# SCRATCH-OHJELMOINTI

## Ohjelmoinnin perusteet graafisessa ohjelmointiympäristössä

# Helppoa ja hauskaa!

TIVIKO-hanke

**TIVIKO-hanke on Opetus- ja kulttuuriministeriön avustama hanke.**

# SISÄLLYSLUETTELO

## Mitä ohjelmointi on?

Lyhyesti sanottuna ohjelmointi on täsmällisten toimintaohjeiden antamista tietokoneelle oikeassa järjestyksessä. Toisin kuin ihmiset, tietokone ei itsessään tiedä yhtään mitään. Esimerkiksi se ei tiedä, miten sen tulee toimia käyttäjän painaessa välilyöntiä. Miten ihmeessä tietokone sitten osaa toimia haluamallamme tavalla meidän käyttäessä sitä? Ratkaisu on ohjelma. Tietokoneessa käynnissä oleva ohjelma kertoo sille, miten sen tulee missäkin tilanteessa toimia, esimerkiksi käyttäjän tehdessä jonkin käyttötoimenpiteen. Kaikki tarvittavat toimintaohjeet sijaitsevat ohjelmassa, jonka ohjelmoija on muodostanut ohjelmoimalla. Tietokone on toimintaohjeiden suhteen hyvin tarkka. Se ei osaa tulkita epämääräisiä ohjeita, vaan sille annettavien ohjeiden tulee olla täsmällisiä ja yksikäsitteisiä. Lisäksi ohjeiden tulee olla oikeassa järjestyksessä. Lähes aina ohjelmat kommunikoivat käyttäjien kanssa. Ne ottavat käyttäjiltä vastaan tapahtumia ja tietoja syötteinä, käsittelevät ne ja tulostavat uutta tietoa tai menevät käyttäjän haluamaan tilaan.

Toimintaohjeita sisältäviä ohjelmia tekevät ohjelmoijat. Ohjelmoijan työnkuva on varsin monipuolinen. Ohjelman laadinta aloitetaan suunnittelusta, jossa selvitetään mitä käyttäjän tulee voida ohjelmalla tehdä ja miten tällainen ohjelma voidaan toteuttaa. Huolellisen suunnitelman pohjalta ohjelmoija aloittaa ohjelmoinnin (koodauksen), eli toimintaohjeiden antamisen tietokoneelle. Ohjelmaa tehdessään ohjelmoija testaa ohjelmaa koko ajan ja virheitä havaitessaan korjaa ne.

Ongelmanratkaisutaito on ohjelmoijalle erittäin oleellinen. Jo itse ohjelman tarkoitus on ratkaista jokin käyttäjän ongelma ja varsinainen ohjelmointi pitää sisällään lukuisia pieniä ongelmanratkaisutehtäviä. Ohjelmointia voisi verrata ruuan valmistuksessa käytettävän ohjeen, eli reseptin kirjoittamiseen. Oletuksena on, että tällä ”reseptillä” sellainenkin ihminen, joka ei tiedä ruuan laitosta yhtään mitään kykenee valmistamaan ruuan onnistuneesti.

No miten ohjelmoija sitten antaa nämä toimintaohjeet tietokoneelle? Ohjelmoijan on tunnettava jokin ohjelmointikieli kyetäkseen antamaan toimintaohjeita tietokoneelle. Ohjelmointikieli pitää sisällään mm. komentoja, muuttujia, ehto- ja toistorakenteita, erilaisia operaattoreita ja tapahtumankäsittelijöitä. Ohjelmointikieliä on useita ja niillä jokaisella on oma käyttötarkoituksensa ja syntaksinsa. Yleisesti käytettyjä ohjelmointikieliä ovat mm. C#, Java, Java Script ja PHP. Jos ohjelmoija tekee Internetissä toimivaa ohjelmaa, hän voi ohjelmoida sen esimerkiksi PHP:llä. Tietokoneeseen asennettavan ohjelman hän voisi toteuttaa käyttäen esimerkiksi C#-kieltä.

Tietokone ei suoraan ymmärrä ohjelmointikieltäkään, vaan ennen ohjelman suorittamista se tulee kääntää suoritettavaan muotoon. Jotkin kielet ovat tulkkaavia, jolloin kääntäminen

tapahtuu ohjelmaa suoritettaessa (esim. PHP www-serverillä). Ohjelman lähdekoodin kirjoitus ja kääntäminen suoritetaan yleensä ohjelmointiympäristössä, jossa on monia ohjelmoijan tarvitsemia työkaluja samassa käyttöliittymässä. Aivan kuten tekstinkäsittelyohjelmassa on kaikki kirjoittajan tarvitsemat työkalut. Ohjelmointiympäristö on siis ohjelma, jolla tehdään toisia ohjelmia.

## Mikä Scratch on?

Vuosia sitten ohjelmointi perustui lähes pelkästään kirjoitettuun ohjelmakoodiin ja sen opiskelu oli varsin työlästä. Näin ei ole enää. Nykyään meillä on käytettävissä graafisia ohjelmointiympäristöjä, joissa ohjelmointi perustuu ”vedä ja pudota”-menetelmällä toistensa alapuolelle liitettäviin koodipalikoihin. Tällainen ohjelmointi on hyvin helppoa, nopeaa ja palkitsevaa. Näyttäviä tuloksia saadaan nopeasti aikaan ja onnistumiset innostavat kokeilemaan uusia asioita.

Scratch on suosituin graafinen ohjelmistoympäristö. Se on ilmainen ja keskeisiltä osiltaan suomennettu. Scratch on kehitetty Yhdysvaltalaisessa MIT:n teknisessä yliopistossa juuri lasten ohjelmointiympäristöksi. Scratchissa ei tarvitse kirjoittaa koodia käsin, vaan ohjelmointi perustuu valikosta raahattaviin koodipalikoihin. Ohjelmointi Scratchissa onkin lähes samanlaista kuin Lego-palikoilla leikkiminen. Lapsen opittua käyttämään ympäristöä se on aivan yhtä helppoa, mutta monin verroin monipuolisempaa ja antaa luovuudelle lähes rajoittamattomat mahdollisuudet digitaaliseen askarteluun ja omaan tuottamiseen.

Scratchin aloituskynnys on hyvin matala, ensimmäisen ohjelman saa aikaan hyvin nopeasti pienellä oppimisella. Ja mikä parasta, ohjelmat voivat olla hyvin näyttäviä niin grafiikan kuin ääntenkin osalta. Vaikka Scratch on graafinen ohjelmointiympäristö, se opettaa kuitenkin ohjelmoinnin keskeiset ja yleispätevät perusteet tehokkaasti. Näiden perusteiden avulla lapset oppivat, miten digitaalinen maailma toimii ja he voivat hyödyntää niitä myöhemmin opiskellessaan todellisia ohjelmointikieliä. Ohjelmoidessa Scratchillä lapsi oppii ohjelmoinnillista ajattelua, ongelmien pilkkomista osiin ja ongelmien ratkaisemista algoritmeilla sekä ennen kaikkea toimintaohjeiden antamista tietokoneelle oikeassa järjestyksessä. Ohjelmoidessaan lapset oppivat ajattelemaan luovasti, järjestelmällisesti ja työskentelemään yhteisöllisesti toistensa kanssa. Scratchillä lapsi voi luoda ohjelmien ja pelien lisäksi myös interaktiivisia tarinoita ja animaatioita.

