelin versio 1.0** **68**
Itsearviointi - Osa 2 77

## Tietoa opettajalle

Tämä oppimiskokonaisuus on suunniteltu *POPS 2014* ja *Uudet lukutaidot* -hankkeen osaamistavoitteiden pohjalta. Oppimiskokonaisuudessa korostuvat erityisesti laaja-alaisen osaamisen alueista L1, L2, L4 ja L5.

Oppimiskokonaisuus on kestoltaan noin 24 tuntia. Se toimii erityisen hyvin monialaisena oppimiskokonaisuutena tai valinnaisena kurssina. Kohderyhmä on ohjelmointia jo aikaisemmin harjoitelleet vuosiluokkien 5-9 oppilaat.

# 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!**

## **Osa 1**, *sivut 6-51*

### Yleiskatsaus

Osassa 1 kerrataan Scratchin käyttö, tutustutaan muuttujiin ja koordinaatiston hyödyntämiseen ja tehdään prototyyppi oppimispelistä. ***Luetaan***-osioissa esitellään uudet käsitteet ja työkalut. ***Ohjelmoidaan***-osioissa työkalujen käyttöä harjoitellaan käytännössä seuraamalla pääasiassa tutkien, testaten ja ratkaisten haasteita. ***Projekti***-osuus esittelee osan 1 pääprojektin. Osan 1 suorittaminen kokonaan on suositeltua ennen osaan 2 siirtymistä.

**Koko oppimiskokonaisuudessa suositaan parityöskentelyä**, mutta toteutuksen voi tehdä myös yksin tai 3 hengen ryhmissä.

### Ajankäyttö

Osa 1 koostuu noin 12-14 oppitunnista (45 min/oppitunti).

### Tarvikkeet

- Tietokone, näppäimistö, hiiri (väh. 1/oppilaspari)

### Tärkeää osassa 1

- Rohkaise oppilaita jakamaan tietoa ja osaamista toisilleen.
- Tarjoa mahdollisuus tehdä peli sopivalle kohderyhmälle (rinnakkaisluokka tai nuoremmat oppilaat).
- Anna tilaa ja aikaa luovuudelle sekä itseilmaisulle.

---

# 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**.

# Aloitetaan - Osa 1

## Ajankäyttö

10 min

## Työtapa

Työskennellään **2-3** hengen yksiköissä.

Keskustelua voidaan käydä myös parien välillä, **4 hengen ryhmissä**.

---

## Orientoituminen

Monet oppilaat käyttävät eri sovelluksia ja pelejä joka päivä. Tämän sivun harjoituksissa oppilaat miettivät näitä ohjelmia ja ohjelmistoja. Niistä voi saada inspiraatiota oman projektin tekemiseen.

Oppilaille voidaan näyttää esimerkkejä Scratchilla toteutetuista oppimispeleistä, jotta he paremmin ymmärtävät, mitä tässä oppimiskokonaisuudessa heiltä odotetaan.

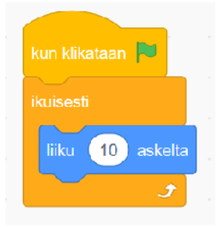
Esimerkkiprojekteja:

- Lippuvisa: **codeschool.fi/pt11**
- Matematiikkapeli 1: **codeschool.fi/pt16**
- Matematiikkapeli 2: **codeschool.fi/pt17**

# **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

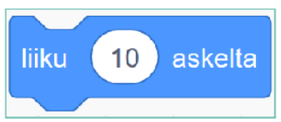
Voit luoda **omia ohjelmia** kuten pelejä ja sovelluksia käyttämällä Scratchia! Tässä lista **tärkeistä sanoista** joihin sinun täytyy tutustua ennen kuin siirryt Scratchiin:

### Hahmo on...

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

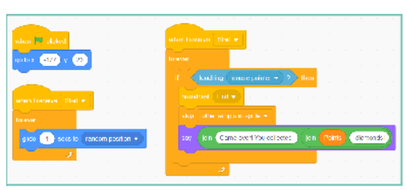
### Komento on...

**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.





# **Luetaan** - Scratchin kertaus

Tässä osiossa oppilaille oppilaat kertaavat Scratchin perusteet olettaen, että oppilailla on hieman kokemusta Scratchin käyttämisestä.

## Ajankäyttö

30 min

## Työtapa

Työskennellään **2-3** hengen yksiköissä, lukien ja keskustellen

## Avainsanat

*Ohjelmointi, ohjelma, koodi, toimintaohje, hahmo, komento, lohko, skripti*

---

## Hahmo, komento, koodi, rakenne...

Ohjelmointi Scratchissa on toimintaohjeiden antamista hahmoille. Hahmot ovat käytännössä kuvia tai kuvaobjekteja, joille annamme sielun koodin avulla. Varsinaisesti emme kuitenkaan käske hahmoja vaan tietokonetta, joka ohjaa hahmoja toimintaohjeiden mukaan.

Scratchissa koodia ei suoriteta ilman **tapahtumia**. Tapahtumiin rakennamme **skriptejä**, jotka koostuvat yhdestä tai useasta **komennosta**. Nämä komennot suoritetaan järjestyksessä ylhäältä alas. Jokaisella hahmolla on oma **koodi**, joka on kokoelma sen kaikista skripteistä.

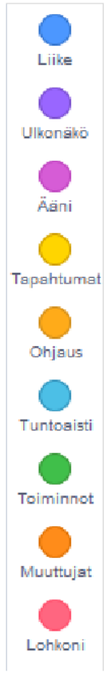
Scratchissa kaikista koodaukseen käytettävistä palikoista (komennot, tapahtumat, rakenteet) voidaan käyttää yleistä nimitystä **lohkot**.


*Esimerkki **skriptistä**, joka alkaa tapahtumasta **kun klikataan [vihreää lippua]**. Skriptissä on 7 **komentoa**, jotka suoritetaan järjestyksessä. Tämä skripti kertoo vitsin.*

## 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)**.**

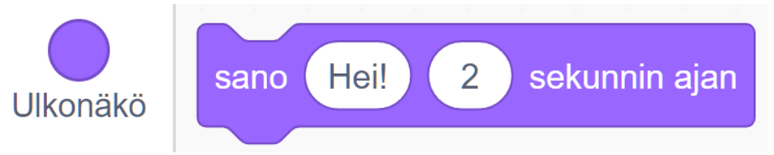
***Muuttujat***-lohkoja käytetään **tietojen tallettamiseen**.

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

*Scratchin päänäkymä esitellään seuraavalla sivulla →*

## Scratchin valikot

Scratchissa on värikoodatut valikot kaikille lohkoille. Lohkojen väristä voi päätellä, mistä valikosta se on haettu. Esimerkiksi violetti "*sano*" lohko on haettu *Ulkonäkö*-valikosta. Oppilaita kannattaa hoksauttaa värikoodauksesta, jotta ohjelmointi on sujuvaa esimerkiksi tilanteissa, joissa opettaja demonstroi valkotaululla tai näytöllä.



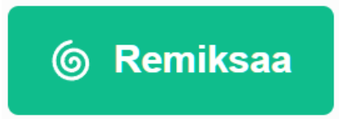

*Värikoodaus auttaa oppilaita löytämään oikean lohkon.*

Lisätietoa eri lohkoista: https://en.scratch-wiki.info/wiki/Block_Categories

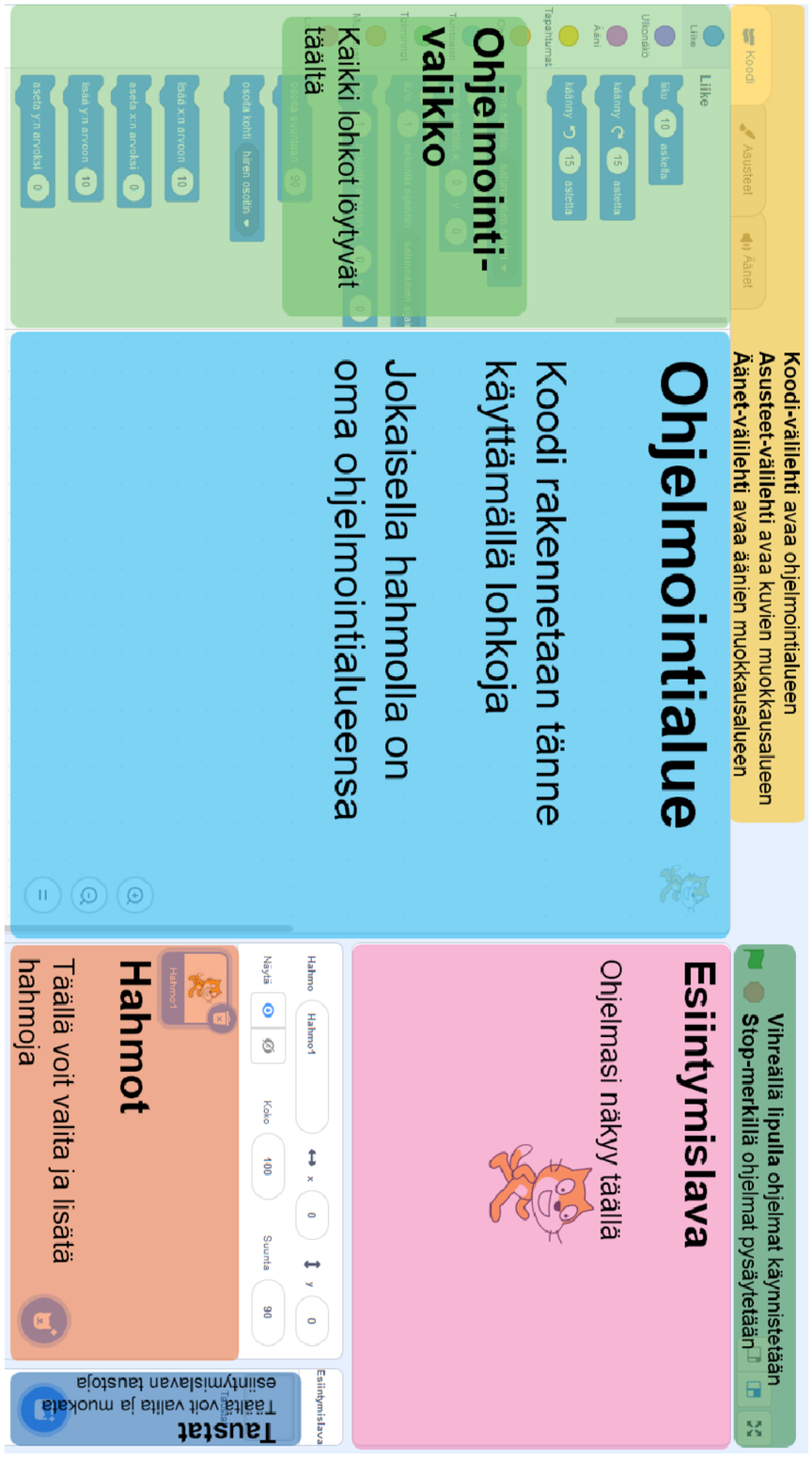
## Scratchista yleisesti

Scratch on tällä hetkellä **maailman suosituin graafinen ohjelmointiympäristö** ja sillä on noin 57 miljoonaa käyttäjää ympäri maailmaa. Se on käännetty yli 70 eri kielelle, mukaan lukien suomeksi. Jos oppilaiden Scratch ei ole valmiiksi suomen kielellä, kielen saa vaihdettua vasemman yläkulman maapallokuvakkeesta.

Scratchin perusidea on projektien jakaminen ja niiden "remiksaaminen" omiksi projekteiksi. Älä siis epäröi käyttää valmiita projekteja omiin tarkoituksiisi; se ei ole pelkästään sallittua vaan suositeltavaa! Esimerkiksi hakusanoilla "sustainability" tai "math quiz" Scratchista löytyy useita projekteja, joita voit pienin muutoksin käyttää opetuksessasi.




Scratchissa voit helposti remiksata valmiista projekteista omia projektejasi!



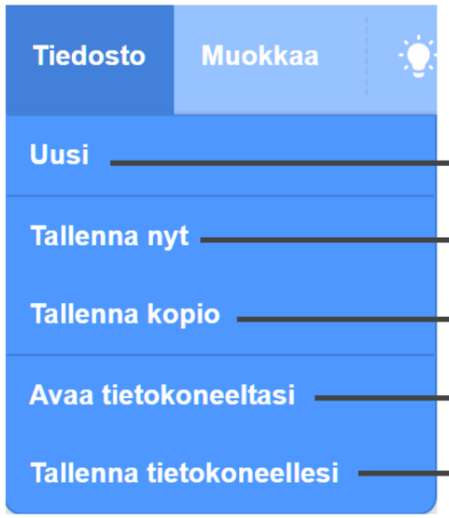

## Terminologia

Opettajan kannattaa käyttää ohjelmointiympäristön osista ja valikoista puhuessa johdonmukaisia nimiä. Esimerkit eri osioiden nimistä on esitelty oppilaan kirjan sivulla.

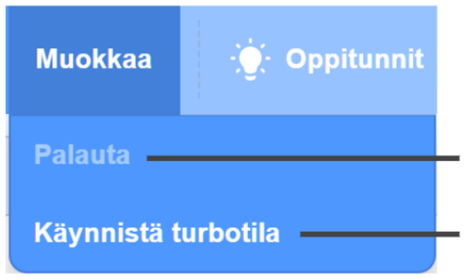
Kirjoissa käytetään sanaa **lohko** ja **komento** kuvaamaan yksittäisiä toimintaohjeita. Sanat ovat siis samaa tarkoittavia.

## *Tiedosto* ja *Muokkaa* yläpalkissa

Yläpalkissa on tärkeitä ohjelmointiympäristön käyttöön liittyviä valikoita. Näitä ei esitellä oppilaan kirjassa. Ne on syytä esitellä oppilaille sopivassa vaiheessa opettajan ohjaamana.




- **Uusi** — Uusi projekti/ohjelma
- **Tallenna nyt** — Tallenna nykyinen ohjelma projektikansioon.
- **Tallenna kopio** — Tallenna kopiona. Tämän jälkeen voit muokata kopiota ohjelmasta. Alkuperäinen säilyy projektikansiossa.
- **Avaa tietokoneeltasi** — Tuo projekti tietokoneelta Scratchiin.
- **Tallenna tietokoneellesi** — Tallenna projekti tietokoneelle. Tällä voidaan tallentaa projekti myös ilman Scratch-tunnuksia.




- **Palauta** — Jos vahingossa poistat skriptin, asusteen tai koko hahmon, voit palauttaa viimeksi poistetun täältä.
- **Käynnistä turbotila** — Maksimoi ohjelman suoritusnopeus.

## 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.*

### Keskustelu

*Keskustele*-työskentelyssä on tarkoitus keskustella pareittain tai pienissä ryhmissä, mutta jokainen kirjaa vastaukset lähtökohtaisesti kuitenkin omaan kirjaan tai vihkoon.

# **Ohjelmoidaan –** Komennot, toistorakenteet ja ehtolauseet

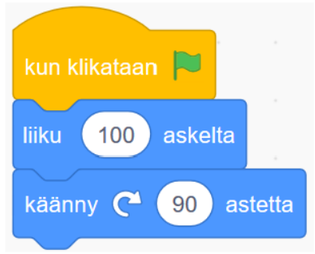
## Komennot ja tapahtumat

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.



# **Ohjelmoidaan** - Komennot, toistorakenteet ja ehtolauseet

## Ajankäyttö

2 oppituntia

## Työtapa

Työskennellään **1-3** hengen yksiköissä, mieluiten pareittain

## Avainsanat

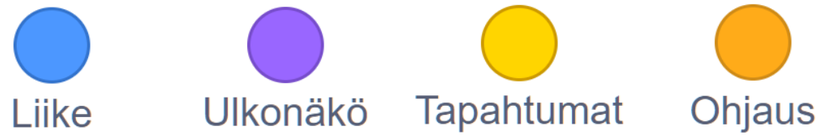
*Ohjelma, koodi, hahmo, komento, lohko, skripti, peräkkäisyys, tapahtuma, toistorakenne, ehto, ehtolause*

---

## Harjoitus (ehdotus): Lähtötaso

Jos opettaja haluaa nähdä oppilaiden lähtötason, voi hän esimerkiksi teettää heillä pienen projektin ennen kirjan harjoituksiin siirtymistä. Projektin voi rajata esimerkiksi seuraavasti:

- Tee pieni oma projekti käyttämällä valikoiden ***Liike, Ulkonäkö, Tapahtumat*** ja ***Ohjaus*** lohkoja (30 min).




## Viiveelliset komennot

Kun ***sano ( ) (2) sekunnin ajan*** -lohko suoritetaan, seuraavaan lohkoon siirrytään vasta 2 sekunnin jälkeen. Komennossa on siis **viive**. Myöhemmin esitellään myös ***odota***-lohko, jolla saadaan aikaa pelkkää viivettä.

## 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**.*

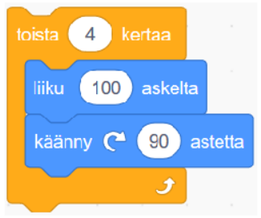
**Toista**-lohkoja käytetään ohjelmissa haluttujen komentojen **toistamiseen**. Ne tunnetaan yleisemmin nimellä **looppi** tai **silmukka**. Niiden avulla komentoja voidaan toistaa *ikuisesti* tai jonkin *määrityksen* tai *ehdon* mukaisesti.

### Ohjeet

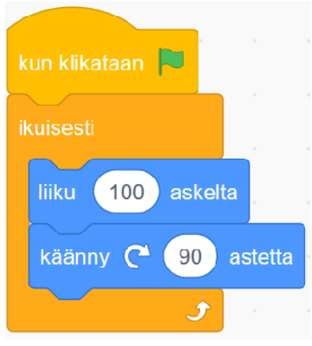
1. **Jatka** aiemmin aloitetun projektin parissa.

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?*



## Toistorakenteet

### Määrätty toisto...

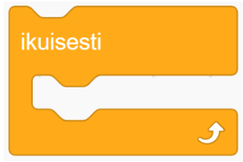
**Toistaa** komentolohkoja **niin monta kertaa** kuin toistossa määrätään.


### Ehdollinen toisto...

**Toistaa** komentolohkoja kunnes **ehto muuttuu todeksi**.


### Ikuinen toisto...

**Toistaa** komentolohkoja **loputtomasti**.



## 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](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**.

### Tuttuja harjoituksia

Osa oppimiskokonaisuuden harjoituksista saattavat olla jo entuudestaan tuttuja oppilaille. Joitain samoja harjoituksia käytetään *Ensiaskeleet ohjelmointiin* -oppimiskokonaisuudessa.

### Kokeiluja

**Lisätehtävä:** Kun harjoitukset ja keskustelu on tehty, voivat oppilaat muuttaa esimerkkiohjelmaa haluamallaan tavalla.

### Harjoitus (ehdotus): Code.org toistorakenteet

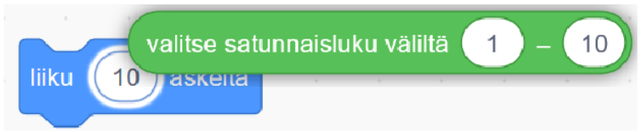
Code.org-sivustolla on yksinkertaisia pelillistettyjä harjoituksia ohjelmoinnin työkaluihin liittyen. Alla oleva harjoitus sopii hyvin toistorakenteen käytön harjoitteluun.

**Code.org piirtoharjoitus:** https://studio.code.org/s/course4/lessons/10/levels/1

*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).*





## Toimintolohkot

Tällä sivulla tutustutaan logiikka- ja operaattorilohkoihin. Ohjelmoinnissa näistä käytetään yleisesti vain nimeä operaattorit ja ne jaetaan kolmeen luokkaan: *Aritmeettiset operaattorit, loogiset operaattorit* ja *vertailuoperaattorit.* Scratchissa on myös *valitse satunnaisluku* ja erilaisia merkkijonoihin liittyviä operaattoreita, joita on hankala kategorisoida.

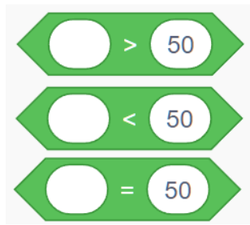

***Aritmeettiset operaattorit*** *antavat* numeerisen arvon.

Lohkoja on helppo testata tuomalla niitä ohjelmointialueelle, asettamalla sisään arvot ja klikkaamalla lohkoja.


---

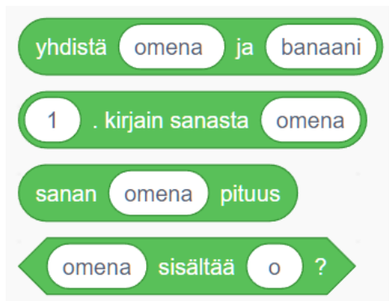

***Loogiset operaattorit*** antavat totuusarvoja.
*JA* lohko on tosi jos molemmat ehdot ovat tosia.
*TAI* lohko on tosi jos toinen ehdoista on tosi.
*EI* lohko muuttaa ehdon totuusarvon päinvastaiseksi..
Lisätietoa loogisista operaattoreista seuraavalla sivulla.

---



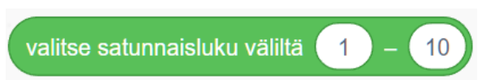

***Vertailuoperaattori*** ottaa arvoa ja vertailee niitä. Ehto on tosi jos vertailu on tosi (esimerkiksi 1 = 1)

---




***Merkkijonojen operaattorit*** käsittelevät merkkijonoja (tekstiä) ja antavat erilaisia arvoja (numero, merkkijono tai totuusarvo).

---




***Valitse satunnaisluku*** antaa satunnaisen luvun kahden arvon välillä.

## 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...

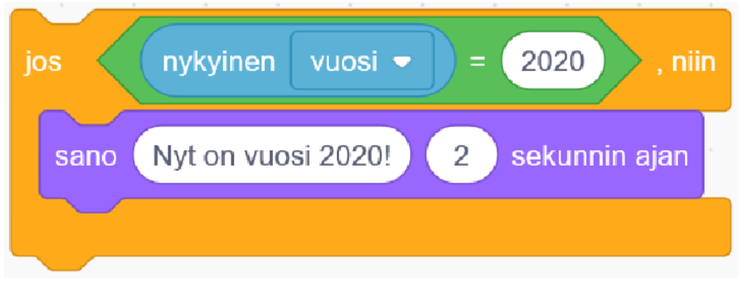
ilmaistaan luvuin ***1*** ja ***0***, jotka tarkoittavat ***tosi*** ja ***epätosi***. Niitä käytetään *ehtolauseissa* tarkistamaan *täyttyykö asetettu ehto (tosi)* vai *jääkö asetettu ehto täyttymättä (epätosi)*.

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.

## JA, TAI, EI

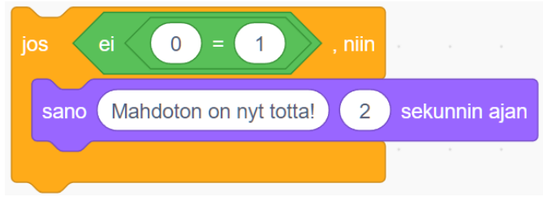
Tässä on esitelty loogiset operaattorit, jotka opettaja voi halutessaan esitellä

- **JA** yhdistää ehdot: Vain jos molemmat ovat tosia, koko ehto on tosi.


- **TAI** yhdistää ehdot: Jos edes toinen on tosi, koko ehto on tosi.


*Huomio: Päivä 7 on Scratchissa lauantai ja päivä 1 on sunnuntai.*

- **EI** kääntää ehdon. Jos sisään asetettu ehto ei ole tosi NOT, siitä tulee tosi, ja toisinpäin.



## Jos-tai muuten -ehto...


**Tarkistaa onko ehto tosi**.

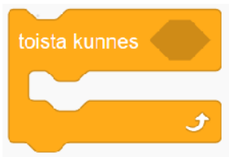
Jos ehto **on tosi**, **ylempänä oleva komento suoritetaan**. Jos ehto on **epätosi**, **alempana oleva komento suoritetaan**.

**Esimerkki:**


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

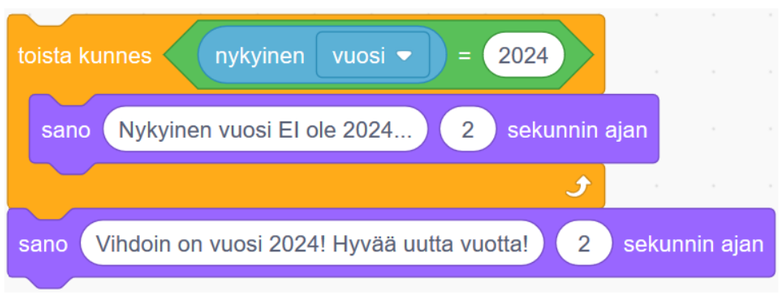
Jos **vuosi ei ole 2020**, hahmo sanoo “Nyt ei selvästikään ole vuosi 2020.”.

## 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!”*.

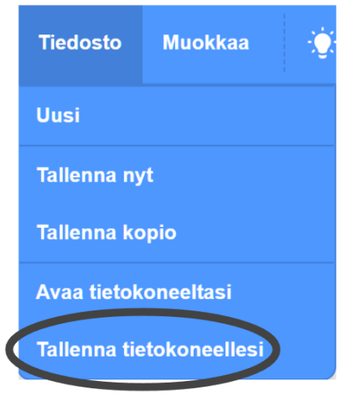
## Harjoitus (ehdotus): Code.org ehtolauseet

**Code.org sokkeloharjoitukset:** https://studio.code.org/s/course3/lessons/8/levels/1

## Tallentaminen

Jos oppilaat kirjautuvat Scratch-tunnuksilla, heidän projektit tallentuvat automaattisesti. Oikeassa yläkulmassa on myös “Tallenna nyt” painike tilanteisiin, josta viimeisimmät muutokset tallentuvat, mikäli niitä ei ole automaattisesti tallennettu.


Jos Scratch-tunnukset eivät ole käytössä, ohjelman voi helposti tallentaa suoraan tietokoneelle valitsemalla **Tiedosto->Tallenna tietokoneellesi**.




## Version hallinta

Isoja projekteja tehdessä on hyvä tallentaa eri versioita samasta ohjelmasta niiden kehittämisen matkan varrelta. Tämä auttaa esimerkiksi tilanteissa, joissa oppilas on vahingossa poistanut merkittäviä osia ohjelmaa.

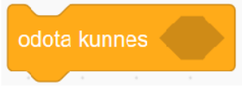
Eri versiot voidaan nimetä esimerkiksi näin:

- “Minun peli 1”, “Minun peli 1”... tai
- “Minun peli 0.1”, “Minun peli 0.2”...

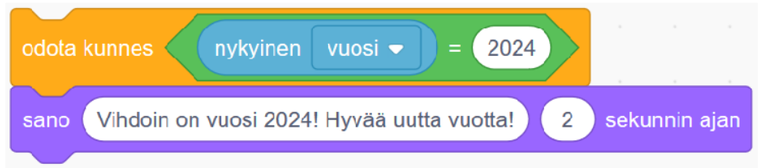
Kun käytetään Scratch-tunnuksia, eri versioita voidaan tallentaa valitsemalla **Tiedosto->Tallenna kopio**. Tietokoneelle tallentaminen toimii tietenkin myös tunnuksilla.

## Odota kunnes -ehto...

**Odottaa kunnes asetettu ehto täyttyy.**




**Esimerkki:**




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

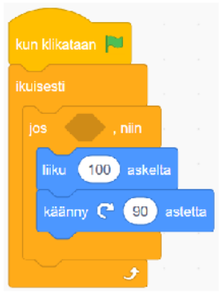
Sen jälkeen seuraava komento suoritetaan, ja hahmo sanoo *"Vihdoinkin on vuosi 2024! Hyvää uutta vuotta"*.

---

## *Ohjeet*

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

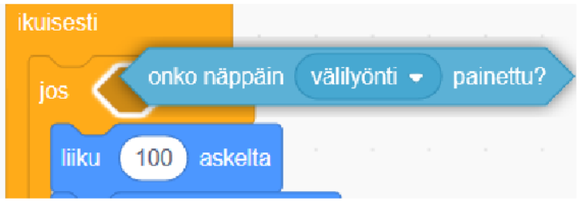
2. **Tuo** jos ( ), niin-lohko ohjelmointialueelle ja aseta se ikuisesti-lohkon sisään. Tämä kutsutaan *sisäkkäin asetteluksi*. Siirrä liiku ( ) askelta ja käänny ( ) astetta-lohkot sen sisään.

   ***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*.

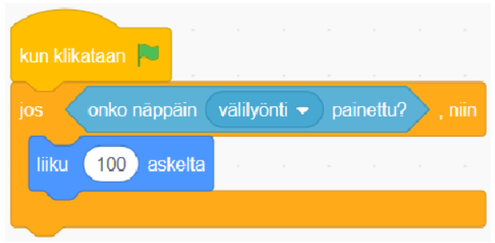
4. Mene Tuntoaisti-valikkoon ja **tuo** onko näppäin ( ) painettu?-lohko ohjelmointialueelle. Aseta se jos ( ), niin-lohkoon.



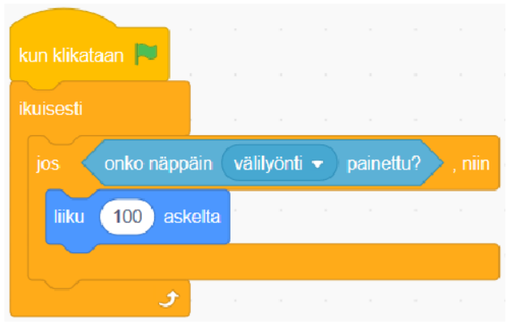

*Onko näppäin ( ) painettu? on **toimintalohko** ja siinä on **terävät reunat**. Se sopii tyhjään aukkoon jos ( ), niin-lohkossa.*

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*

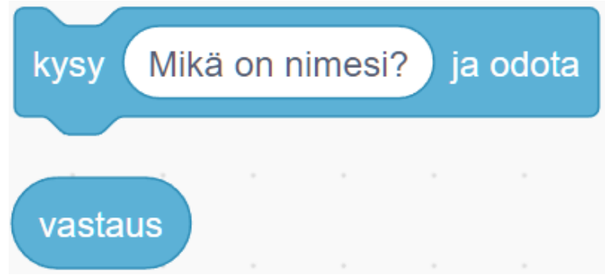



*Ehtolause asetettuna ikuisesti-lohkon sisälle*

## Tuntoaisti-valikko

Tuntoaisti-valikosta löytyy erilaisia ehtoja, operaattoreita ja muuttujia. Näillä lohkoilla ohjelma saa erilaista tietoa hahmoista ja käyttäjän tekemisistä: Mihin väriin hahmo tai hiiren osoitin koskee? Mikä on hiiren osoittimen ja hahmon välinen etäisyys? Mikä vuosi nyt on?

Valikosta löytyy myös "*kysy*"-komento, jolla voidaan kysyä pelaajalta kysymys. Kun kysymykseen on vastattu, vastaus varastoidaan "*vastaus*"-muuttujaan.




*Tuntoaisti-valikon lohkot ovat hyvin tärkeitä interaktiivisen sovelluksen, kuten pelin, tekemisessä.*

## 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**.

   _____

   _____

   _____

   _____

## Ehtolauseiden testaaminen

Tässä harjoituksessa oppilaat testaavat ehtolauseita ja tekevät testien perusteella päätelmiä.

Tehtävä 3 on tarkoitus käydä keskustellen pareittain tai pienissä ryhmissä Se voidaan myös ottaa keskustellen koko ryhmän kesken.

**Esimerkkivastaus tehtävään 4:**

*Kun E-näppäintä painetaan, hahmo alkaa jatkuvasti testaamaan koskettaako hiiren osoitin sitä. Jos ehto on **tosi**, hahmo reagoi. Jos ehto on **epätosi**, hahmon tila palautetaan normaaliksi.*

# 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.

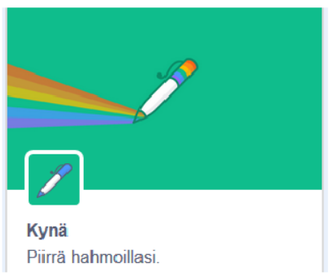
## Vinkkejä haasteisiin

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

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

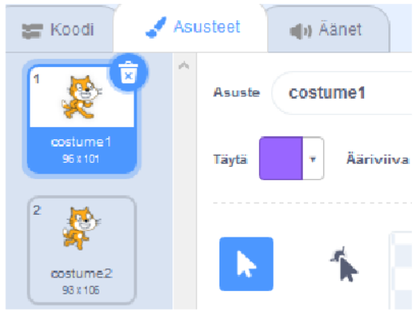
  Löydät valikon alareunasta **Lisää laajennus** kuvakkeen. Klikkaa sitä ja valitse **Kynä**.

  Nyt **kynä-lohkot** löytyvät valikosta.




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

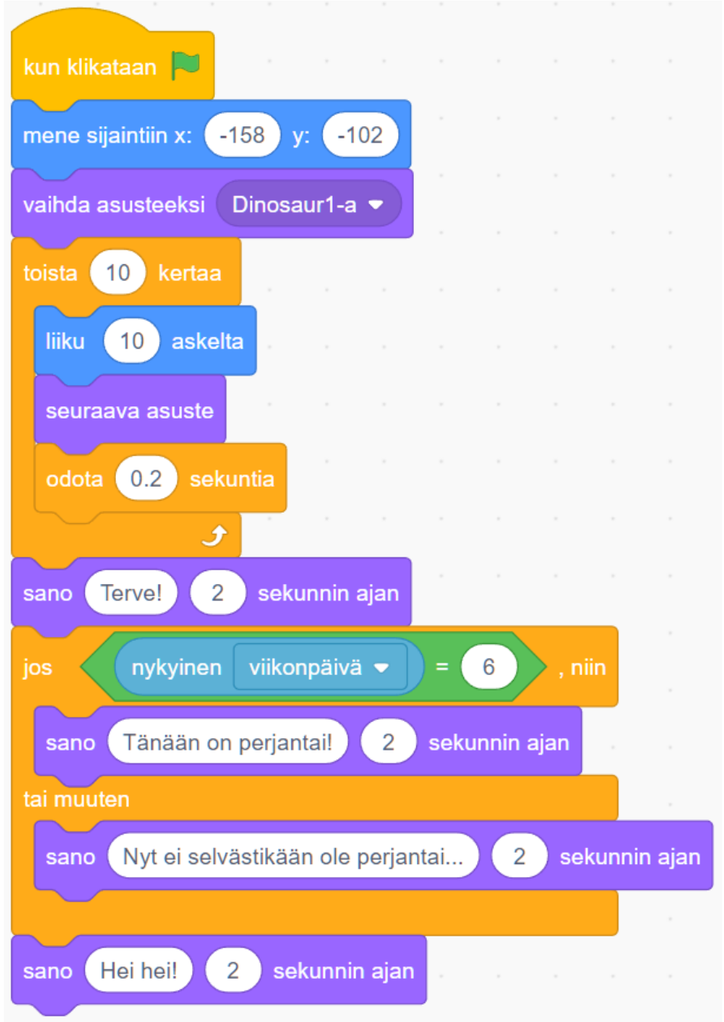
  Luomalla **erilaisia asusteita** samalle hahmolle, voit **animoida** hahmoja. Lisäämällä seuraava asuste- tai vaihda asusteeksi ( )-lohkoja koodiin, voit laittaa hahmon vaihtamaan asusteesta toiseen ja saat hahmon näyttämään siltä, kuin se liikkuisi.






## Esimerkkiratkaisut sivun 21 **haasteisiin 1-2**

**Haaste 1** (esimerkki): **https://codeschool.fi/pt18**




**Haaste 2** (esimerkki):
**https://codeschool.fi/pt19**

# 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!

## Testaa tietosi

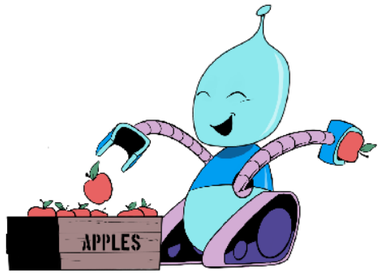
Tällä sivulla oppilaat ohjataan pohtimaan juuri opittua. Opettaja voi näiden vastausten perusteella päätellä, tuleeko ehtolauseisiin tai toistorakenteisiin liittyen tarkentaa jotain. Lisäksi kun oppilaat piirtävät itsensä ohjelmoijana, saattaa heidän ennakkoluulonsa ja tunteensa ohjelmointia kohtaan tulla esille.

**Esimerkkivastaus tehtäviin 1-2:**

1. ***Jos**-lohko kokeilee, toteutuuko siihen asetettu ehto. Jos toteutuu, rakenteen sisällä oleva komento suoritetaan.*
2. ***Ikuisesti** suorittaa rakenteen sisässä olevaa komentoa ikuisesti. **Toista** suorittaa komentoa vain määrätyn verran, esimerkiksi 10 kertaa.*

# **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**.

   _____

   _____

   _____



# **Luetaan** - Muuttujat

## Ajankäyttö

10-20 min

## Työtapa

Työskennellään **2-3** hengen yksiköissä, lukien ja keskustellen

## Avainsanat

*Muuttuja, tieto, luokitus, nimi, arvo*

---

## Harjoitus (ehdotus): Code.org muuttujat

**Code.org muuttujaharjoitukset:** https://studio.code.org/s/course4/lessons/6/levels/1

---

**Esimerkkivastaus tehtävään 2c:**

*Muuttujalla on nimi (luokka) ja arvo. Voimme käyttää muuttujia arvojen tallentamiseen ja uudelleenkäyttämiseen.*

# **Ohjelmoidaan –** Muuttujien määrittäminen ja muuttaminen

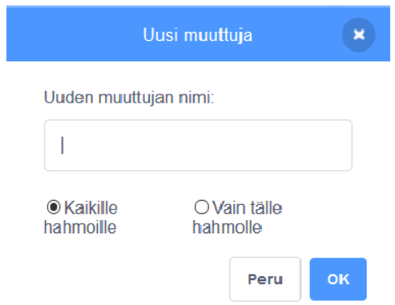
Tässä harjoituksessa **luomme muuttujan** joka muistaa **kuinka monta kertaa hahmoa klikataan**.

## *Ohjeet*

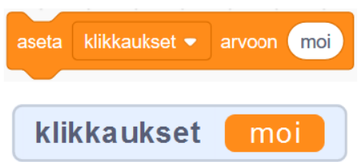
1. Luo **uusi Scratch projekti**. Anna sille nimi *“Muuttujaharjoitus”*.


2. Mene Muuttujat-valikkoon ja tee uusi muuttuja klikkaamalla **Tee muuttuja** nappia.
3. Anna muuttujalle nimi “klikkaukset”.