Scratchistä on olemassa kaksi versiota. Toista versiota käytetään palveluna www-selaimella ja toinen on tietokoneeseen asennettava ohjelma. Tässä materiaalissa käytetään www-selaimessa toimivaa palvelua, sillä se on helppo ottaa käyttöön missä ympäristössä tahansa. Www-version ohjelmistovaatimukset ovat:

- Www-selain: Chrome 7, Firefox 4, Internet Expolorer 8 (tai uudempi versio)
- Adobe Flash Player 10.2 asennettuna (tai uudempi versio)
- Näytön resoluutio 1024 x 768 tai suurempi

Useimmissa tietokoneissa nämä vaatimukset täyttyvät. Tarvittaessa voit asentaa www-selaimesta ja Adobe Flash Playeristä uusimmat versiot. Jos ongelmia edelleen ilmenee, voit ladata Scratchin sivustolta tietokoneeseen asennettavan ohjelman, asentaa sen ja käyttää sitä.

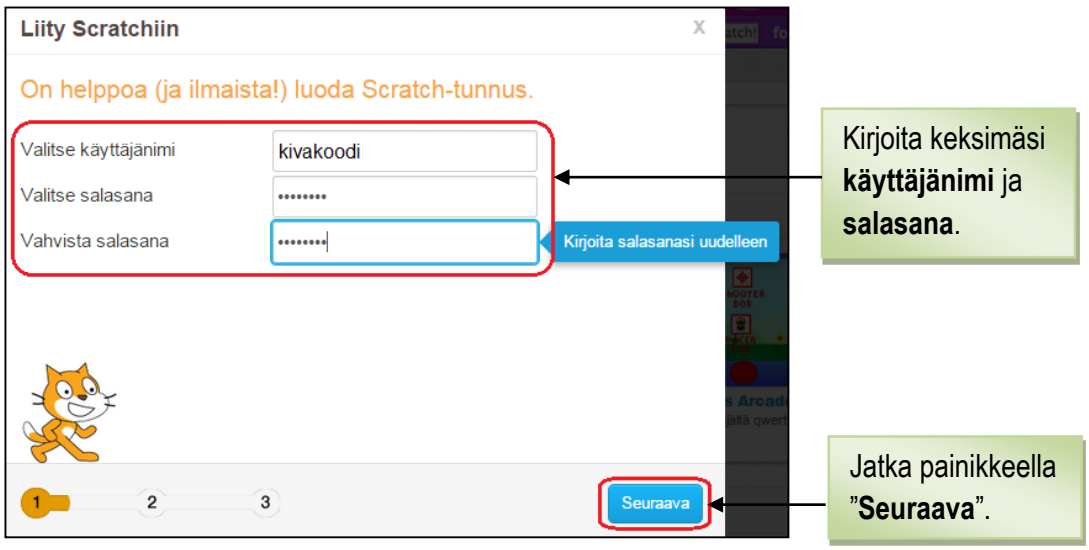
# SCRATCHIN KÄYTÖN ALOITUS

## Rekisteröityminen Scratchiin

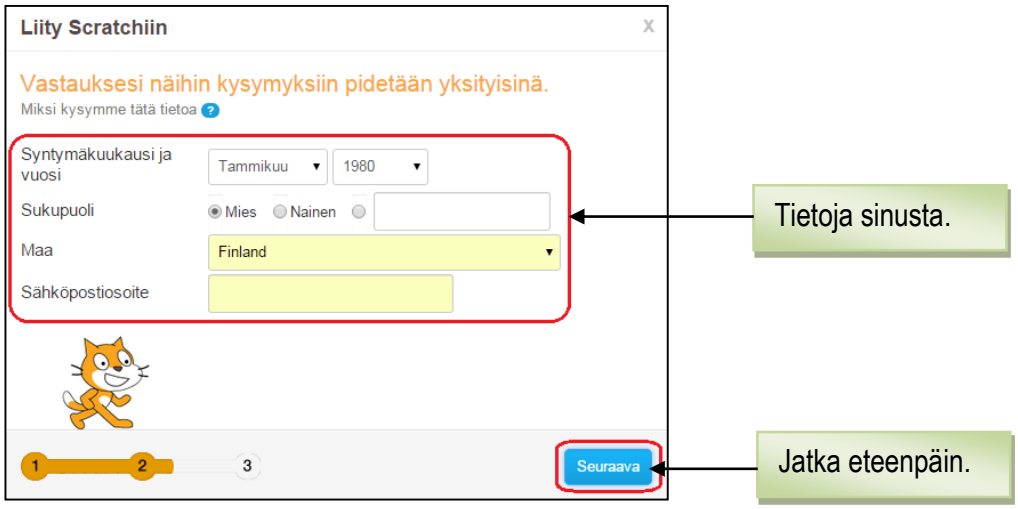
Heti aluksi Sinun kannattaa rekisteröityä Scratchiin. Sen jälkeen voit kätevästi tallentaa laatimasi ohjelmat pilvipalveluun ja käyttää niitä miltä koneelta tahansa. Halutessasi voit myös jakaa tekemäsi ohjelmat muiden käyttäjien kanssa.

1. Käynnistä www-selain ja siirry osoitteeseen http://scratch.mit.edu
2. Klikkaa selaimeen aukeavan sivun yläosasta tekstiä ”**Liity Scratchiin**”.