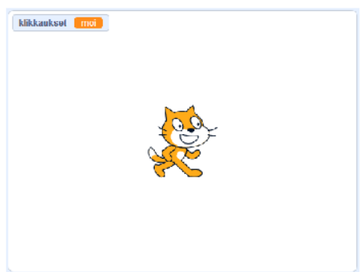
***Vinkki:*** *Kun nimeät muuttujia,* ***anna niille nimi joka kuvaa niiden käyttötarkoitusta****. Tämä auttaa sinua muistamaan, mihin käyttöön muuttuja on tarkoitettu. Jos tekisit esimerkiksi pistelaskuri muuttujan, voit nimetä sen nimellä “pisteet”.* ***Tässä harjoituksessa muuttuja on nimeltään “klikkaukset”,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



# **Ohjelmoidaan 1/2** - Muuttujien määrittäminen ja muuttaminen

## Ajankäyttö

2 oppituntia

## Työtapa

Työskennellään **1-3** hengen yksiköissä, mieluiten pareittain

## Avainsanat

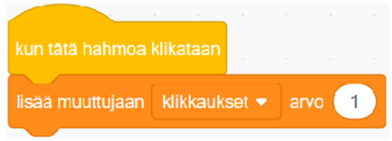
*Muuttuja, tieto, nimi, arvo, aseta, muuta, vertaa, remiksaus*

---

## Muuttujan nimeäminen

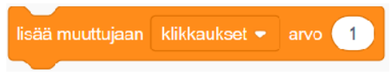
Hyviin ohjelmointikäytäntöihin (mainittu POPS 2014) kannattaa totutella varhaisessa vaiheessa. Scratchissa ohjelmat on kategorisoitu eri hahmoille omiksi skripteiksi, mikä helpottaa koodin luettavuutta, mutta muuttujien huonolla nimeämisellä voidaan saada sotkua aikaan. Jotta muutkin voivat lukea oppilaan koodia ja auttaa häntä haastavissa tilanteissa, tulee muuttujille ja oikeastaan kaikelle itse nimettävälle antaa kuvaavat nimet.

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.



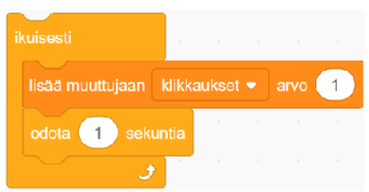

## *Testaa ja tutki!*

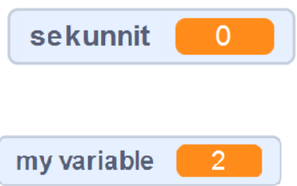
On aika testailla! **Jatka edellisellä sivulla aloittamasi projektin parissa**. Tässä osiossa **tutkimme muuttujien toimintaa**. Tästä on hyötyä myöhempää *Haaste!* -osiota varten.

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:

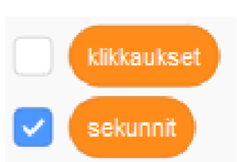
Muuttuja **muuttaa arvoaan yhdellä** aina sekunnin välein, toimien kuten ajastin!



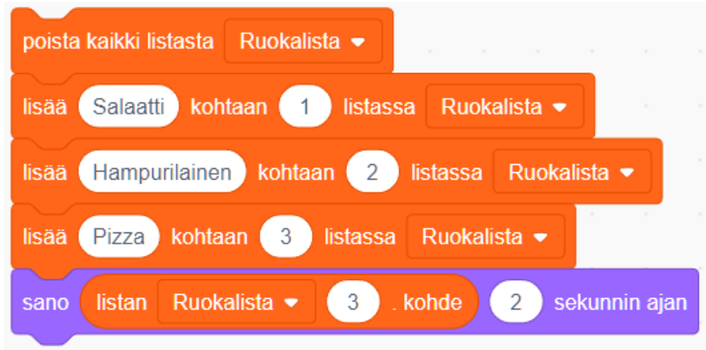

Muuttujan **arvo määrittyy** joksikin luvuksi lukujen 1 ja 10 välillä joka kerta, kun skripti aktivoituu.


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




### Listat

Scratchissa voi tehdä myös ***listoja***. Ne ovat muuttujia, joille voi antaa useita yhtäaikaisia arvoja, joilla on oma järjestyspaikka listalla. Listoja ei mainita oppilaan kirjassa, mutta opettaja voi ne halutessaan esitellä oppilaille.




*Esimerkki “Ruokalista” -listasta, jossa kolme eri arvoa on tallennettu järjestyksessä omalle paikalleen.*

### Jaetaan tutkimuksen tulokset

Oppilaita kannattaa rohkaista esittelemään toisilleen hauskat ja mielenkiintoiset löydöksensä muuttujien käyttöön liittyen (*Tutki ja testaa*).

## 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!*

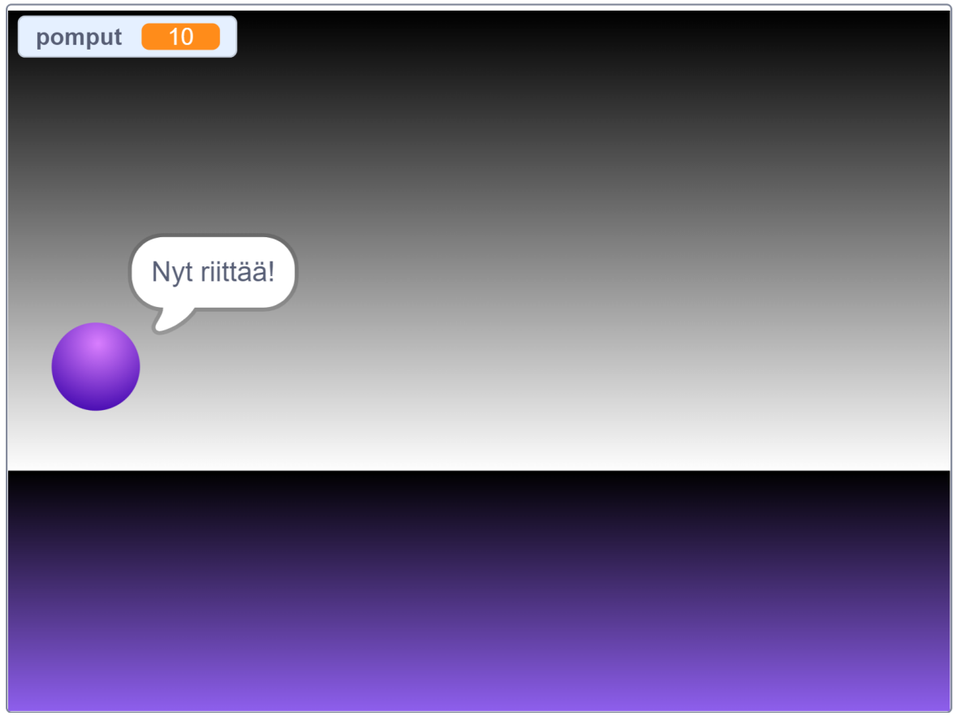

*Voit käyttää myös odota kunnes < >-lohkoa jos ( ), niin-lohkon sijaan. Se toimii hieman eri tavalla. Osaatko selittää eron?*

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

## Vaihtoehtoista tutkimusta

Tässä vaiheessa oppilaille annetaan avoimia tutkimustehtäviä. Oppilaille voidaan antaa myös valmis ohjelma, johon he tekevät muutoksia ja sitä kautta tutkivat muuttujien käyttöä ja vertailua.

Esimerkkiohjelma remiksattavaksi: **https://codeschool.fi/pt20**

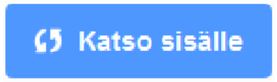


## Testaa ja tutki!

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**?

   _____

   _____

   _____

Esimerkkivastaukset **tehtävään 2** sivulla 27

***2a.*** *Muuttujan arvo asetetaan arvoon 0 ja kissat siirtyvät aloituspaikoille.*
***2b.*** *Muuttujan arvo kasvaa yhdellä.*
***2c.*** *Muuttujan arvo vähenee nopeasti välilyönnin ollessa pohjassa.*
***2d.*** *Molemmat kissat vaihtavat asustetta. Nyt niillä on aurinkolasit.*
***2e.*** *Vasen kissa palaa normaaliksi ja oikea kissa jatkaa aurinkolasit päässä.*

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**.




## Esimerkkivastaukset **tehtävään 3** sivulla 28

***3a.*** *Oikeanpuolimmainen kissa odottaa kunnes muuttujan arvo on 10. Vasemmanpuolimmainen suorittaa jatkuvasti ehtolauseen mukaista testaamista ja kun muuttujan arvo ei ole 10, se palaa normaaliksi.*

***3b.*** *Oikeanpuolimmaisen kissan skripti loppuu kun se saa aurinkolasit.*

**Tehtävässä 4**, oppilaat tekevät muutoksia koodiin. Tämä on avoin ongelmanratkaisutehtävä, jossa kokeillaan eri vertailulohkoja eri osana eri ehtolauseita.

This is an open-ended task to test the differences of comparison blocks in different conditional statements.

# 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!

## Yksinkertainen klikkeri

Klikkeripelit kuulostavat ajatuksena tylsältä ja typerältä, mutta luovuuden ja huumorin voimin niistä saadaan hienoja harjoitusprojekteja. Tarkoitus on tehdä peli, jossa pelaaja klikkailee hahmoa ja peli muuttuu klikkien määrän perusteella.

Tässä pienessä projektissa oppilaat muuttavat valmista ohjelmaa. Projekti antaa paljon mahdollisuuksia muuttujan käytön ja vertailun harjoittelulle.

Projektissa on englanninkielisiä muuttujia ja tekstejä, jotka oppilaat voivat itse suomentaa.

**Esimerkkiohjelma, jossa haasteet 1-3 on ratkaistu: https://codeschool.fi/pt21**

Haaste 3 voi olla haastava ymmärtää. Sen voi demonstroida näyttämällä yllä olevaa esimerkkiohjelmaa. Siinä haasteeseen on vastattu seuraavalla tavalla:

- Yli 5000 klikkausta: Hahmo vaihtaa asustetta
- Yli 10000 klikkausta: Hahmo alkaa liikkumaan
- Yli 20000 klikkausta: Hahmosta tulee pienempi

Tässä toinen esimerkki klikkeripelistä: **https://codeschool.fi/pt22**

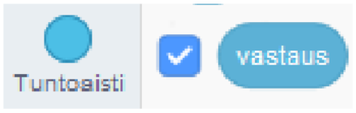
## Kysyminen ja vastaaminen

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.**



# **Ohjelmoidaan 2/2** - Kysyminen ja vastaaminen

## Ajankäyttö

2 oppituntia

## Työtapa

Työskennellään **1-3** hengen yksiköissä, mieluiten pareittain

## Avainsanat

*Kysy, vastaus, syöte, muutuja, arvo, nimi, aseta, muuta, vertaa, vuorovaikutteinen*

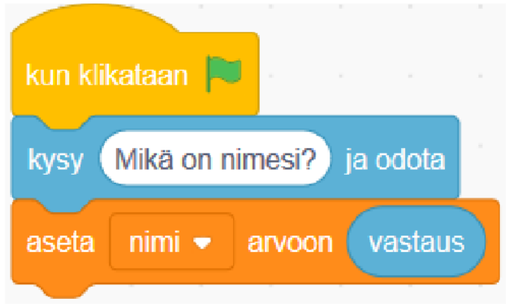
---

## *Kysy* ja *vastaus* -lohkot

*Kysy () ja odota* lohko pyytää käyttäjää syöttämään tekstiä ohjelmaan. Kun se odottaa vastausta, skriptin suorittaminen on jumissa tässä komennossa, kunnes vastaus on annetut. Vastaus tallennetaan *vastaus*-lohkoon. Jos käyttäjälle esitetään toinen kysymys, siihen annettu vastaus tallennetaan edellisen päälle ja vanha vastaus poistetaan.

**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****.*

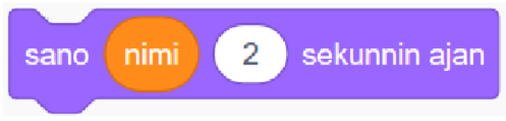
   *Sen jälkeen* ***aseta [nimi] arvoon (vastaus)-lohko asettaa nimi-muuttujan vastaamaan vastaus-muuttujan arvoa****.*

   *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.

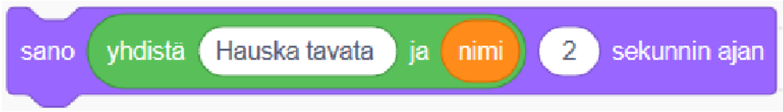



## Muuttujan arvon välittäminen toiseen muuttujaan

Voimme aina välittää yhden muuttujan arvon toiseen, kuten sivun 31 ensimmäisessä kuvassa: *Vastaus*-muuttujan arvo tallennetaan *nimi*-muuttujaan. Näin voidaan kysyä uusi kysymys ilman, että edellisen vastaus menetetään.

Tämä on tärkeä muistaa, kun teemme tietovisan tai matematiikkapelin.

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.



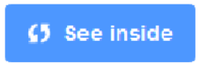

***Vinkki:*** *Muista jättää yksi tyhjä välilyönti ensimmäisen lauseen jälkeen!*

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.

*Testaa ja tutki* sivulla 32

Tässä harjoituksessa kerrataan muuttan arvon siirtäminen. Mikäli aikaa on vähän,voi tämän harjoituksen jättää väliin.

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

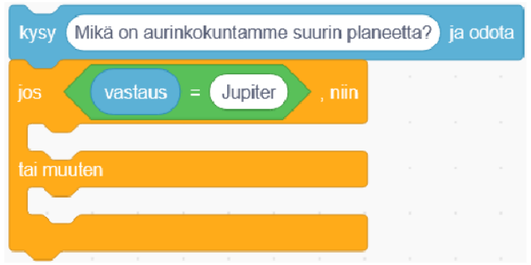
**Voit ohjelmoida vastauksen tarkistuksen** käyttämällä **( ) = ( ) logiikkatoimintoa**.

## 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!**

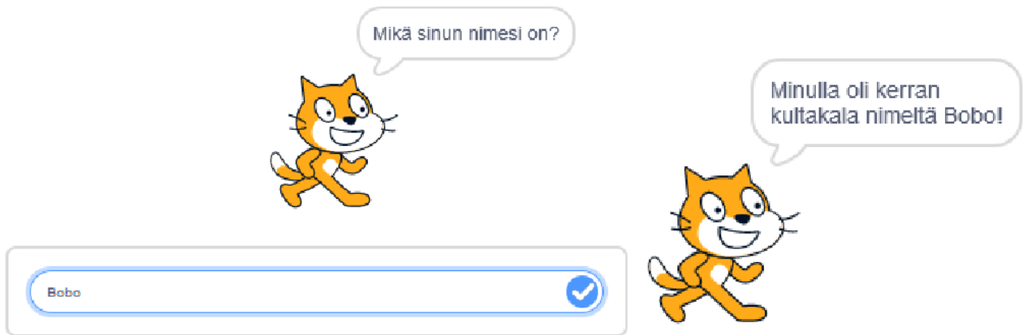



# Haaste!

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.*

## Tylsä tietovisa

Tässä linkki hyvin yksinkertaiseen tietovisaan, josta voi lähteä rakentamaan omaa versiota, mikäli skriptien rakentaminen itse tuntuu haastavalta.

Linkki: **https://codeschool.fi/pt9**

## Haasteet 1-3 sivulla 34

Oppilaat voivat kokeilla kaikkia haasteita lyhyesti **tai** keskittyä yhteen haasteista syvemmin. Tarkoitus on harjoitella ja saada ideoita varsinaista projektityötä varten. Vinkkejä haasteisiin löytyy seuraavalta sivulta.

## Jaetaan ideoita

Tässä vaiheessa ei ole tarkoitus tehdä hienoja, täysin toimivia pelejä, vaan soveltaa aikaisemmin opittua ongelmanratkaisun kautta. Oppilaita kannattaa silti rohkaista jakamaan ideoita ja näyttämään tekemäänsä muille. Tästä voi olla hyötyä myöhemmin projektityössä.

Näitä “miniprojekteja” voidaan käyttää myöhemmin pohjana varsinaisille oppimispeliprojekteille.

## Vinkkejä haasteisiin

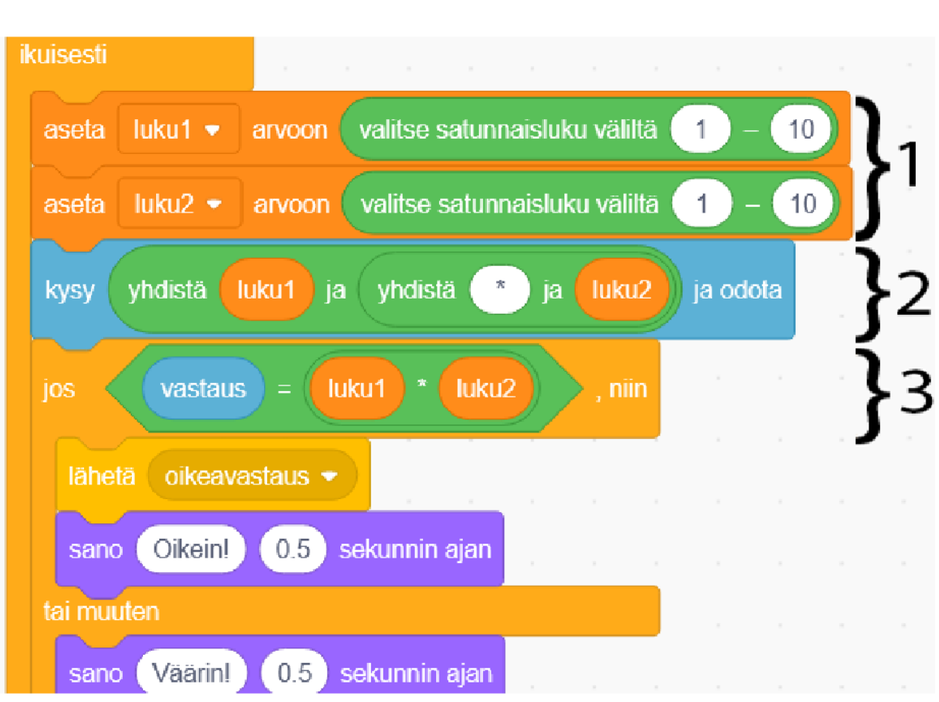
- Haasteessa 2 sinun tulee käyttää **muuttujia**, että **ohjelma muistaa pelaajan nimen**.
- Haasteessa 3 voit tehdä **matikkapelistäsi älykkään**! ***(vaativa)***

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.




### Muistutus

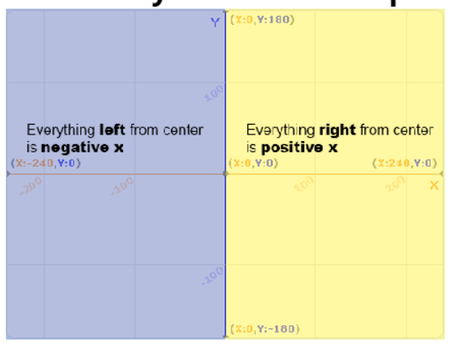
Sivulla 41 aloitetaan varsinainen projektityö. Mikäli on tarkoitus tehdä projekti kohderyhmälle, esimerkiksi nuoremmille oppilaille, tulee teema tai aihe oppimispeleille sopia kohderyhmän opettajan kanssa ennen projektityön aloittamista.