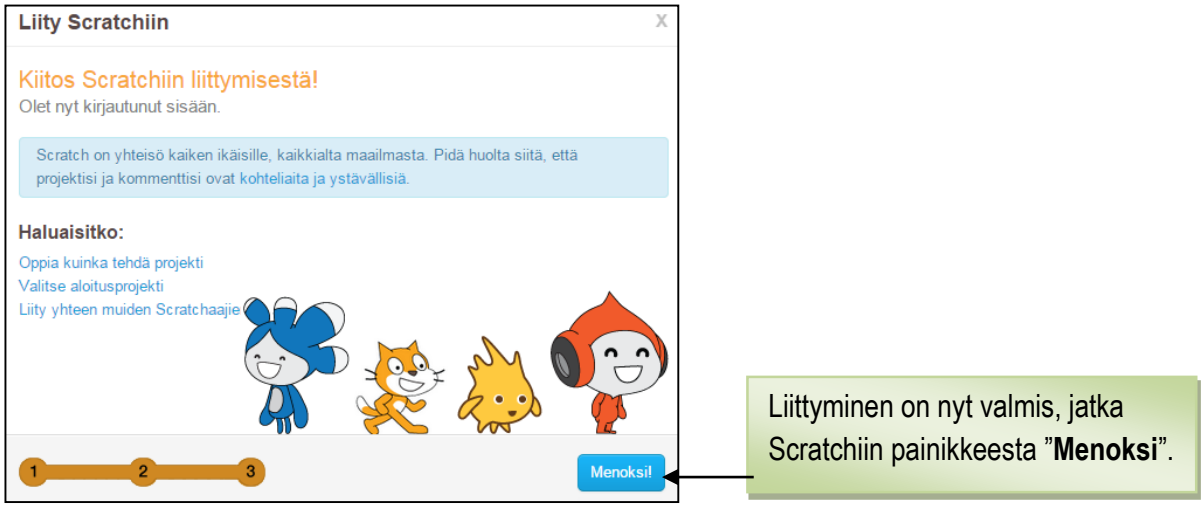

3. Valitse seuraavaksi itsellesi **käyttäjänimi** ja **salasana**. Tarvitset näitä aina kirjautuessasi Scratchiin, joten valitse sellaiset jotka muistat. Käyttäjänimi näkyy toisille käyttäjille, esimerkiksi jakaessasi tekemiäsi ohjelmia. Älä siis käytä oikeaa nimeäsi, vaan keksi itsellesi käyttäjänimi. **Kerro myös oppilaillesi, että he antavat käyttäjänimeksi keksityn nimen, ei oikeaa nimeä**. Jos syöttämäsi käyttäjänimi on jo varattu, saat siitä ilmoituksen. Tiedot syötettyäsi klikkaa painiketta ”**Seuraava**”.



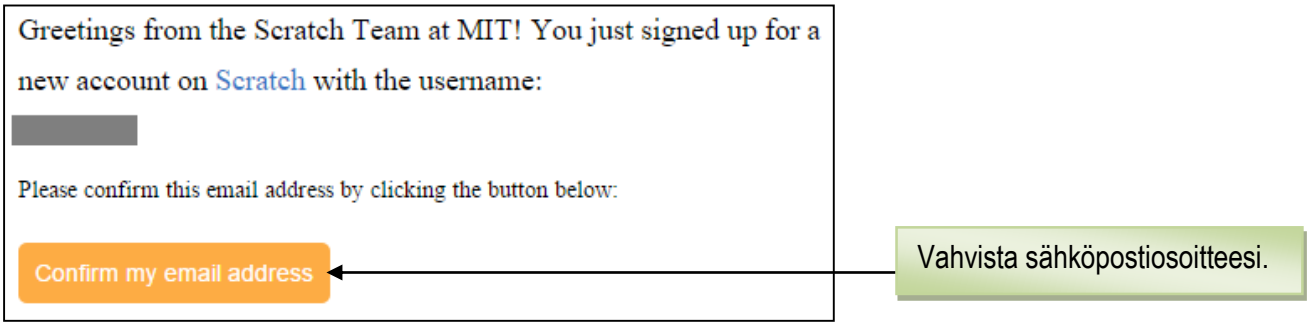
4. Seuraavaan ruutuun syötät tietoja itsestäsi: **Syntymäkuukauden** ja **vuoden**, **sukupuolen**, **maan** ja **sähköpostiosoitteesi**. Sähköpostiosoitteesi on tarpeellinen, jos esimerkiksi unohdat salasanasi, syötä siis oikea ja toimiva osoite. Tiedot syötettyäsi klikkaa painiketta ”**Seuraava**”.




5. Nyt olet liittynyt Scratchiin. Jatka klikkaamalla painiketta ”**Menoksi**”.




6. Rekisteröitymisen lopuksi käy katsomassa sähköpostisi ja varmenna osoite klikkaamalla viestissä olevaa painiketta ”Confirm my email address”.

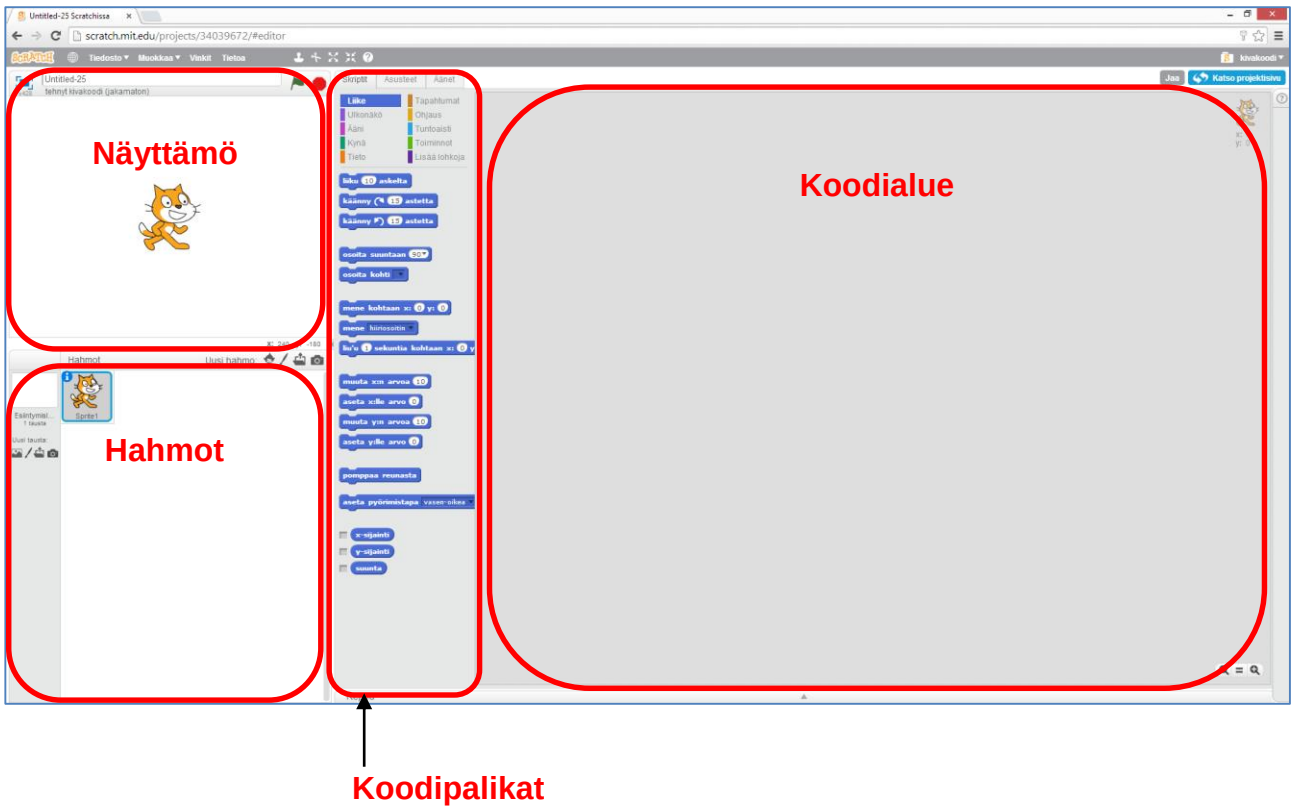


# SCRATCHIN OHJELMOINTIYMPÄRISTÖÖN TUTUSTUMINEN

## OHJELMOINTIYMPÄRISTÖN KÄYTTÖLIITTYMÄ

Liittymisen jälkeen ohjelmointiympäristöön pääset klikkaamalla sinisen palkin tekstiä **Luo**.

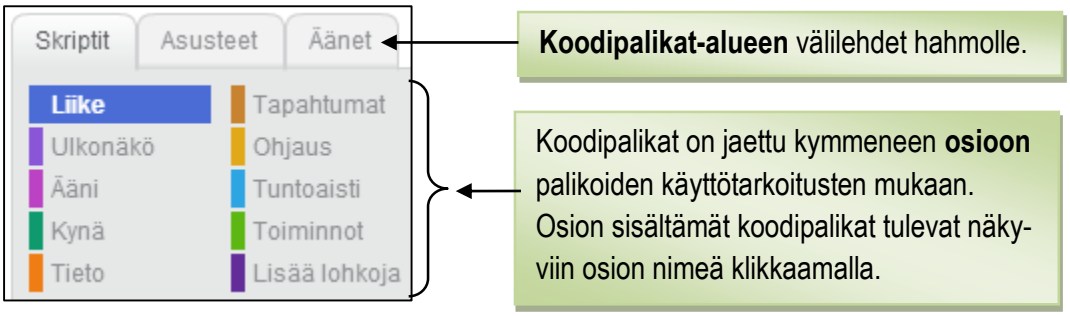

Pääset alla olevan kuvan mukaiseen näkymään, joka on Scratchin ohjelmointiympäristö. Ympäristössä on neljä keskeistä aluetta: **Näyttämö**, **Koodipalikat**, **Koodialue** ja **Hahmot-ikkuna**.



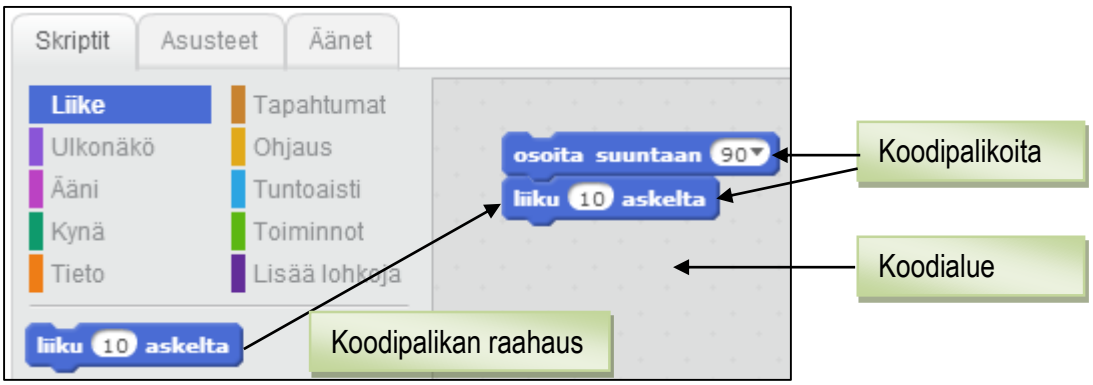

**Näyttämöllä** tapahtuu kaikki ohjelmaan ohjelmoitu toiminta. Animaatioissa hahmot liikkuvat ja taustat vaihtuvat itsestään, peleissä puolestaan pelaajan suorittamat toiminnot saavat nämä aikaan. Näyttämön voi myös suurentaa koko kuvaruudun kokoiseksi, mikä onkin mielekästä silloin, kun ohjelma suoritetaan.

**Koodipalikat-alue** on jaettu välilehdillä kolmeen osaan. Hahmolle on käytettävissä välilehdet Skriptit, Asusteet ja Äänet. Taustoille puolestaan välilehdet Skriptit, Taustat ja Äänet. Tässä vaiheessa ainoastaan Skriptit-välilehti on sinulle oleellinen.

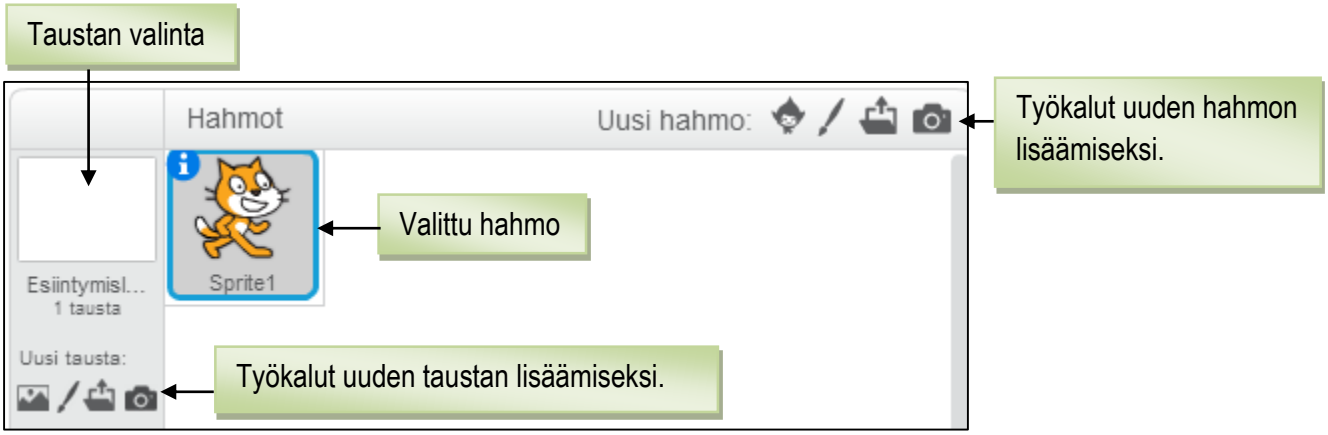
**Skriptit-välilehteä** tarvitset jatkuvasti, sillä sieltä löytyvät kaikki ohjelman rakentamisessa tarvittavat koodipalikat. Asusteet-välilehdeltä näet hahmon asusteet, voit muokata niitä ja muodostaa uusia asusteita. Äänet-välilehdellä voit lisätä hahmolle uusia ääniä joko kirjastosta tai itse nauhoittamalla mikrofonin kautta. Taustalle on käytettävissä vielä Taustat-välilehti, jossa voit muodostaa näyttämölle uusia taustoja joko lataamalla ne kirjastosta, tiedostosta tai piirtämällä itse.




**Koodialueelle** muodostetaan ohjelman koodi raahaamalla sinne koodipalikoita allekkain, kuten alla olevassa kuvassa on tehty. Koodipalikat raahataan koodialueelle Skriptit-välilehdeltä.