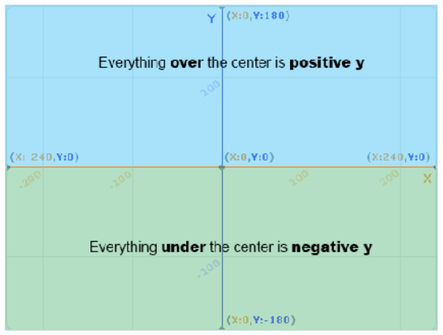
# **Luetaan –** Koordinaatit

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

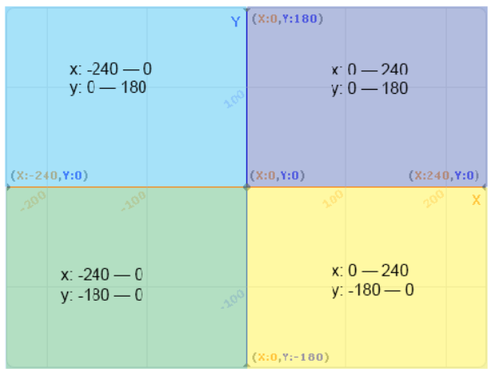
**X-akseli kulkee vasemmalta oikealle**, ja se määrittää **esiintymislavan leveyden**, joka on **480 pikseliä**. Se on jaettu keskeltä negatiiviseen ja positiiviseen puoliskoon, ja sen vasen ääripiste on ***-240*** ja oikea ääripiste on ***240***.

**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*

## **Luetaan** - Koordinaatit

### Ajankäyttö

10 min

### Työtapa

Työskennellään **2-3** hengen yksiköissä, lukien ja keskustellen

### Avainsanat

*Koordinaatit, Koordinaatisto, x, y, akseli, arvo, paikka, 2d, pikseli*

# **Ohjelmoidaan –** Koordinoimaan opettelu

## Ohjeet

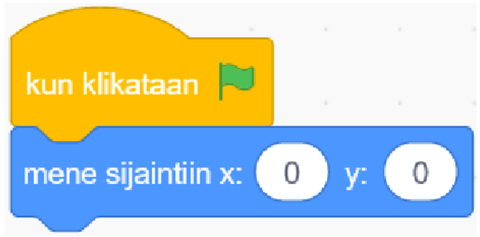
1. **Aloita uusi Scratch projekti**. Lisää projektiin hahmo. Voit nähdä hahmon koordinaatit esiintymislavan alapuolelta.

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


## Testaa ja tutki!

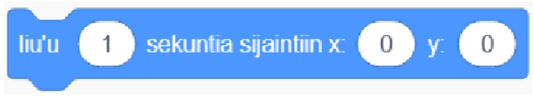
1. Tuo mene sijaintiin x: ( ) y: ( )-lohko ohjelmointialueelle. Sen avulla voit määrittää hahmon **x:n** ja **y:n** sijainnin.

*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.*



# **Ohjelmoidaan** - Koordinoimaan opettelu

## Ajankäyttö

1-2 oppituntia

## Työtapa

Työskennellään **1-3** hengen yksiköissä, mieluiten pareittain

## Avainsanat

*Koordinaatit, Koordinaatisto, x, y, akseli, arvo, paikka, liike, 2d, pikseli*

---

## Koordinaattipeli

Koordinaattien kanssa päästään alkuun valmiin Scratch-ohjelman avulla.

Linkki: **https://codeschool.fi/pt7**

Esimerkkiratkaisu:

**1:** -212, 155

**2:** 200, 155

**3:** 200, -100

**4:** 0, -100

**5:** 0, 85

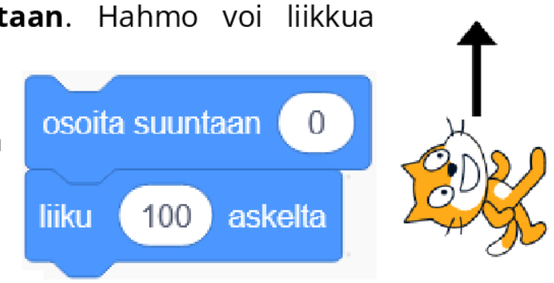
**6:** 100, 85

**7:** 100, 20

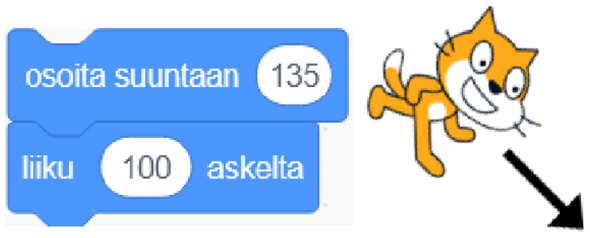
## X:n ja y:n arvojen muuttaminen

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.

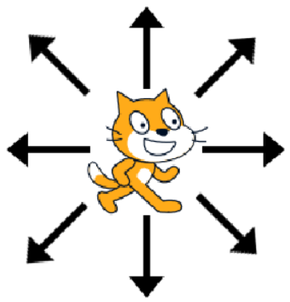
Liikuttaaksesi hahmoa **ylös**, hahmon täytyy osoittaa **ylöspäin**.




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?

## Liikkuminen ja paikan asettaminen

Liikkuminen ja paikan asettaminen ovat hieman eri asioita. Komennolla ***liiku*** hahmo saadaan liikkumaan senhetkisestä paikasta tietyn verran hahmon osoittamaan suuntaan. Komennolla ***mene*** hahmo siirretään suoraan tiettyyn paikkaan sen koordinaatteja muuttamalla.

Sivun 38 esimerkissä esitellään tämä ero. Esimerkki antaa myös uusia ideoita hahmojen liikkumisen toteuttamiseen esimerkiksi omassa projektissa.

Linkki esimerkkiin: **https://codeschool.fi/pt8**

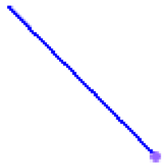
## Testaa ja tutki!

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

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

   a) **sivuttaissuunnassa**

   b) **vinottain**

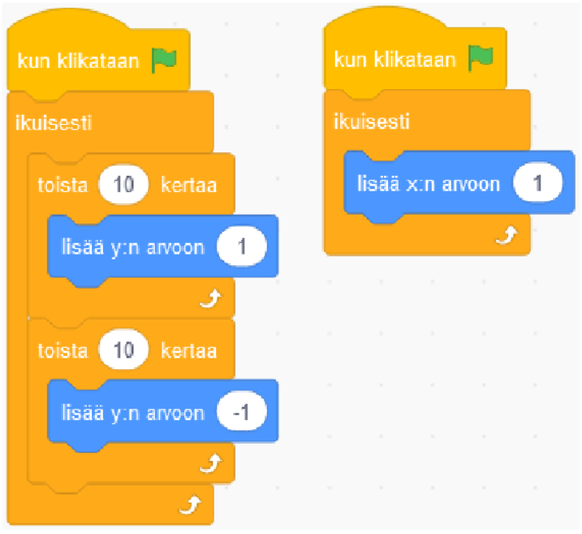
   

   c) **aaltoliikkeessä**

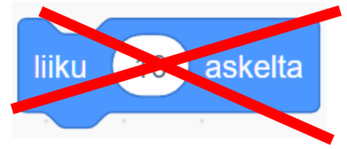
### Vinkkejä hahmon liikuttamiseen x:n ja y:n avulla

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

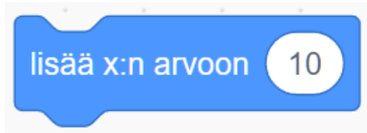



### Esimerkkiratkaisuja

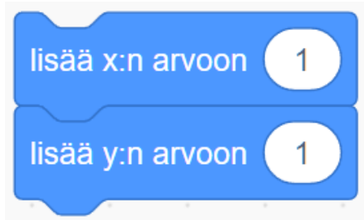
Jotta oppilaat perehtyvät tässä tutkimustehtävässä haluttuun asiaan, oppilaita tulee ohjata käyttämään koordinaatteihin perustuvia komentoja *liiku*-komennon sijaa.



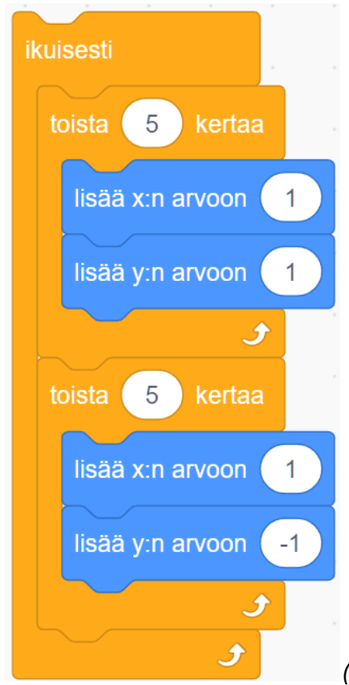

a)




b)




c)

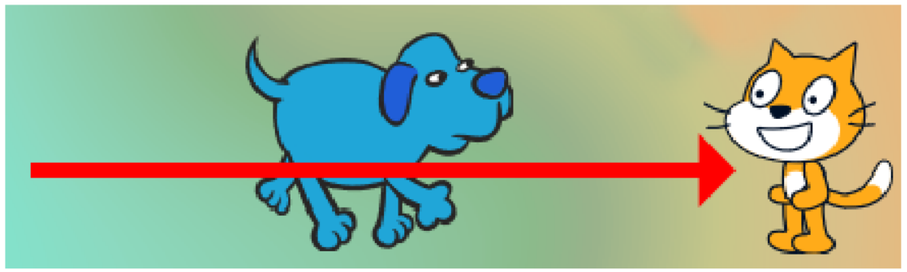



*(Toinen esimerkki oppilaan kirjassa)*

# 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:

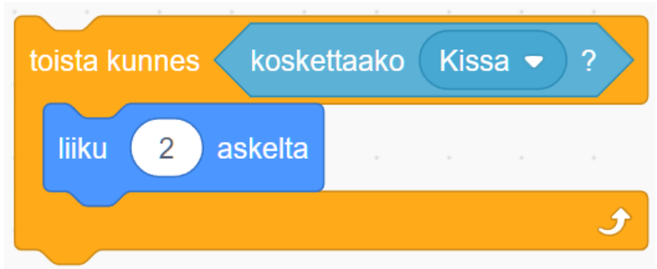
Jos käyttäjä kirjoittaa “**ylös**”, hahmo liikkuu ylös.
Jos käyttäjä kirjoittaa “**alas**”, hahmo liikkuu alas.
Jos käyttäjä kirjoittaa “**vasen**”, hahmo liikkuu vasemmalle.
Jos käyttäjä kirjoittaa “**oikea**”, hahmo liikkuu oikealle.

## 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**.


## Esimerkkiratkaisuja sivun 40 **haasteisiin 1-2**

**Haaste 1** (esimerkki):




**Haaste 2** (kaksi esimerkkiä):


# **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 →**



# **Projekti** - Oppimispelin prototyyppi

## Ajankäyttö

3-5 oppituntia

## Työtapa

Työskennellään **projektiryhmissä (2-4 oppilasta)**

## Avainsanat

*Projekti, suunnittelu, prototyyppi, muotoiluajattelu, sanallistaminen, kohderyhmä, palaute, teema*

---

## Projektiryhmät

Peliprojektit tehdään pareittain tai ryhmissä. Projektia työstetään samoissa ryhmissä sekä tässä että myöhemmässä projektivaiheessa.

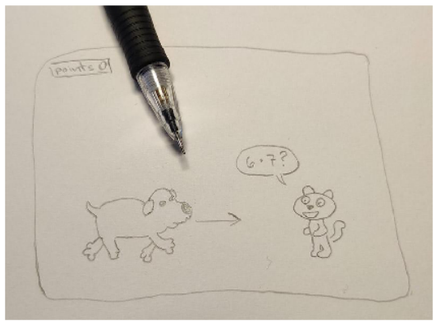
## Projektin vaatimukset

Projektin vaatimukset on esitelty sivulla 41. Nämä vaatimukset on suunniteltu oppimistavoitteiden näkökulmasta. Näiden vaatimusten lisäksi ryhmillä on omat tavoitteensa. Tarkoitus ei siis ole lopettaa projektia kun vaatimukset on saavutettu vaan siitä alkaa oppilaiden oma luovuus ja projektin omaksi tekeminen.

## 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**.



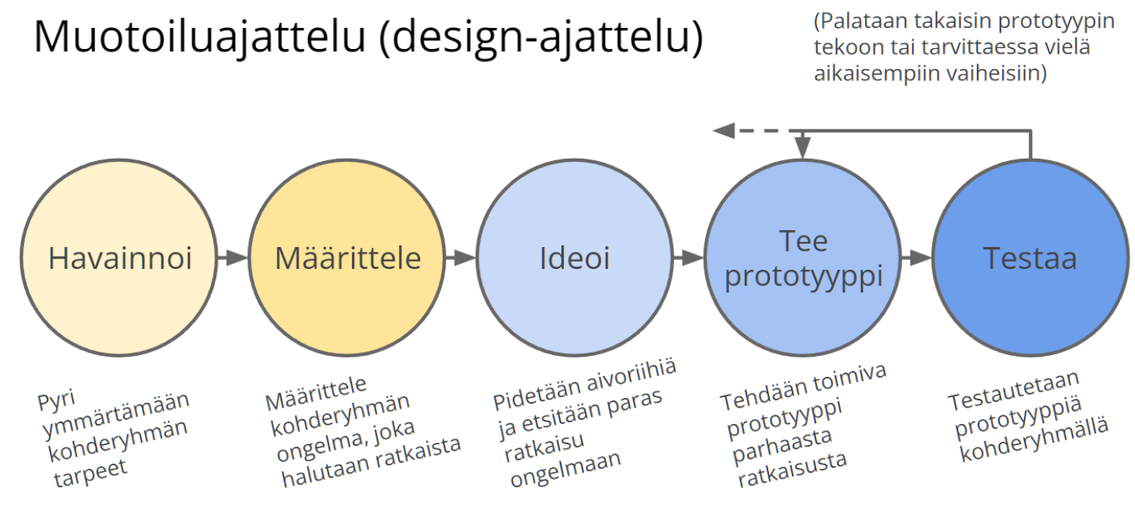

**Luonnostele ja piirrä** paperille, miltä peli voisi näyttää. On paljon kätevämpää tehdä luonnokset ensin paperille, kuin alkaa suoraa piirtämään Scratchissa. Pohdi valmiiksi esimerkiksi mihin eri napit tulevat, miltä hahmot näyttävät, miten hahmot liikkuvat, mihin asiat on sijoiteltu ja niin edelleen.

**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.*

## Muotoiluajattelu / design-ajattelu

Tuotekehittämistä on hyvä lähestyä muotoiluajattelun kautta. Se voidaan esittää viisivaiheisena prosessina: *Havainnoi, Määrittele, Ideoi, Tee prototyyppi* ja *Testaa*. Opettaja voi esitellä prosessin ja hyödyntää sitä oppilaiden kanssa.




**Havainnoi (Empathise):** Pyritään ymmärtämään kohderyhmän tarpeet ja toiveet. Opettaja voi järjestää oppilaille mahdollisuuden haastatella kohderyhmään kuuluvia.

**Määrittele (Define):** Kun ymmärretään tarpeet ja toiveet, voidaan määritellä, mitä pelin tulisi opettaa ja miten. Tässä projektissa tämä vaihe ja ideointivaihe menevät helposti päällekkäin, mutta se ei ole ongelma.

**Ideoi (Ideate):** Tässä vaiheessa on tarkoitus rohkeasti kokeilla ja ideoida kaikkea mieleen tulevaa, joka sopii projektin teemaan.

**Tee prototyyppi (Prototype):** Nyt tehdään pelistä prototyyppi eli ensimmäinen toimiva versio, jonka ei tarvitse olla täydellinen. Pelin prototyypissä esiintyy pelin tärkeimmät ominaisuudet, mutta niitä ei ole hiottu. Tästä vaiheesta voi myös palata takaisin ideointiin, mikäli sille esiintyy tarvetta.

**Testaa (Test):** Prototyypin testaamisella saadaan tärkeää tietoa prosessin edeltävien vaiheiden onnistumisesta. Testaamisen perusteella palataan takaisin suunnittelupöydälle ja uuden prototyypin tekemiseen. Oppilaille kannattaa antaa mahdollisuus testauttaa peliä kohderyhmällä. Tähän palataan tässä oppimiskokonaisuudessa myöhemmin.

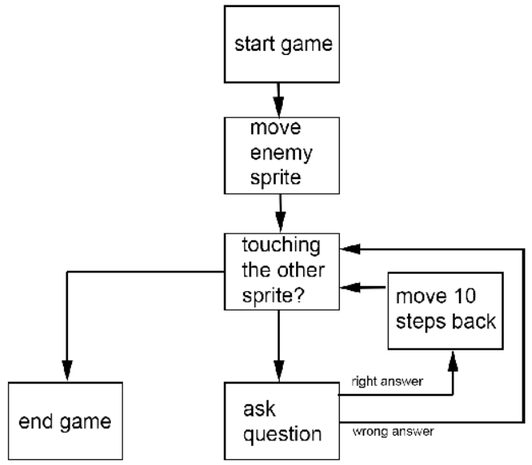
Ollaksesi vielä tarkempi, voisit **kuvailla vihollishahmon toimintoja** seuraavalla tavalla:

> *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 →**

## Muotoiluajattelu pelisuunnittelussa

Peli on erinomainen projekti harjoitella muotoiluajattelua, koska pelin tekeminen on myös pyrkimys arvioida, miten pelaaja tulee pelaamaan peliä. Vain prototyyppien testaamisella voidaan saada oikeasti parempi ymmärrys pelaajan tekemistä valinnoista. Muotoiluprosessin etu on se, että virheellisen tai heikosti toimivan pelin tekeminen ei tuomitse koko projektia epäonnistuneeksi vaan antaa mahdollisuuden parannella peliä.

# Pelinteko ja tuotekehitys

Tee suunnitelma

Ryhmän jäsenet:

Kuvaile peliä:

_____

_____

_____

_____

| | Kyllä | Ei | Ehkä |
|---|---|---|---|
| Tarvitsemme apua ohjelmoinnissa | ○ | ○ | ○ |

**Luonnostele ja piirrä** miltä peli näyttää:

**Valmistaudu esittelemään suunnitelmasi opettajalle ja muille oppilaille!**

## *Arviointi*

### Vertaisarviointi: Kuinka antaa palautetta?

Projektien alustavat suunnitelmat esitellään toisille ja suunnitelmista keskustellaan. Oppilaita on hyvä herättää ajattelemaan: Millaista on hyödyllinen palaute? Kun oppilaat antavat palautetta toistensa suunnitelmista, voidaan käyttää erilaisia tapoja antaa palautetta. Ryhmät voidaan esimerkiksi vastuuttaa antamaan palautetta yhdelle toiselle ryhmälle esimerkiksi ääninauhoitteella, videolla tai kaaviokuvana.

# Pelinteko ja tuotekehitys

**Sanallista** tai **piirrä kaavio** pelisi toiminnasta:

**Valmistaudu esittelemään suunnitelmasi opettajalle ja muille oppilaille!**

## Rohkaisu

Oppilaat eivät aina usko omiin kykyihinsä. Tässä voi auttaa, kun oppilaat ymmärtävät digitaaliseen tuottamiseen liittyvät epävarmuuden:

> ***"Tiedämme mitä haluamme tehdä, mutta emme vielä tiedä miten se tehdään. Kunhan tiedämme mistä aloittaa, projekti suunnittelee itsensä samalla kun me opimme."***

# Voit nyt alkaa luomaan ja ohjelmoimaan!

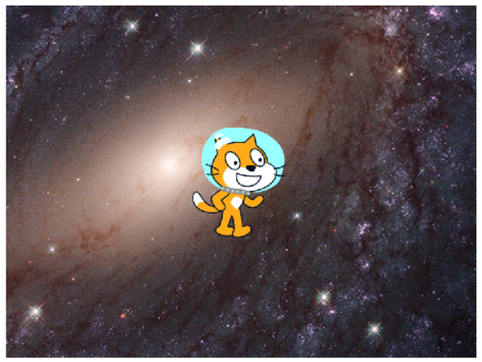
## 💡 Vinkkejä oppimispelin tekemiseen

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**.


### Projektityö alkaa

Suunnittelun ja suunnitelmien esittelyn jälkeen aloitetaan projektien prototyyppien tekeminen. Työskentelylle kannattaa antaa reilusti aikaa, mutta aikaraja on myös tärkeä asettaa.

### Prototyypin aikaraja

Aikaraja (deadline) nähdään usein puuduttavana ja luovuutta syövänä asiana. Se on kuitenkin tärkeä osa projektityötä: Se auttaa meitä tekemään kompromisseja, jotta projekti tulee oikeasti valmiiksi. Tässä vaiheessa oppilaiden täytyy saada tehtyä pelin prototyyppi, jota voidaan testata kohderyhmällä. Ammattitermi tälle prototyypille on *Minimum Viable Product (MVP)* eli *Pienin toimiva tuote*.

### Ohjeita oppilaille

**Tehkää yhteistyötä:** Oppilaat työskentelevät pääasiassa omassa tiimissään/ryhmässään. Heille voidaan myös osoittaa “serkkuryhmä”, jonka kanssa he tekevät tiivistä yhteistyötä ideoiden jakamiseen ja ongelmanratkaisuun liittyen.

**Pyydä apua**: Kun saman ongelman kanssa tuntuu menevän aivan liian kauan aikaa, on hyvä pyytää apua toisilta oppilailta ja opettajalta, jotta ei jäädä jumiin.

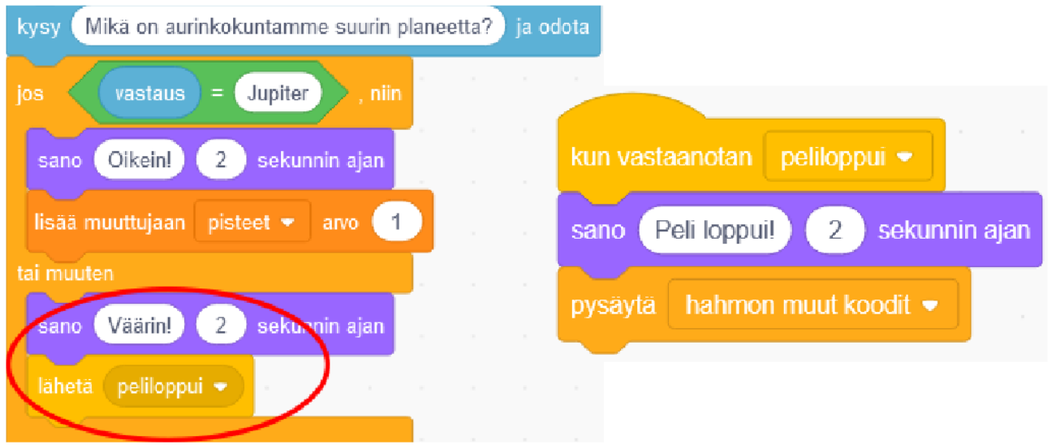
**Auta toisia:** Tarjotaan omaa apua muille. Kyseessä ei ole kilpailu. Jokainen voi olla opettaja toiselle.

**Ole rohkea:** Luovuus ja erikoisimmatkin ideat saavat välittyä lopulliseen tuotokseen. Näin projektista tulee aidosti ryhmän oma tuotos.

### Vinkit

Sivuilla 46-50 on vinkkejä pelin ohjelmointiin ja kehittämiseen liittyen.

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...


## Opettajan rooli

Tämän kaltaisissa projekteissa opettaja toimii usein *mentorina* tai *työnohjaajana*.

Roolissa tärkeintä on...

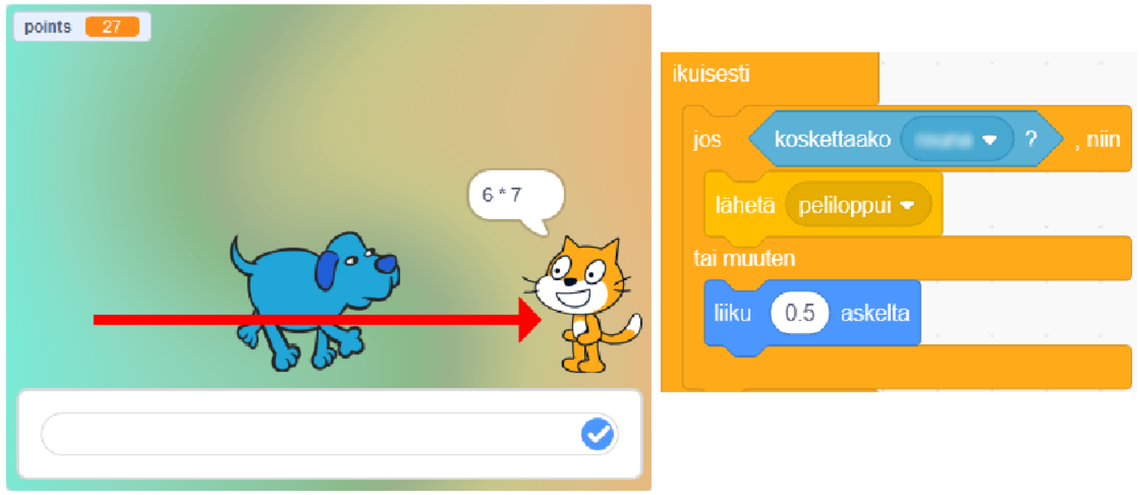
- Auttaa ryhmiä löytämään toisensa avuntarpeen noustessa
- Nostaa usein kysyttyjä kysymyksiä yhdessä käsiteltäväksi
- Arvioida yksittäisten oppilaiden aktiivisuutta ja työpanosta ryhmän tavoitteiden saavuttamiseksi
- Tarjota palautetta ja kehittämisehdotuksia ryhmille ja yksittäisille oppilaille

# Arviointi

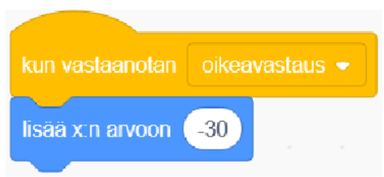
## Havainnoi oppilaita

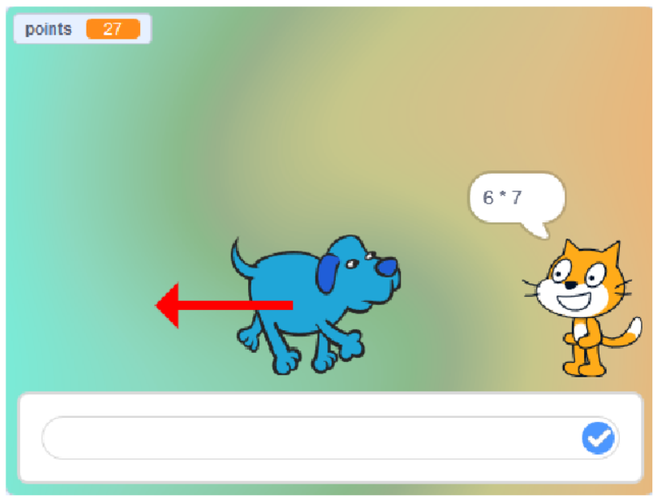
Projektivaihe on arvioinnin kannalta otollinen tilaisuus oppilaiden havainnoinnille. Arvioinnin tukena voidaan käyttää oppimistavoite taulukkoa tai vastaavaa työkalua. Oppimistavoitteet kannattaa ottaa oppilaiden kanssa esille oppimiskokonaisuuden eri vaiheissa, erityisesti kun heiltä odotetaan vertaisarviointia.

...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.

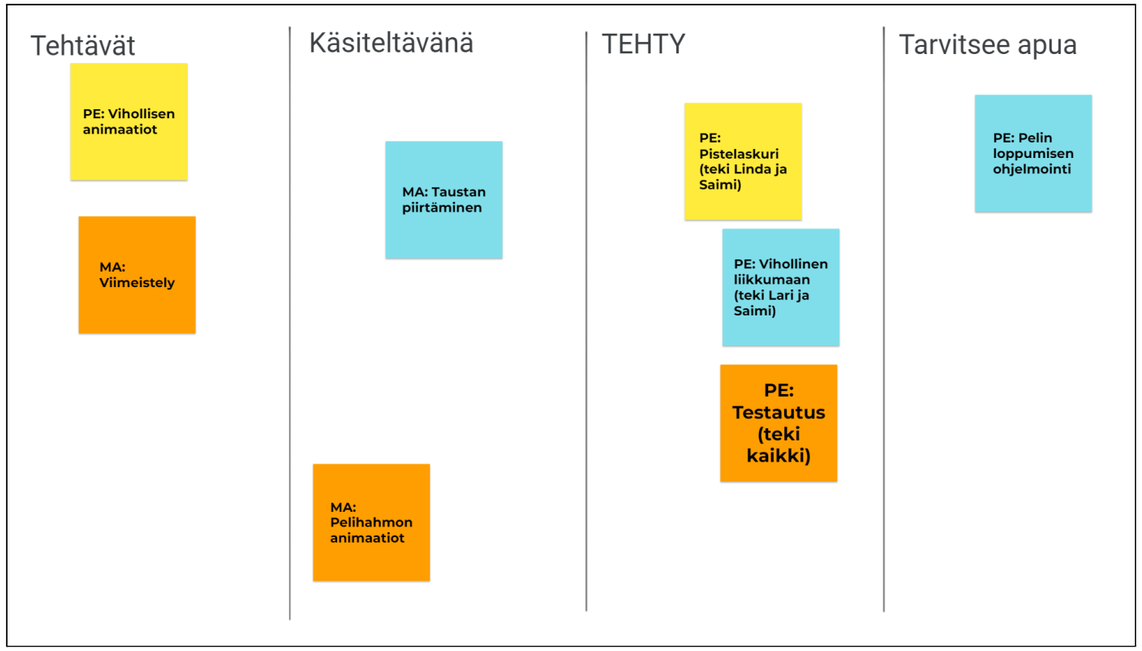



## Projektin hallinta - Kanban-taulu

Opettaja voi esitellä oppilaille projektinhallintatyökaluja, jotka auttavat projektin suunnittelussa, toteuttamisessa ja aikataulutuksessa. Yksi helppo esimerkki on Kanban-taulu. Siinä projektin tehtävät jaetaan esimerkiksi osioihin *Tehtävät, Käsiteltävänä, Tehty* ja *Tarvitsee apua.* Ryhmän jäsenillä voi olla oma väri, josta tunnistaa, kenen vastuulla mikäkin tehtävä on. Tehtäviä tehdään silti yhdessä. Kun yksittäinen tehtävä on tehty, siihen voidaan kirjoittaa ketkä sen lopulta tekivät.

Vaikka tämä työkalu onkin oppilaita varten, se auttaa myös arvioinnissa. Opettaja voi arvioida tauluista kunkin oppilaan työpanosta ja siihen voidaan palata vertaisarvioinnin näkökulmasta vielä projektin lopussakin.




*Kanban-taulu voidaan toteuttaa kartonkia ja tarralappuja käyttäen tai esimerkiksi PowerPointilla.*

## Palaute: Projektien esittely ja testaus

Pelien prototyypit kannattaa esitellä luokan kesken: Niitä voidaan esimerkiksi testata ristiin ryhmien kesken tai esitellä yksi kerrallaan koko luokalle.

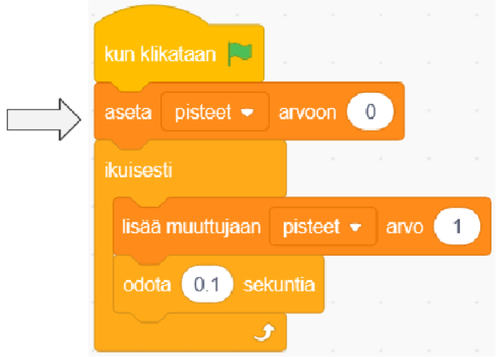
Tämän voi toteuttaa esimerkiksi koko luokan kesken tai ryhmissä (3-5 tuotosta per ryhmä). Projekteista annetaan palautetta ainakin seuraaviin kysymyksiin vastaten:

- Mikä esitellyssä projektissa oli parasta?
- Mitä parannettavaa projektissa on?

## 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?*

## Arviointi

### Välikoe

Opettaja voi halutessaan mitää oppilaille ***välikokeen*** osion yksi päätteeksi (erillinen tiedosto). Kokeen voi teettää yksin, projektiryhmissä tai pareittain.

**Arvosanat (esimerkki):**

| Pisteet | Arvosana |
|---|---|
| 0-10 | *Alle minimirajan* |
| 11-18 | Perustaidot |
| 19-25 | Hyvät taidot |
| 26-31 | Erinomaiset taidot |

### Välikoe, tehtävän 1 ratkaisu

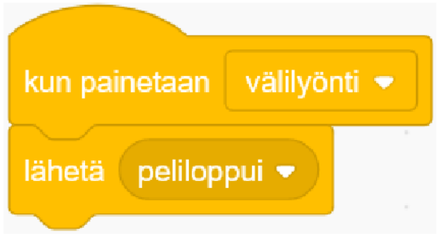
**1** Valitse **tosi**, **epätosi** tai **en tiedä**

| | Tosi | Epä-tosi | En tiedä |
|---|---|---|---|
| Scratchissa annetaan toimintaohjeita hahmoille. | X | | |
| Kahden arvon avulla (x ja y), hahmoa voidaan liikuttaa mihin vain 2-ulotteisella alustalla. | X | | |
| Mikäli sama tapahtuma on kahdesti samalla hahmolla, vain toinen niistä suoritetaan. | | X | |
| Totuusarvo voi olla tosi, epätosi, puoliksi tosi tai 0. | | X | |
| Muuttujiin voidaan varastoida tietoa. | X | | |

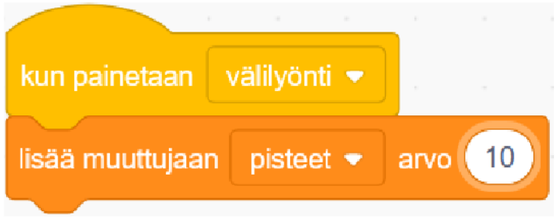
## Pikanäppäimet ja oikotiet

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!

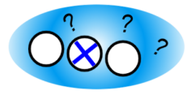
Klikkaamalla esiintymislavalla näkyvää muuttuja-kuvaketta, voit muuttaa sen esitystapaa.


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

---

Välikoe, tehtävän 2 ratkaisu

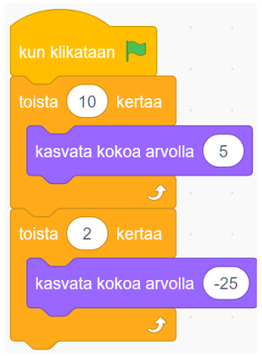
**2** Valitse **paras vaihtoehto** kuhunkin kysymykseen



**a) Katso alla olevaa koodia.**
**Kuinka monta askelta hahmo liikkuu tällä skriptillä?**

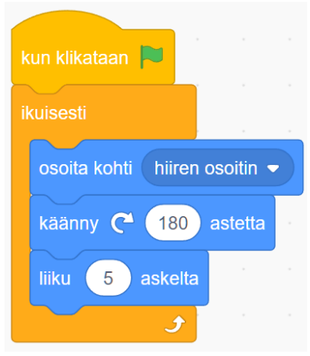

- 20 askelta ○
- 50 askelta ⊗
- 100 askelta ○
- 200 askelta ○

**b) Katso alla olevaa koodia.**
**Jos hahmon koko on aluksi 100, mikä on sen koko tämän skriptin päätyttyä?**




- 1 ○
- 10 ○
- 50 ○
- 100 ⊗

**c) Katso alla olevaa koodia.**
**Miten hahmo käyttäytyy tällä skriptillä?**



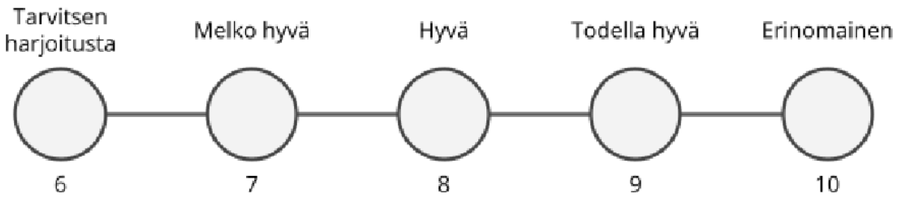

- Se alkaa paeta hiiren osoitinta ⊗
- Se ei tee mitään ○
- Se alkaa seurata hiiren osoitinta ○
- Se alkaa pyörimään ympyrää loputtomasti ○

# 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!



Välikoe, tehtävän 3 ratkaisu

**3** Viimeistele "***Ajastinohjelma***"

**Tavoite:**

Kun vihreää lippua klikataan, hahmo laskee alaspäin **aloittaen luvusta 60 ja lopettaen lukuun 0**. Aikaa tähän kuluu 60 sekuntia.


**Katso alla olevaa koodia.**
**Kirjoita oikeat numerot** niin, että koodi toimii kuten yllä kuvaillussa tavoitteessa (yhteensä 4 numeroa).



**Ratkaisu: 61, 1, -1, 1**

*Seuraavan sivun alussa asiaa itsearvioinnista.*

# 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!**



### *Arviointi*

#### Itsearviointi

Itsearviointiin oppii vain, jos sitä pääsee tekemään. Sivulla 51 oppilaita ohjataan asettamaan itselle tämänhetkinen arvosana. Tässä on tarkoitus käyttää tähän oppimiskokonaisuuteen suunniteltua ***Taulukkoa oppimistavoitteista*** (erillinen tiedosto) tai opettajan itse kokoamaa tavoitteistoa. Pääasia on, että oppilaat tietävän oppimistavoitteet ja heidän itselleen antama arvosana perustuu niihin.

---

## **Osa 2**, *sivut 52-77*

### Yleiskatsaus

Osassa 2 syvennytään ohjelmoinnin perusrakenteiden käyttöön ja animointiin, toteutetaan pelien testautus kohderyhmällä sekä jatketaan omia projekteja prototyypistä täysversioiksi.

### Ajankäyttö

Osa 1 koostuu noin 12-14 oppitunnista (45 min/oppitunti).

### Tarvikkeet

- Tietokone, näppäimistö, hiiri (väh. 1/oppilaspari)

### Tärkeää osassa 1

- Suo oppilaille mahdollisuuksia keskittyä pelien kehittämisessä heitä kiinnostaviin asioihin
- Tarjoa mahdollisuus testauttaa peliä kohderyhmällä
- Luo mahdollisuuksia yhteistyölle ryhmien välillä

---

### Muistutus

Sivuilla 66-67 ohjataan keräämään palautetta pelin prototyypistä. Mikäli peleillä on kohderyhmä, testautuksen päivämäärä on hyvä sopia ja viestiä myös oppilaille.

# 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.

   _____

   _____

   _____

## Aloitus

Vaikka projektilla onkin kohderyhmä, ei sovi unohtaa itse tekijöitä. Oppilaiden omat kokemukset ja kiinnostuksen kohteet saavat valua projektiin.

Oppilaat asettavat itselleen yhden lupauksen, johon palataan oppimiskokonaisuuden lopussa.

# **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**.

# **Luetaan** - Sisäkkäiset toistot & ehdot

## Ajankäyttö

10 min

## Työtapa

Työskennellään **2-3** hengen yksiköissä, lukien ja keskustellen

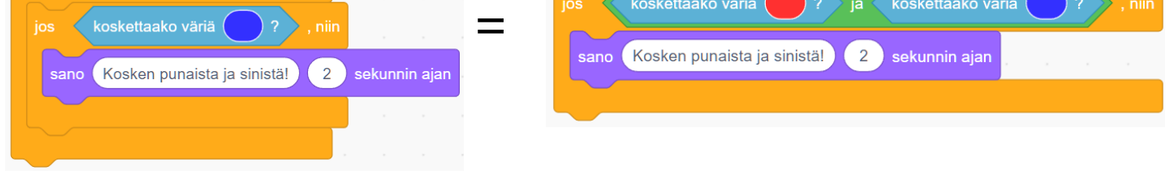
## Avainsanat

*Toisto, silmukka, ehtolause. sisäkkäinen, rakenne*

---

## JA-operaattorin käyttö

Sisäkkäisyyden sijaan usein on selkeämpää käyttää JA-operaattoria. Alla esimerkki eri tavalla rakennetuista skripteistä, jotka kuitenkin toimivat täysin samalla tavalla.



# **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**.

# **Ohjelmoidaan** - Silmukoiden ja ehtolauseiden sisäkkäinen asettelu

## Ajankäyttö

1 oppitunti

## Työtapa

Työskennellään **1-3** hengen yksiköissä, mieluiten pareittain

## Avainsanat

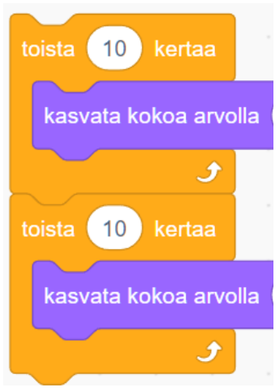
*Toisto, silmukka, ehtolause, sisäkkäinen, rakenne, kynä, geometria*

---

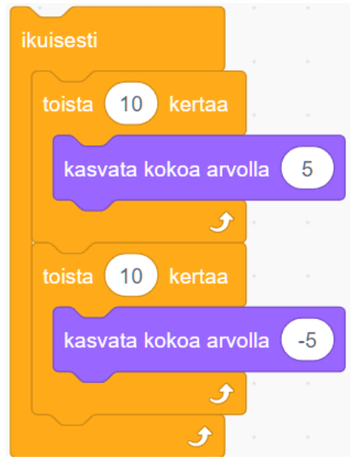
## Sisäkkäisyyden hyödyt

Alta löydät esimerkin sisäkkäisyyden hyödystä. Skripti A kasvattaa hahmoa hieman ja pienentää takaisin alkuperäiseen kokoon. Laittamalla tämän rakenteen kokonaisuudessaan ikuiseen toistoon, saadaan tämä tehoste tapahtumaan jatkuvasti (skripti B). Näin saadaan aikaan sykkivä animaatio.

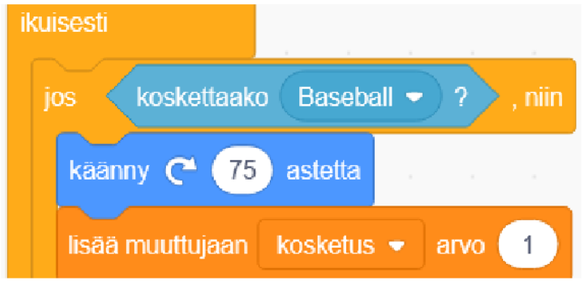
***A)***

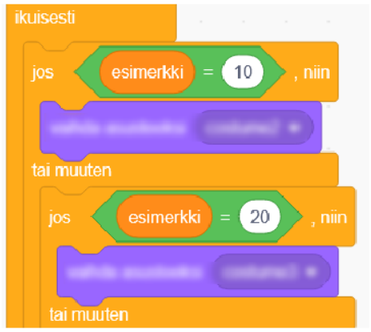



***B)***



**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

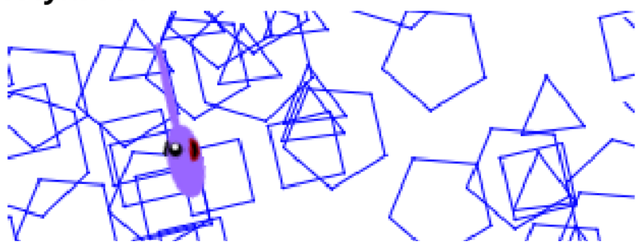
1. **Mene selaimella tähän osoitteeseen: codeschool.fi/pt13**
2. Klikkaa **Katso sisälle** -nappia.

   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**.

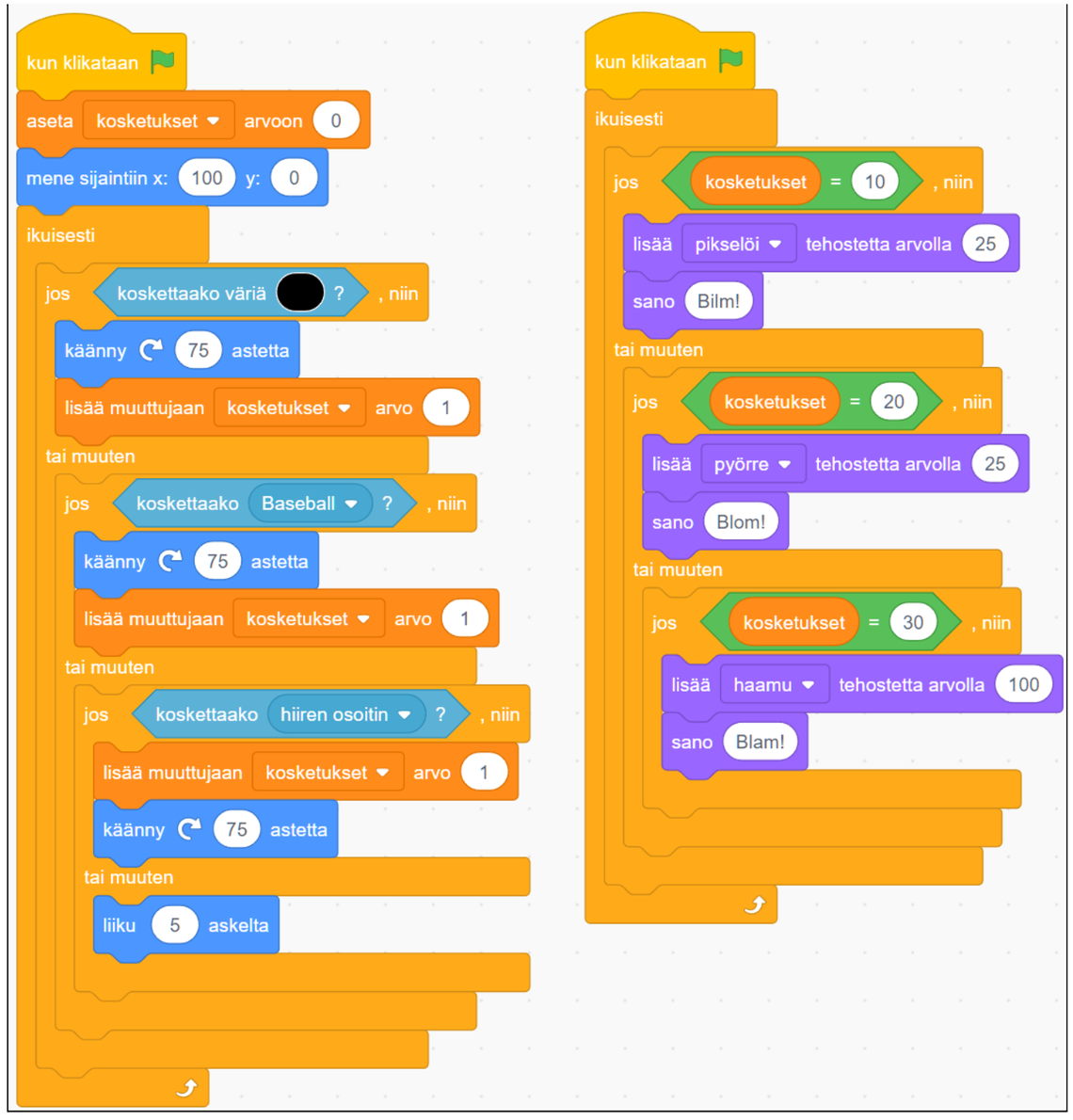
**Lohkot** on aseteltu valmiiksi ohjelmointialueelle. Tehtäväsi on **asetella ne oikeaan järjestykseen**.





Esimerkkiratkaisut sivujen 55-56 **haasteisiin 1-2**

**Haasteet 1 ja 2** (esimerkki, jossa molempien ratkaisu):



## Vinkkejä haasteisiin

- **Haasteessa 1** Mr. Nosey saattaa jäädä jumiin. Muuta käännä ( ) 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 käännä ( ) astetta-lohkoja toista ( )-lohkon sisällä.


Piirtäminen tehdään **yksi sivu kerrallaan**.

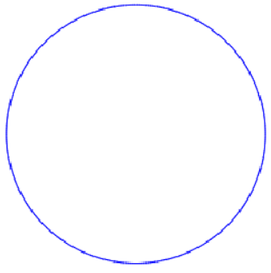
Kolmiossa on **kolme sivua**, joten aseta arvo **3** toista ( )-lohkoon.

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 käännä ( ) astetta-lohkoon. *(360 / 3 = 120)*

### Miksi jaamme luvun 360?

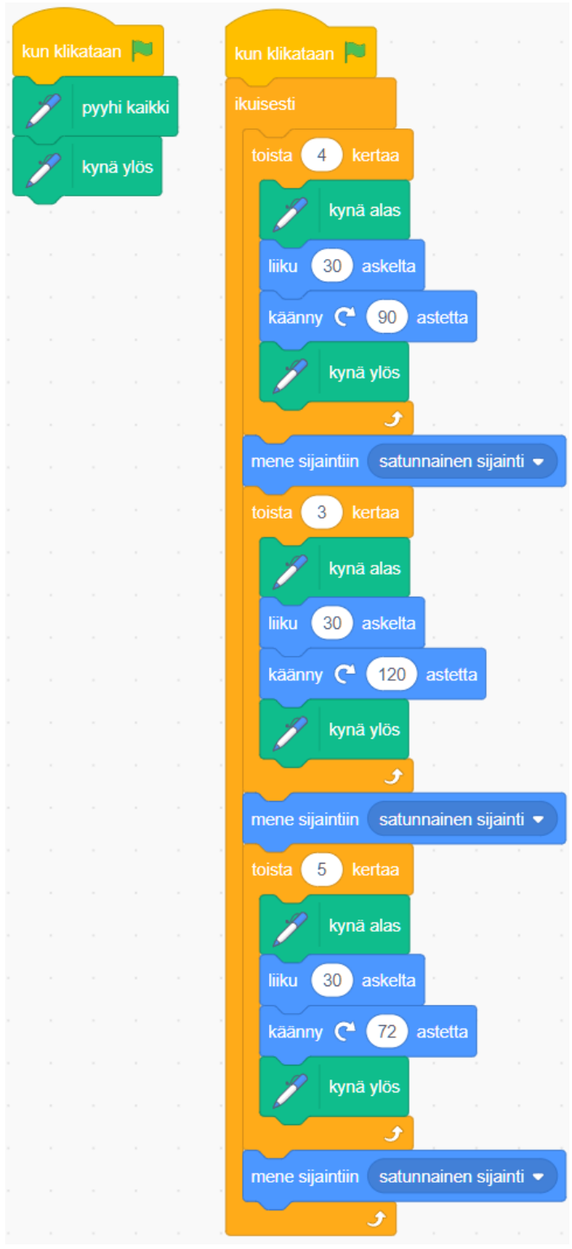
Kun kuljetaan kokonainen kierros, käännytään aina yhteensä **360** astetta. Jokaisen muodon, joka kulkee kokonaisen kierroksen ympäri, **kulmien asteiden yhteenlaskettu summa on aina 360**. Esimerkiksi **suorakulmaisessa neliössä** on **neljä 90 asteen kulmaa**, jotka yhteenlaskettuna muodostavat luvun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





Esimerkkiratkaisut sivun 56 **haasteeseen 3**

**Haaste 3** (esimerkki):



# **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**.

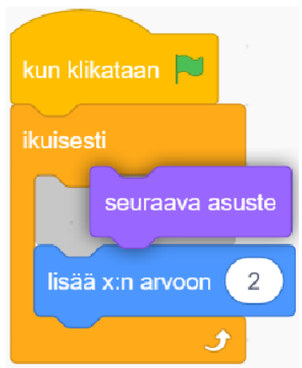
Seuraavaksi harjoitellaan animointia. **Luo uusi Scratch projekti** ja anna sille nimeksi *"animointiharjoitus"*.

## *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!



## **Ohjelmoidaan** - Asusteet & animointi

### Ajankäyttö

2 lesson (45 min each)

### Työtapa

Työskennellään **projektiryhmissä**

### Avainsanat

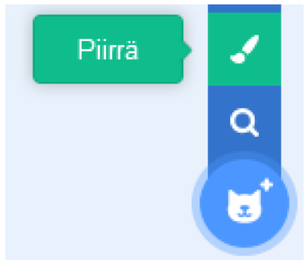
*Asuste, animaatio, piirrä, ryhmäytys, uudelleenmuotoilu, monistus*

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**!

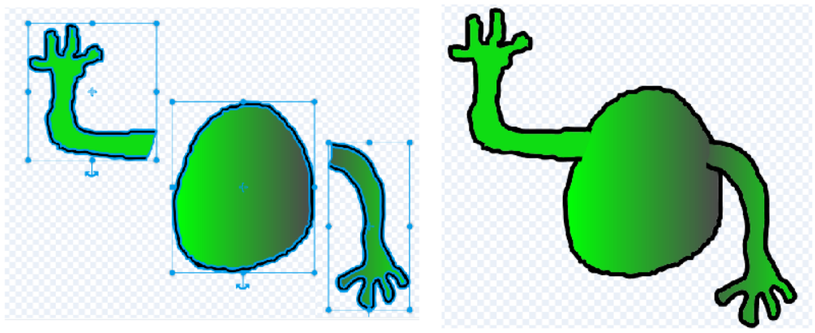
## *Testaa ja tutki!*

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




### Vinkkejä hahmojen piirtämiseen

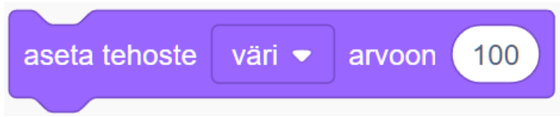
➪ Kun luot hahmoja, piirrä osat joita haluat liikutella animaatioissa **erillisinä osina**. Tällä tavoin helpotat osien liikuttelua ja muokkaamista kun aloitat animoinnin.



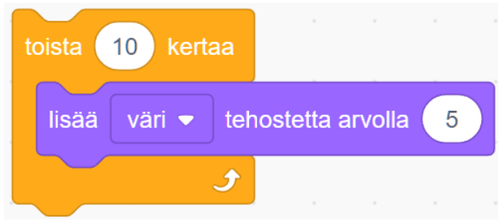
## Animointi tehosteilla

Animoinnissa voidaan käyttää myös graafisia tehosteita:

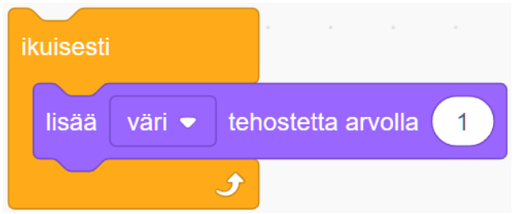
***Aseta tehoste []*** -lohkolla asetetaan tehosteen vahvuus suoraan tiettyyn arvoon:



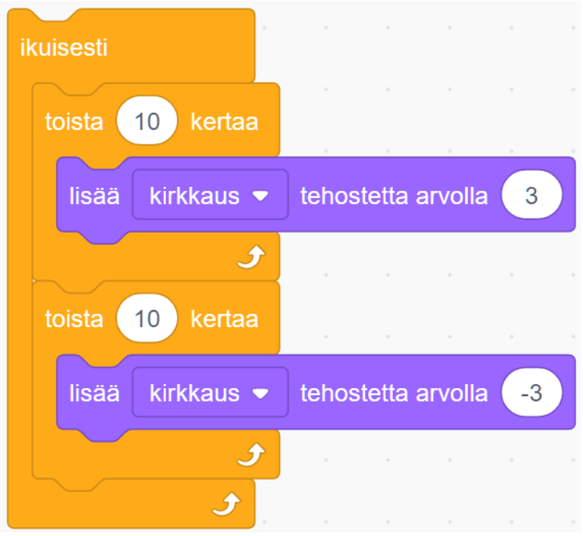

***Lisää [] tehostetta*** -lohkolla voidaan muuttaa arvoa tietyn arvon verran. Yhdistämällä tämän määrättyyn toistoon, saadaan tehosteen muutoksesta liukuvia.




Voidaan myös käyttää ikuista toistoa, jolloin tehosteen muutos on jatkuvaa:

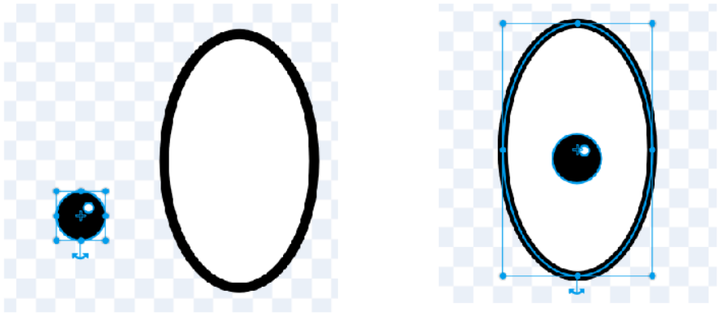



Kun tehostetta liu'utetaan tiettyyn arvoon ja takaisin, saadaan näyttäviä sykkiviä tehosteita:



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

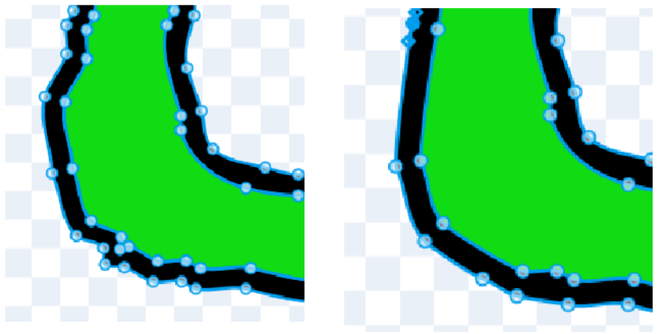
  Voit aina halutessasi myös **purkaa ryhmityksen**.





---

- 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!*

## Bittikartta vs vektori

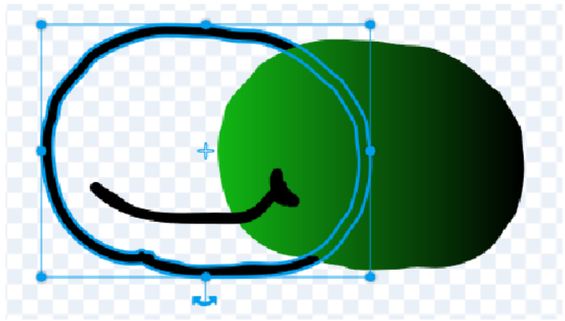
Piirtoalueen alapuolella on painike “**Muunna bittikartaksi**”. Sitä klikatessa hahmo muuttuu yksinkertaiseksi piirrokseksi, jossa eri objekteihin ei enää pääse käsiksi. Tämä vaikeuttaa esimerkiksi hahmon animointia huomattavasti. Lisäksi hahmon tarkkuus laskee ja siitä tulee pikselinen. **Välttele tätä painiketta** ellei sille tule erityistä tarvetta.


**Jos oppilas klikkaa sitä**, muutoksen voi peruuttaa **peruutus**-painikkeella tai näppäinyhdistelmällä **ctrl+z**

*Vektoripiirros*

*Bittikarttapiirros*

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!**

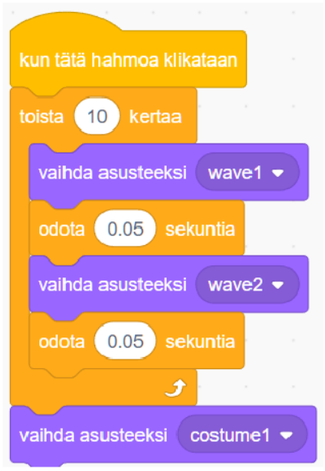
## Hahmojen animointi

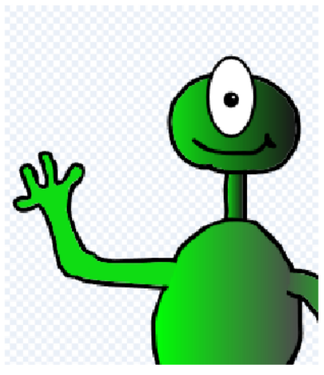
Scratchissa animaatiot tehdään **luomalla useita hieman erilaisia asusteita** hahmoille ja sitten **vaihtamalla asusteita skriptin avulla**. Animoinnissa pätee nyrkkisääntönä **mitä useampi asuste**, **sitä sulavampi animaatio**. Joskus sillä on väliä, joskus taas ei.

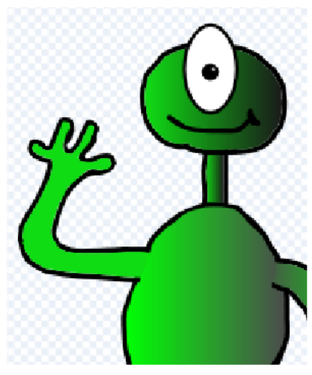
Voit animoida **liikuttelemalla hahmon osia** tai **muotoilemalla niitä**. Harjoitellaanpa!

### 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.





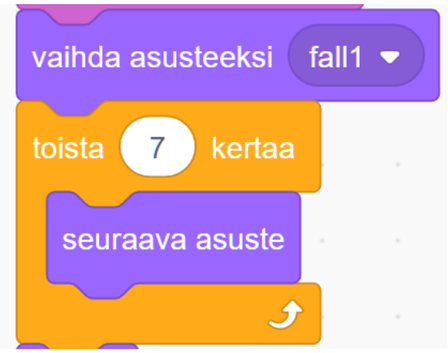
## Animaatiot

Mitä enemmän kuvia animaatiossa on aikayksikköä kohden, sitä sulavampi animaatio on. Esimerkiksi elokuvissa on pitkään näytetty 24 kuvaa per sekunti. Kuvataajuudesta käytetään puhekielessä termiä fps (frames per second), joka tyypillisestä halutaan saada moderneissa peleissä lähemmäs lukua 60.

Jos Scratchissa halutaan suhteellisen sulavia animaatioita, kannattaa pyrkiä tekemään isommalle animaatiolle ainakin 12 asusteenvaihtoa (12 erilaista asustetta). Alla esimerkki animaatiosta, jossa hyvin nopea tapahtuma esitetään kahdeksalla asusteella. Animaatio tapahtuu kun klikataan jäälauttaa.

- Pingviini putoaa veteen: https://codeschool.fi/pt24

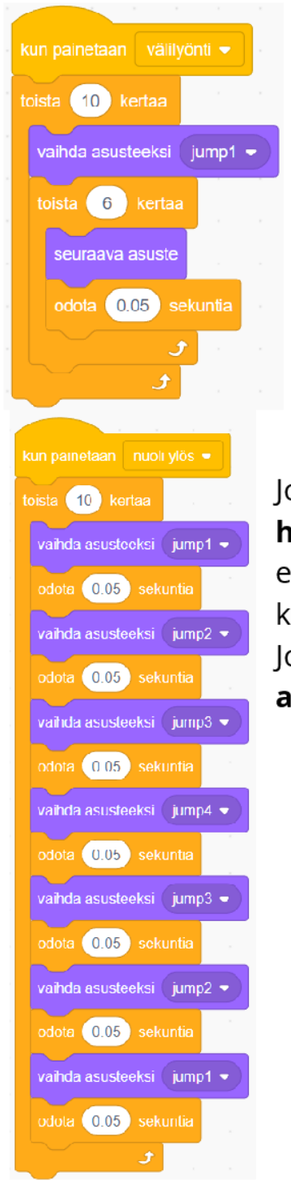
Mitä vähemmän asusteita, sitä nopeampi animaatio on. Tätä voidaan kompensoida laittamalla lyhyitä, 0.01s odotuksia, asusteen vaihtojen väliin, mutta tämä saattaa helposti saada animaation näyttämään pätkivältä.




*Tietty animaatio saadaan helposti tapahtumaan oikein, kun asetetaan oikea ensimmäinen asuste ja vaihdetaan asustetta niin sopivan monta kertaa toistorakenteella. 8 asusteen animaatio saadaan tapahtumaan kokonaan, kun asetetaan ensimmäinen asuste ja vaihdetaan asustetta 7 kertaa.*

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

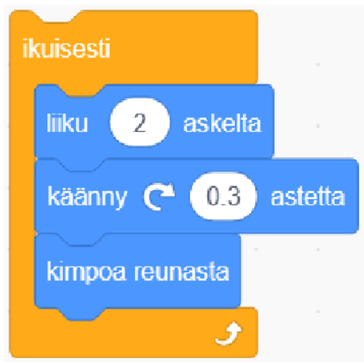
- **Luo varmuuskopioita** hahmoista. Aina silloin tällöin asusteita muokatessa jokin menee pieleen. Jos näin käy, voit palata alkuperäiseen hahmoon varmuuskopion avulla. Tee ainakin yksi varmuuskopio!


---

- 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.



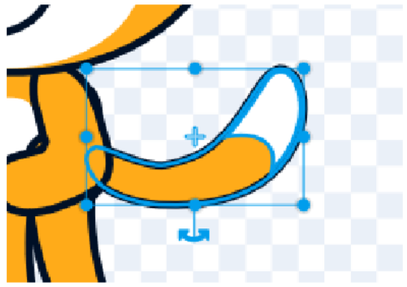

---

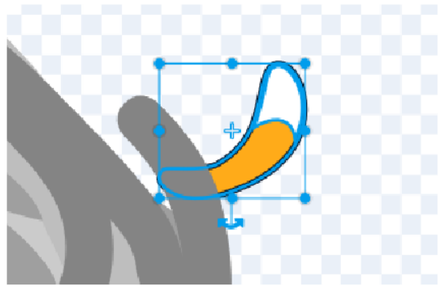
- Voit käyttää animaatioissa myös ihan jotain muuta, kuin hahmon alkuperäisiä asusteita!

  Mene selaimella osoitteeseen [codeschool.fi/pt15](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!*

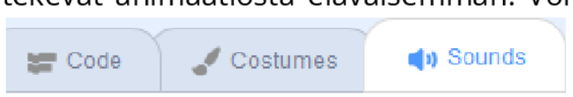




Jotkin **Cat** hahmon osat on kopioitu **Dog2** asusteisiin, joten animaatio näyttää siltä kuin molemmat hahmot olisivat yhä näkyvissä. Siistiä, eikö?

---

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



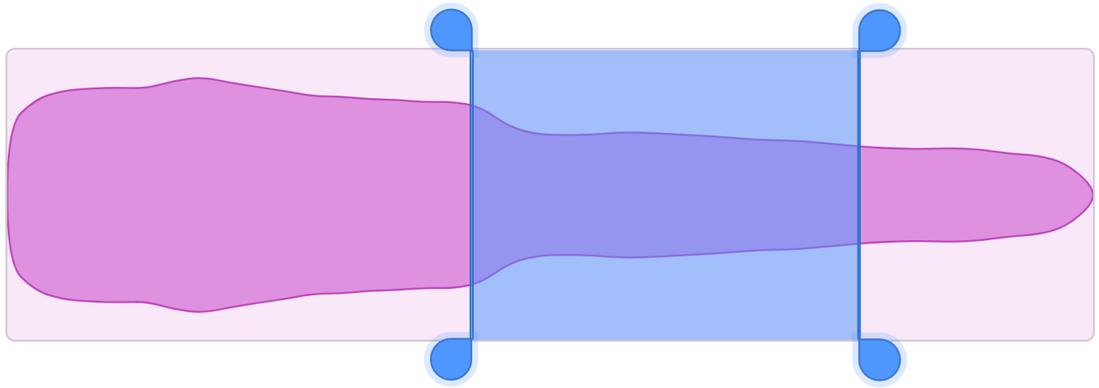

---

## Hyvä tietää äänien käytöstä

Hahmojen äänet toimivat kuten asusteet: Jokaisella on omansa ja ne täytyy hahmoille hakea erikseen. Ääniä voidaan myös äänittää käyttämällä tietokoneen mikrofonia.

Ääni voidaan myös muokata leikkaamalla tai käyttämällä efektejä:


Efekti voidaan määrätä tiettyyn osaan ääntä maalaamalla se hiiren vasemmalla näppäimellä ja klikkaamalla haluttua efektiä.

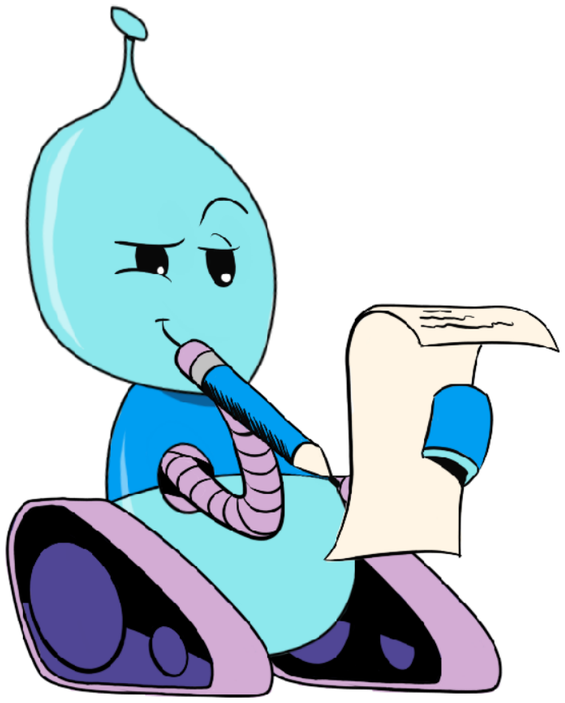


# Testaaminen ja palautteen kerääminen

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 →*

# Testaaminen ja palautteen kerääminen

## Ajankäyttö

1-2 oppituntia

## Työtapa

Työskennellään **projektiryhmissä**

## Keywords

*Testaus, palaute, prototyyppi, data*

---

## Testautuksen järjestäminen

Oppilaat tarvitsevat palautelomakkeita testautusta varten. Heille voidaan tulostaa sopiva määrä oppilaan kirjan lomakeita (sivu 67) tai valmistella he voivat valmistella kyselyn itse muulla tavalla.

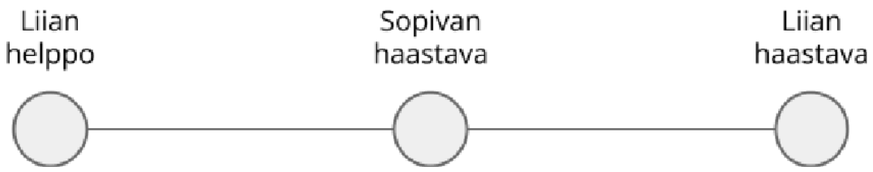
Testautuksen voi järjestää monella tavalla. Tässä **kaksi esimerkkiä**:

- **Jokaiselle ryhmälle oma testiryhmä:** Kullakin ryhmällä on oma testiryhmä, joka kokeilee vain heidän peliä.
- **Järjesteetään testimessut:** Kohderyhmän edustajat kiertävät vapaasti testaten pelejä. Pelin tehneet keräävät palautetta oman pelinsä äärellä. Opettajat pitävät huolen, että jokainen ryhmä saa palautetta vähintään kolmelta kohderyhmän edustajalta.

## 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)

## Palautelomake

Tämän lomakkeen voi tulostaa tai sen pohjalta voi tehdä oman kyselyn.

# **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**

# **Projekti** - Oppimispelin versio 1.0

## Ajankäyttö

4-6 oppituntia

## Työtapa

Työskennellään **projektiryhmissä**

## Avainsanat

*Projekti, suunnittelu, prototyyppi, muotoiluajattelu, kohdeyrhmä, palaute, teema*

---

## Palautteen hyödyntäminen

Koska oppilaiden tulisi harjoitella myös kehittämisen taitoja, palautetta tulee hyödyntää jollain tavalla.

Seuraavat ohjeet saattavat auttaa oppilaita:

- Keskittykää asioihin, joihin teillä riittää taitoja ja aikaa. Peliä ei kannata enää suunnitella täysin uudelleen.
- Joistain itseänne kiinnostavista asioista saattaa joutua luopumaan, jos se toimi saadun palautteen kanssa.
- Kysele toisten ryhmien suunnitelmista ja kehitysideoista. Ne saattavat hyödyttää teitä.

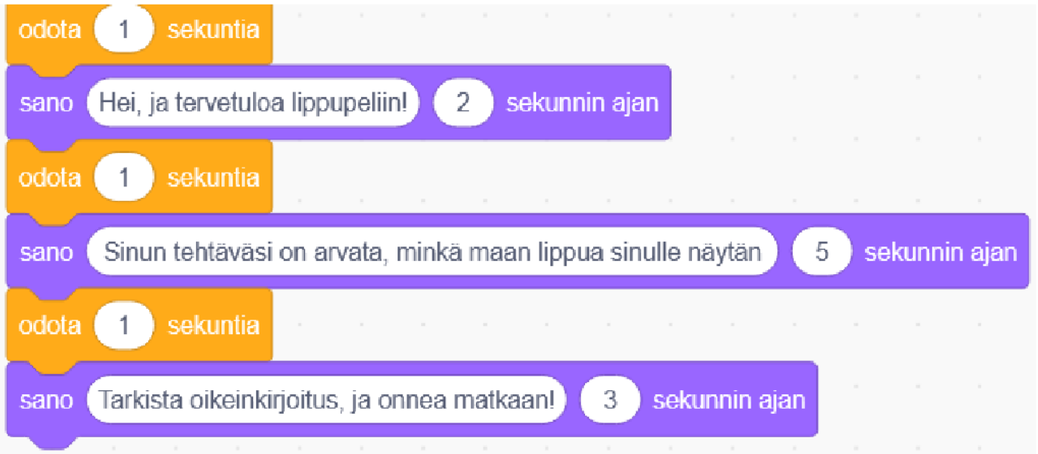
## Voit nyt aloittaa pelisi viimeistelyn!

### 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**!



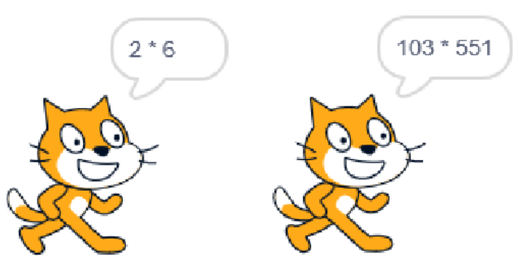

**Anna pelaajalle tarpeeksi aikaa lukea tekstit**! Lisää odota ( ) sekuntia-lohkoja hallitaksesi tekstien nopeutta.

---

#### 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**




#### Oppilaiden motivointi projektityössä (*kertaus*)

Kolme avaintekijää auttavat pitämään oppilaat motivoituneita oppimiskokonaisuuksissa, joissa rakennetaan pitkän ajan saatossa isompia projekteja:

- **Merkityksekkyys**: Oppilaille osoitetaan, että heidän tuotoksiaan tullaan käyttämään johonkin merkitykselliseen ja ne esitellään toisille. Oppilaat saavat tuotoksistaan palautetta oppimiskokonaisuuden aikana.
- **Itsensä toteuttaminen**: Oppilaat ymmärtävät, että he saavat käyttää luovuutta ja tehdä valintoja oman tuotoksensa suhteen. Oppilailla on riittävästi resursseja oman tuotoksen tekemiseen.
- **Tuki**: Oppilaat eivät tunne olevansa yksin, kun he tarvitsevat ohjausta. Oppilaita rohkaistaan yhteistyöhön ja sosiaalinen ympäristö pidetään turvallisena. Oppilas tuntee, että virheitä ei tarvitse pelätä eikä niistä rankaista.

#### Ehdotuksia merkityksellisyyden edistämiseksi (*kertaus*)

- Pelit **esitellään ja toisten pelejä testataan**. Oppilaat antavat positiivista palautetta ja kehittämisehdotuksia (rakentava palaute).
- Pelit esitellään **rinnakkaisluokalle tai nuoremmille oppilaille**. Oppilaat antavat positiivista palautetta ja kehittämisehdotuksia (rakentava palaute).
- Oppilaat **äänestävät parhaista tuotoksista** seuraavissa kategorioissa:
  - *Paras ulkonäkö*
  - *Hauskin peli*
  - *Paras kokonaisuus*

Voittajat saavat pienet palkinnot.

#### Esimerkkiprojekteja

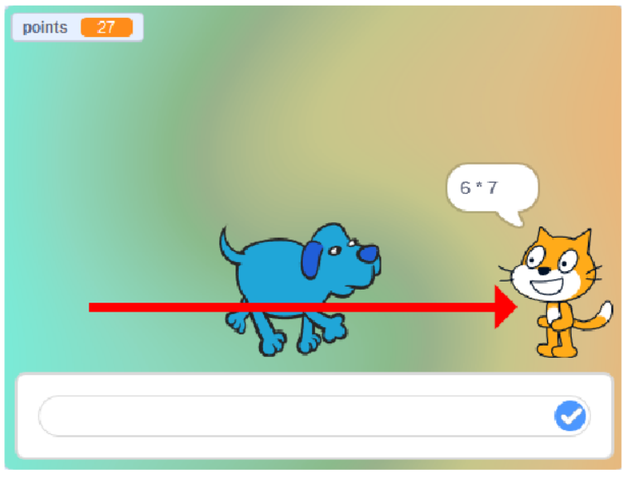
Oppilaat voivat edelleen hyödyntää näitä projekteja:

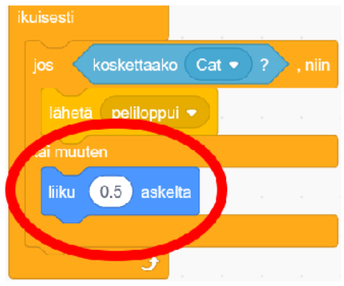
- https://codeschool.fi/pt11
- https://codeschool.fi/pt16
- https://codeschool.fi/pt17

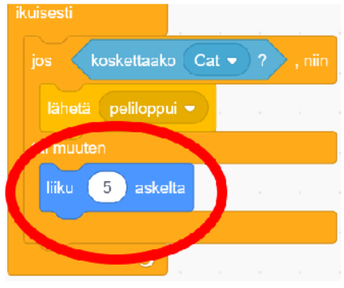
b) **Voit antaa pelaajalle vaihtoehtoja, joista valita tehdäksesi pelistä helpomman**


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


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



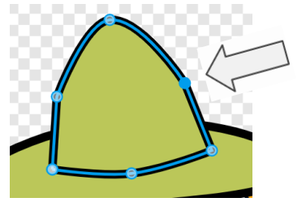



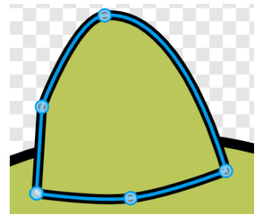




## Muotoilutyökalu

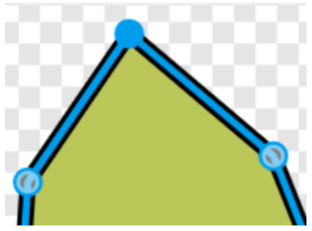
**Muotoile uudelleen**-työkalulla saadaan esille objektien mumotoilupisteet. Niiden liikuttamisen lisäksi voit tehdä seuraavaa:

- **Poistaa** pisteen kaksoisklikkauksella





- Valita, miten piste muuttaa muotoa (kaartuva vai terävä)

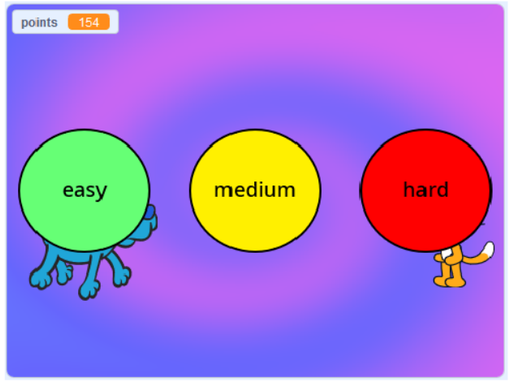



## Vaikeustason valintapainikkeet ***(vaativa)***

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

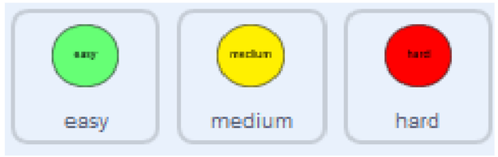
Mene osoitteeseen [codeschool.fi/pt16](codeschool.fi/pt16)

Klikkaa **Katso sisälle** -nappia.

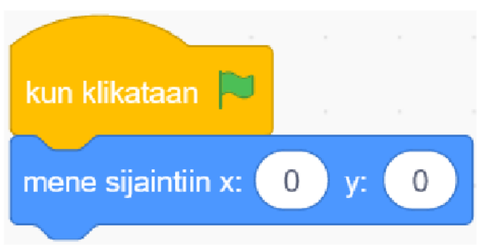


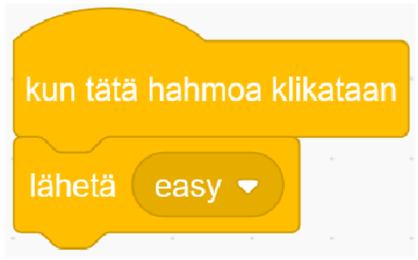

Tässä esimerkissä vaikeustason valinnalla **vaikutetaan Dog2 hahmon nopeuteen**. Mitä vaikeampi vaikeustaso valitaan, sitä nopeammin hahmo liikkuu kohti **Cat** hahmoa.

Vaikeustason valintanäppäimet ovat **omia hahmojaan**.




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

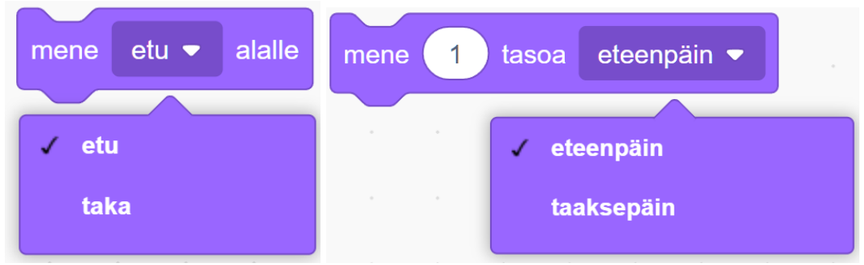



## Tasot näyttämöllä

Aikaisemmin käsiteltiin piirtoalueen tasoja, mutta myös näyttämöllä on tasot.

Hahmo voidaan siirtää suoraan taakse tai eteen ***mene [etu/taka] alalle*** -lohkolla. Hahmo voidaan siirtää tietyn verran eteen- tai taaksepäin käyttämällä ***mene () tasoa [eteenpäin/taaksepäin]*** -lohkoa.

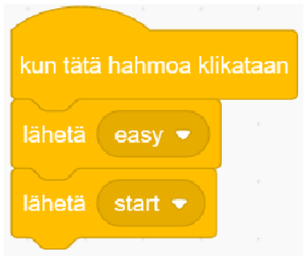



***Huomio:*** *Vertaisarviointiin siirrytään vasta kun projektit on tehty.*

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


*Vaikeustason vaikuttaa hahmon nopeuteen*

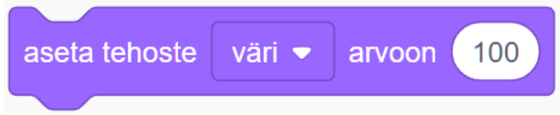
Että **peli ei alkaisi heti kun** 🚩 **kuvaketta klikataan**, vaan odottaisi että pelaaja valitsee vaikeustason, täytyy myös pelin alkamiselle tehdä oma **lähetys**.



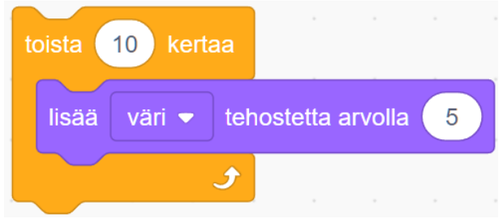

## Graafiset tehosteet

Animoinnin lisäksi tehosteiden käyttämisellä voidaan elävöittää peliä:

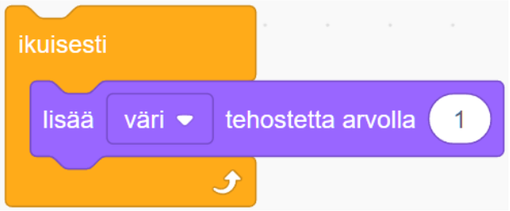
***Aseta tehoste []*** -lohkolla asetetaan tehosteen vahvuus suoraan tiettyyn arvoon:



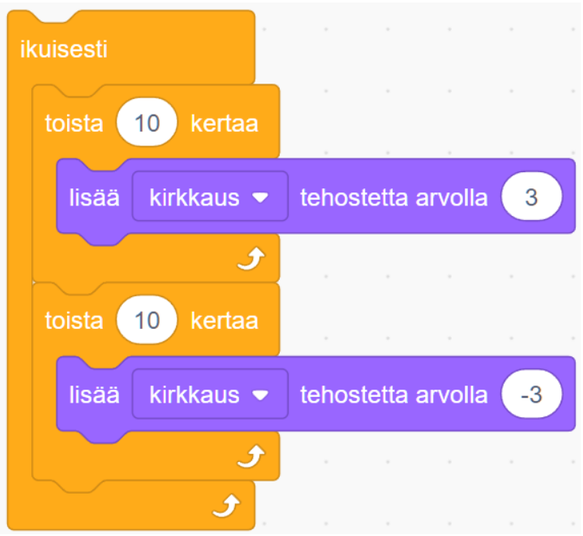

***Lisää [] tehostetta*** -lohkolla voidaan muuttaa arvoa tietyn arvon verran. Yhdistämällä tämän määrättyyn toistoon, saadaan tehosteen muutoksesta liukuvia.




Voidaan myös käyttää ikuista toistoa, jolloin tehosteen muutos on jatkuvaa:




Kun tehostetta liu'utetaan tiettyyn arvoon ja takaisin, saadaan näyttäviä sykkiviä tehosteita:



## 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.

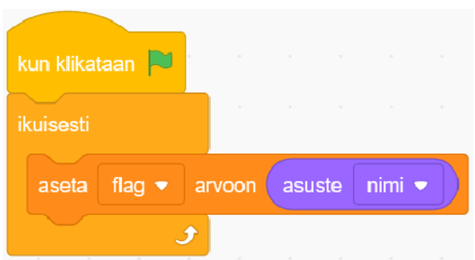
Avaa lippupeli, klikkaa **katso sisälle** kuvaketta ja katso koodia.


**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](codeschool.fi/pt15)

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

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

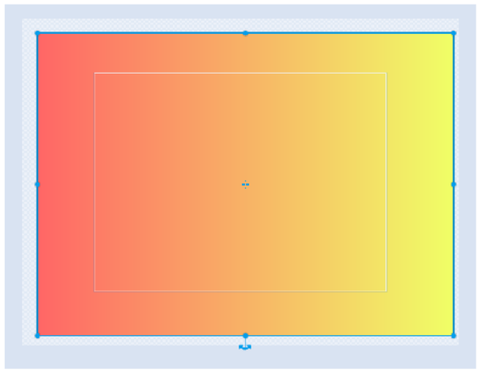
***Taustat:***

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


Valitse **Taustat**-välilehti.

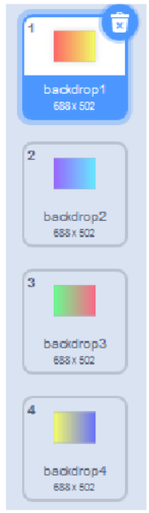

Käytä **suorakuolmiotyökalua** luodaksesi ison suorakulmion joka on **suurempi kuin esiintymislava**.

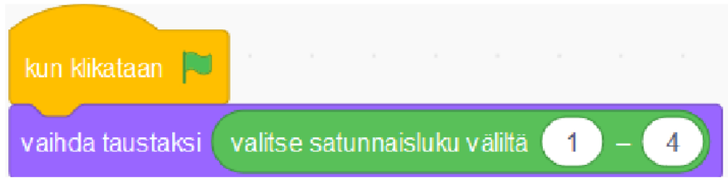

Käytä seuraavaksi **täyttötyökalua** valitaksesi taustallesi värin. Käytä eri **täyttövalintoja** jos haluat taustasi olevan jotain muuta kuin yksivärinen!



# Pelinteko ja tuotekehitys

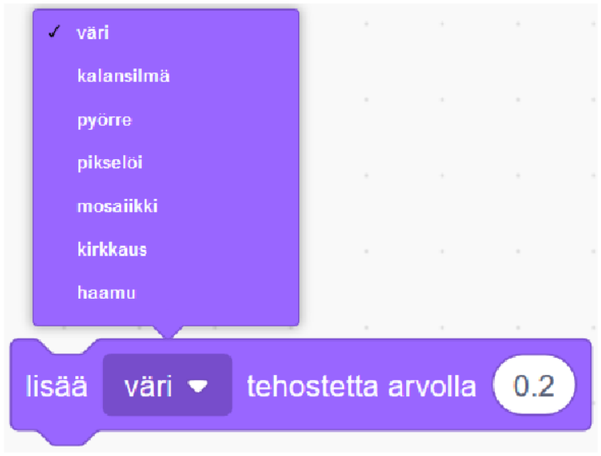
Voit lisätä peliisi useamman erilaisen taustan ja tehdä skriptin, joka valitsee pelin alussa satunnaisesti yhden taustoista pelin taustaksi.

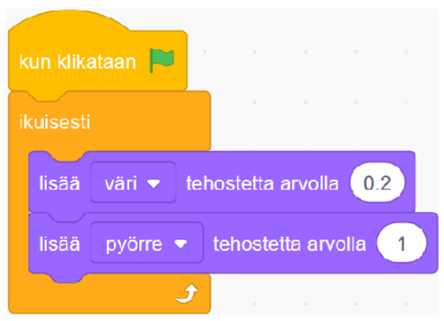



***Erikoistehosteet:***

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





## Syvyysvaikutelmaa varjoilla

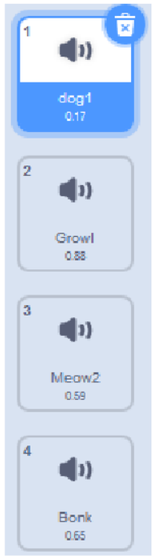
Opettaja voi esitellä oppilaille valmiin projektin, jossa syvyysvaikutelmaa on haettu varjoja käyttämällä. Varjot on saatu käyttämällä mustaa toisena värinä liukuvärissä.

- https://codeschool.fi/pt23

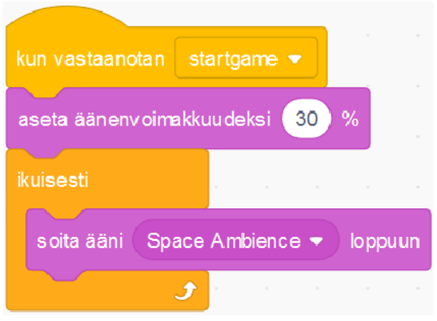
***Äänet:***

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.




## Palaute: Projektien esittely ja testaus

Kun projektit on tehty, ne **esitellään toisille oppilaille.**

Tämän voi toteuttaa esimerkiksi koko luokan kesken tai ryhmissä (3-5 tuotosta per ryhmä). Projekteista annetaan palautetta ainakin seuraaviin kysymyksiin vastaten:

- Mikä esitellyssä projektissa oli parasta?
- Mitä parannettavaa projektissa on?

**Vaihtoehtoisia tapoja esittelyyn:**

Oppilaat jättävät projektinsa auki tietokoneelle. Luokassa kierretään järjestyksessä projektilta toiselle. Jokaisen tietokoneen luona on kynä ja paperia, jolle kerätä kehuja ja kehittämisehdotuksia.

## “Tuotteen toimitus”

Oppilaille voidaan järjestää toinen testautus kohderyhmän kanssa. Se voi olla ikään kuin tapahtuma, jossa valmis tuote toimitetaan ja annetaan viimeiset kommentit. Todellisuudessa testausta tulisi tehdä sykleissä useita kertoja, mutta tähän ei välttämättä koulussa ole aikaa.

# *Arviointi*

## Tuotosten arviointi

Opettaja voi suorittaa tuotosten arvioinnin kahdella eri tavalla tai niiden yhdistelmällä:

A. **Vertaisarviointi:** Jokainen yksikkö/ryhmä antaa opettajan määrittämällä asteikolla arvosanan jokaisen ryhmän tuotokselle, myös omalleen. Dokumentoinnin helpottamiseksi opettajan kannattaa numeroida ryhmät. *Tässä arviointitavassa oppilaille täytyy tehdä selväksi, mitä he arvioivat: Ei esimerkiksi tuotosten tehneitä ihmisiä vaan sitä, kuinka hyvin projekti saavuttaa **projektin vaatimukset** ja mahdolliset opettajan asettamat lisävaatimukset. Arvioinnin kohteeksi voidaan ottaa myös esimerkiksi tuotoksen ulkonäkö digitaalisen piirtämisen näkökulmasta.*

B. Opettajan arviointi: Opettaja arvioi tuotokset **projektin vaatimusten** ja mahdollisten opettajan asettamien lisävaatimusten perusteella. Tavoitetaulukkoa voidaan myös käyttää, jos tässä vaiheessa ajatellaan Perustason vastaavan kouluarvosanaa 8.

   Arvioinnin tukena voidaan käyttää myös itsearviointia: Oppilas antaa arvosanan omalle työpanokselleen ryhmässä tehtyyn projektiin liittyen.

*Lopullisessa oppimiskokonaisuuden arvioinnissa käytetään seuraavaa kouluarvosanoihin vastaavuutta: Perustaidot kuvaa arvosanaa 6-7, Hyvät taidot arvosanaa 8 ja erinomaiset taidot arvosanoja 9-10.*

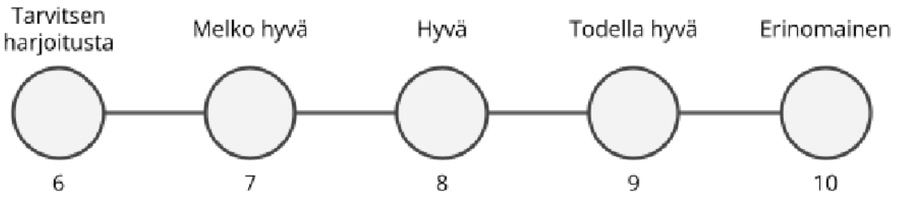
Projektiryhmissä voidaan myös antaa vertaispalautetta ***Vertaispalaute-lomakkeen*** avulla (erillinen tiedosto).

# Itsearviointi - Osa 2

**Vastaa seuraaviin kysymyksiin tehtyäsi kaikki osan 2 harjoitukset:**

1. Arvioi oma osaamisesi arviointiperusteiden mukaan.

   Keskustele arvioinnista parin kanssa!



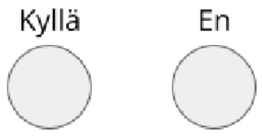

2. Mieti omaa opiskeluasi osaan 2 liittyen. Kirjoita itsellesi kolme kehua, jotka liittyvät toimintaasi ja onnistumisiisi.

   _____

   _____

   _____

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




   b) Miksi, miksi et?

   _____

   _____

   _____

## Loppukoe ja itsearviointi

### Ajankäyttö

1-2 tuntia

### Työtapa

Työskennellään projektiryhmissä, pareittain tai yksin

### Avainsanat

*Koe, arviointi, itsearviointi, reflektointi*

---

### *Arviointi*

### Loppukoe

Opettaja voi halutessaan mitää oppilaille ***loppukokeen*** oppimiskokonaisuuden päätteeksi (erillinen tiedosto). Kokeen voi teettää yksin, projektiryhmissä tai pareittain.

Tässä kokeessa oppilaat **remiksaavat** valmista ohjelmaa.

Opettajan kannattaa näyttää, miltä kokeen tehtävien mukainen ohjelma lopulta näyttää, muttei näyttää koodia oppilaille.

Linkki oikein tehtyyn ohjelmaan: **https://codeschool.fi/pt25652**

**Arvosanat (esimerkki):**

| **Pisteet** | **Arvosana** |
|---|---|
| 0-14 | *Alle minimirajan* |
| 15-24 | Perustaidot |
| 25-34 | Hyvät taidot |
| 35-40 | Erinomaiset taidot |

# Arviointi

## Arvosana

Arviointi perustuu oppimistavoitteisiin. Voit käyttää esimerkiksi ***Taulukko oppimistavoitteista*** -dokumenttia. Sivulla 77 oppilaat antavat itselleen arvosanan. Opettajan ja oppilaan näkemys arvosanasta voi olla eriävä, mutta molemmat on tärkeää ottaa huomioon.

## Arviointisuunnitelma

Alla kaksi esimerkkiä arviointisuunnitelmasta lopullisen arvosanan antamiseksi.

**Arviointisuunnitelma 1:**

| **Prosenttia arvosanasta** | **Arvioinnin kohde** |
|---|---|
| 33.3 % | Itsearviointi sivulla 77 |
| 33.3 % | Väli- ja loppukokeen arvosanat |
| 33.3 % | Opettajan arviointi arviointitaulukon perusteella |

**Arviointisuunnitelma 2:**

| **Prosenttia arvosanasta** | **Arvioinnin kohde** |
|---|---|
| 50 % | Itsearviointi sivulla 77 |
| 50 % | Opettajan arviointi arviointitaulukon perusteella |

Nämä ovat esimerkkejä. Opettaja voi jalostaa niitä tai tehdä täysin oman arviointisuunnitelman.

**www.codeschool.fi**

**ISBN:** 978-952-7403-30-3