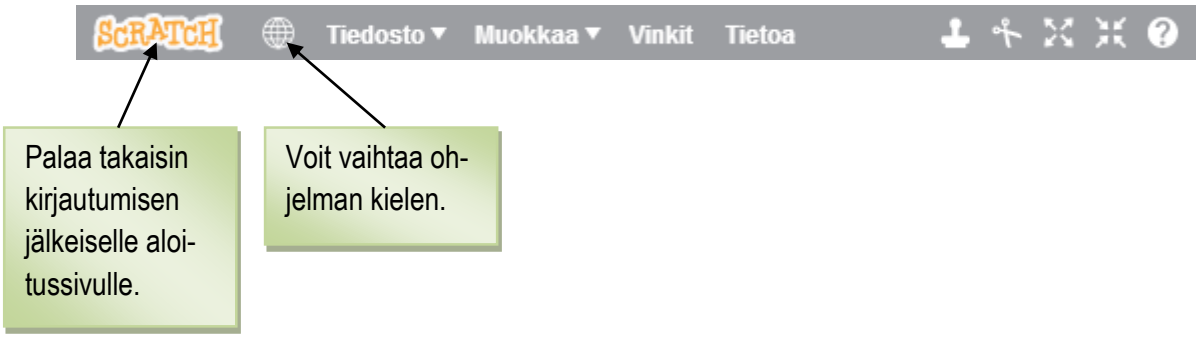
**Hahmot-ikkunassa** näkyvät kaikki ohjelmaan lisätyt hahmot. Jokaiseen uuteen projektiin muodostuu oletuksena kissa-hahmo. Hahmot-ikkunan työkaluilla voit lisätä projektiin uusia hahmoja joko tiedostoista, kirjastosta tai itse piirtämällä. Samoilla tavoilla voit lisätä myös taustakuvia. **Hahmot-ikkunassa valitaan myös kohde, tietty hahmo tai tausta, jolle ohjelmakoodi muodostetaan**. Valitun hahmon kuvakkeen ympärillä on sininen kehys.



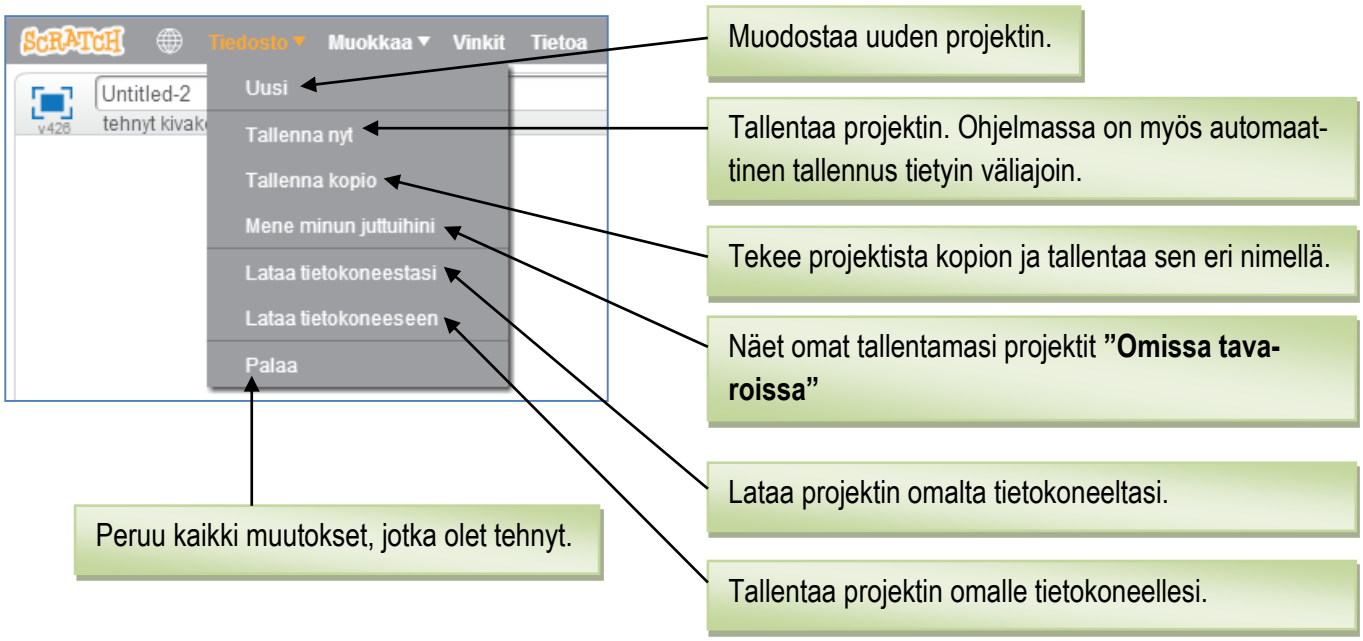
## VALIKKORIVI JA VALIKOT

Alla olevissa kuvissa on esitetty valikkorivi toimintoineen ja valikoineen.

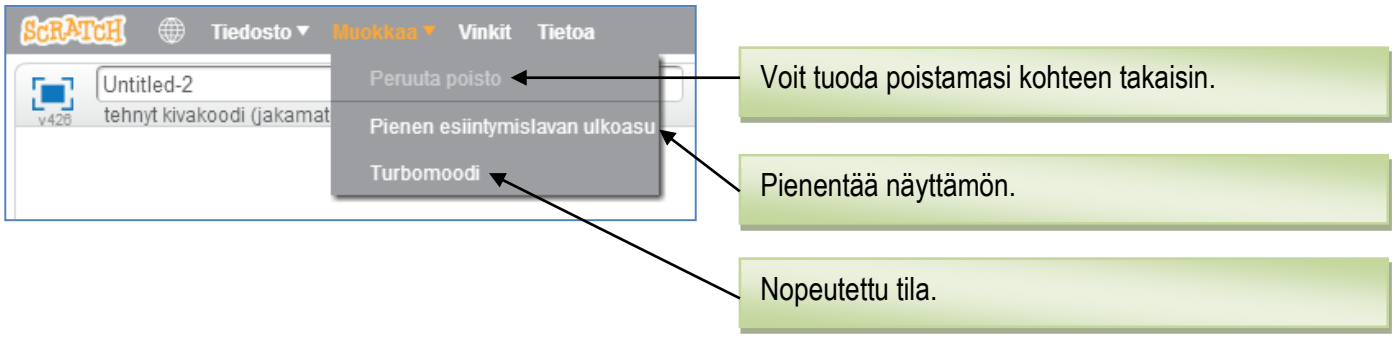
**Valikkorivi**:



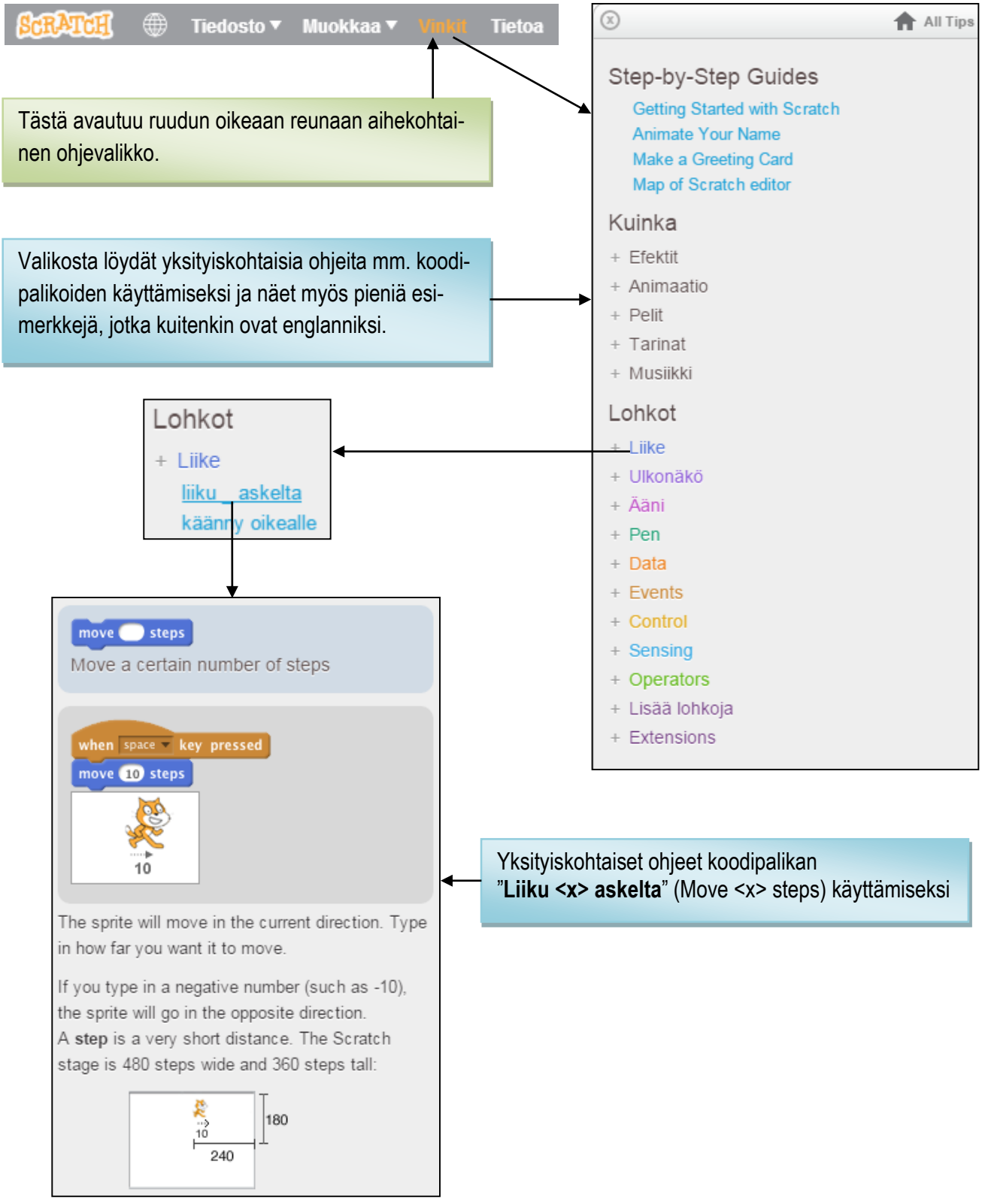

**Tiedosto**-valikko:




**Muokkaa**-valikko:

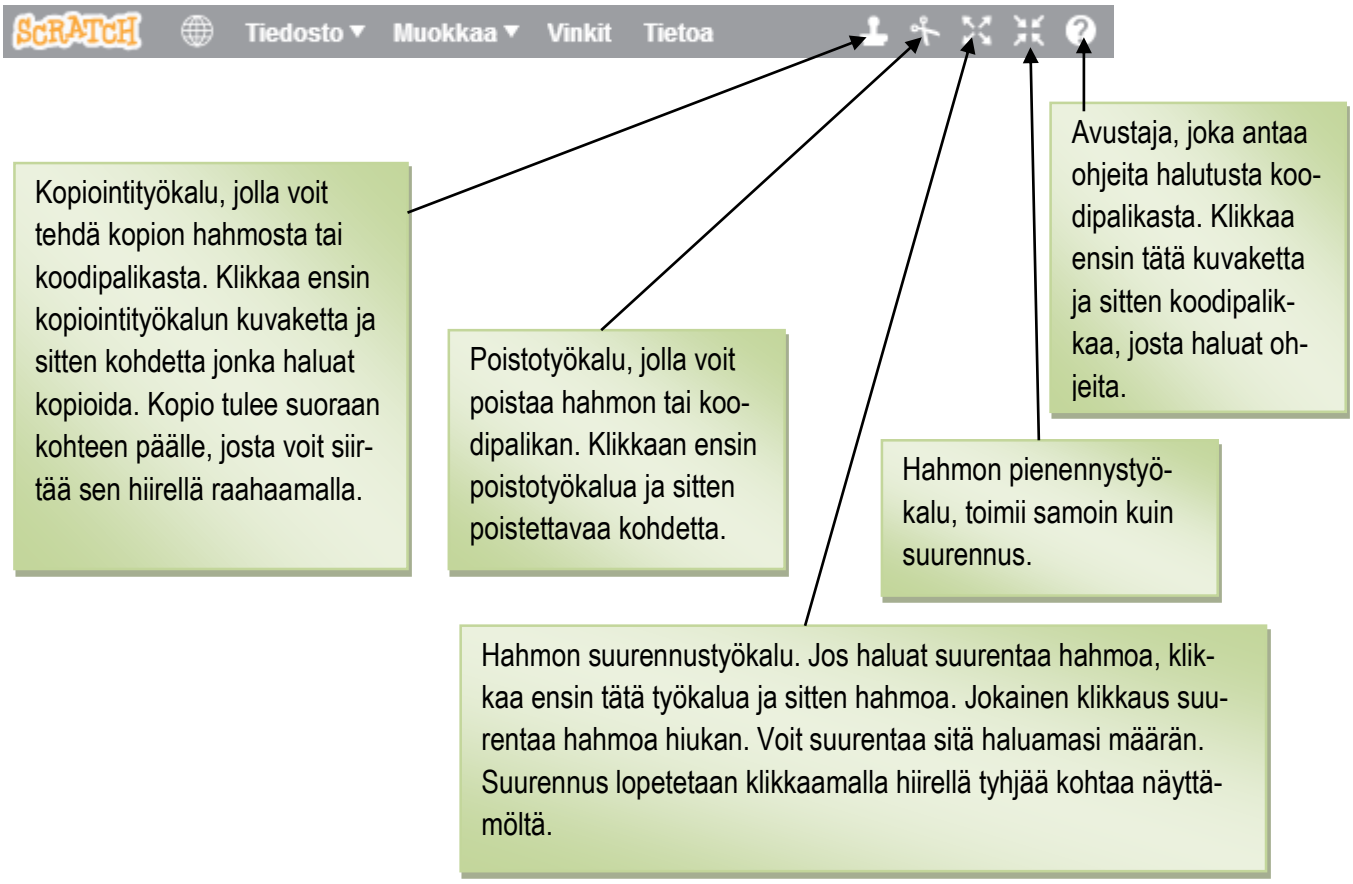


**Vinkit**-valikko:

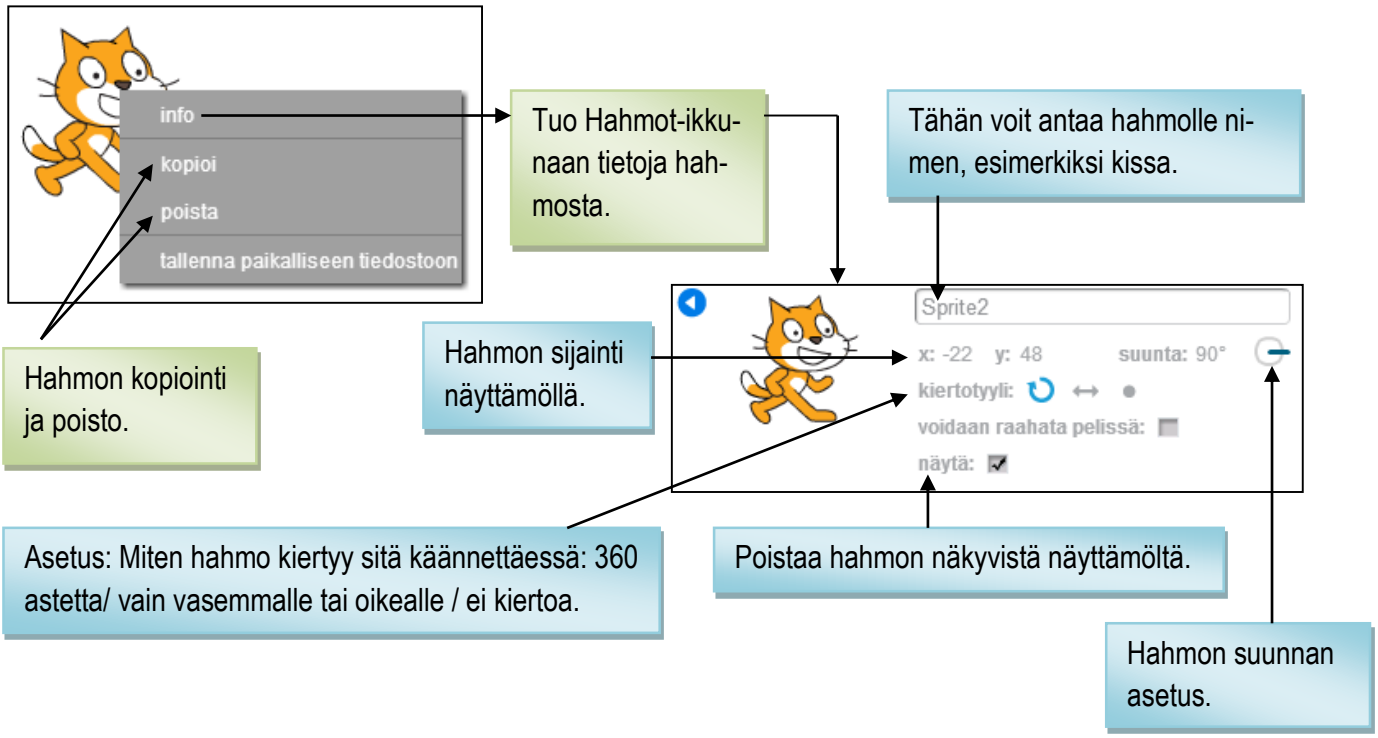


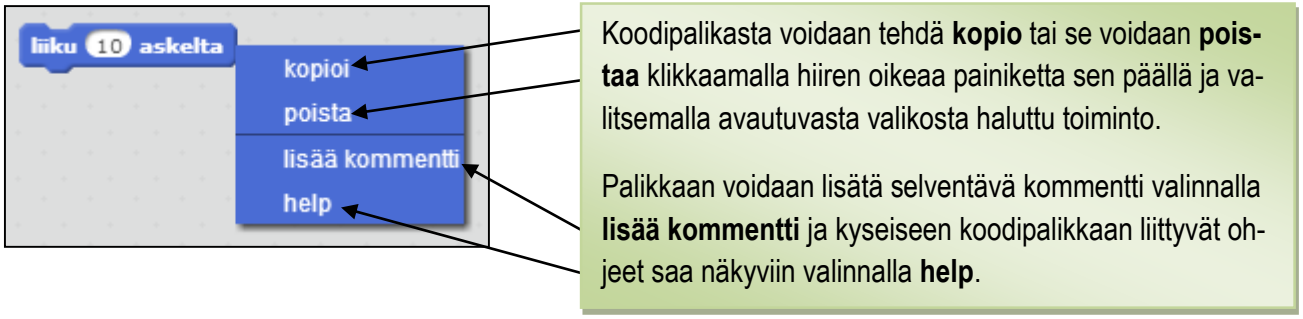
Valikko **Tietoa** vie sinut Scratchin tukisivustolle.

Valikkorivin **pikakuvakkeet**:

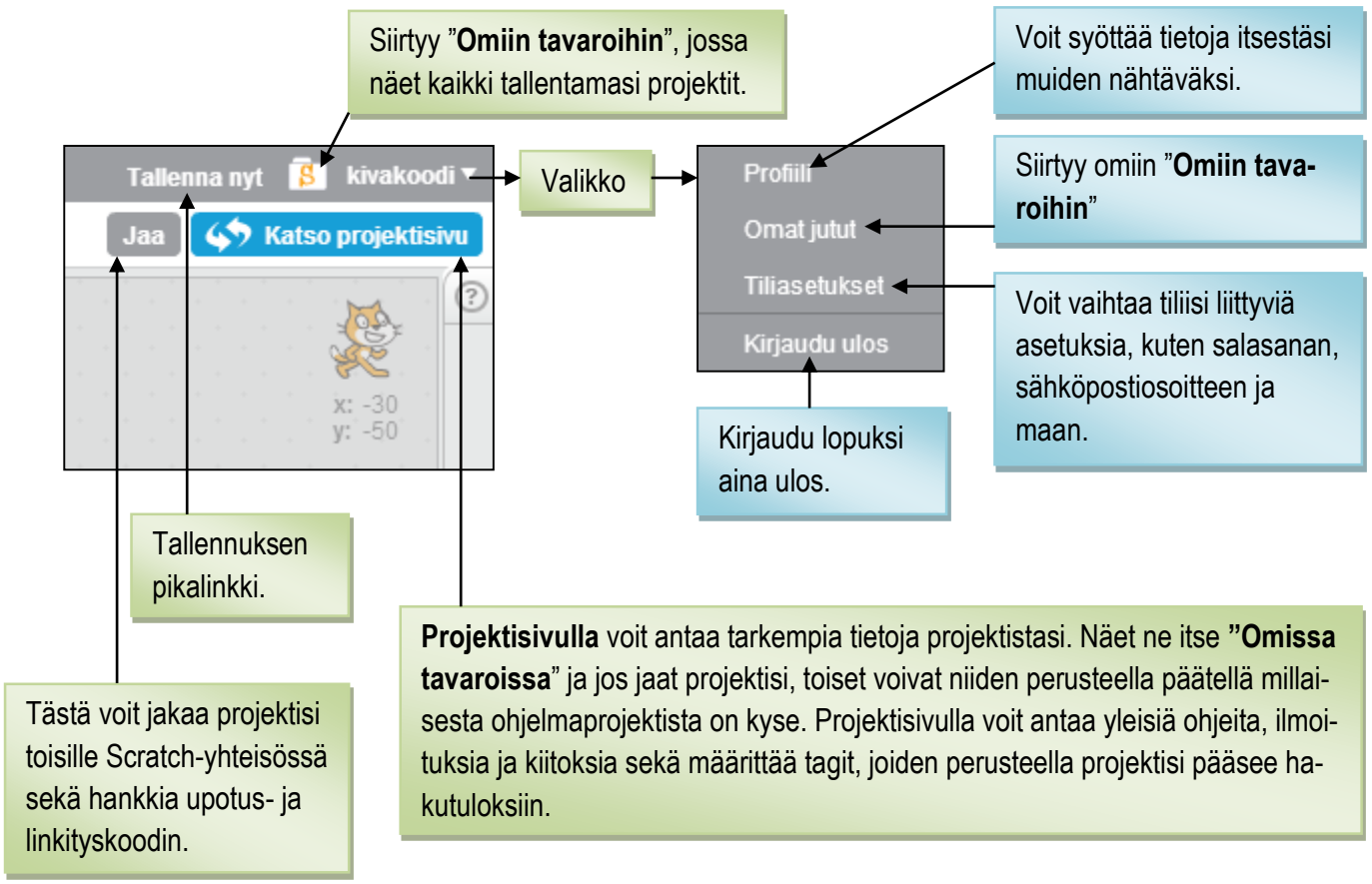



Hahmosta ja koodipalikasta voi tehdä kopion myös klikkaamalla niiden päällä hiiren oikeaa painiketta ja valitsemalla avautuvasta valikosta valinta **Kopioi**. Samassa valikossa on myös valinta **Poista**, jolla kyseinen kohde voidaan poistaa.



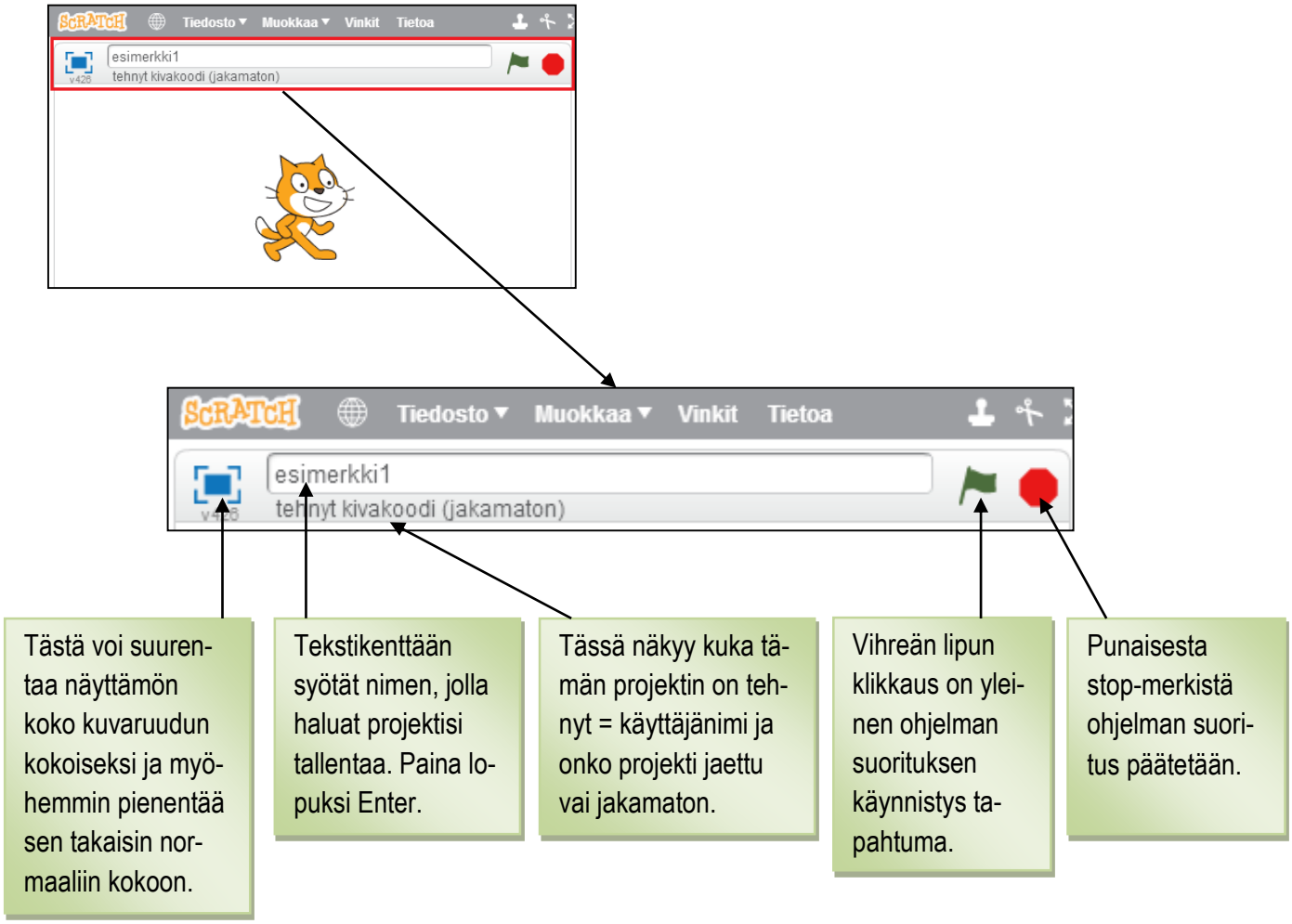



Näyttömön oikeassa reunassa valikkorivillä ja sen alapuolella on myös toimintoja. Ne on esitetty ao. kuvassa.



## Näyttämö

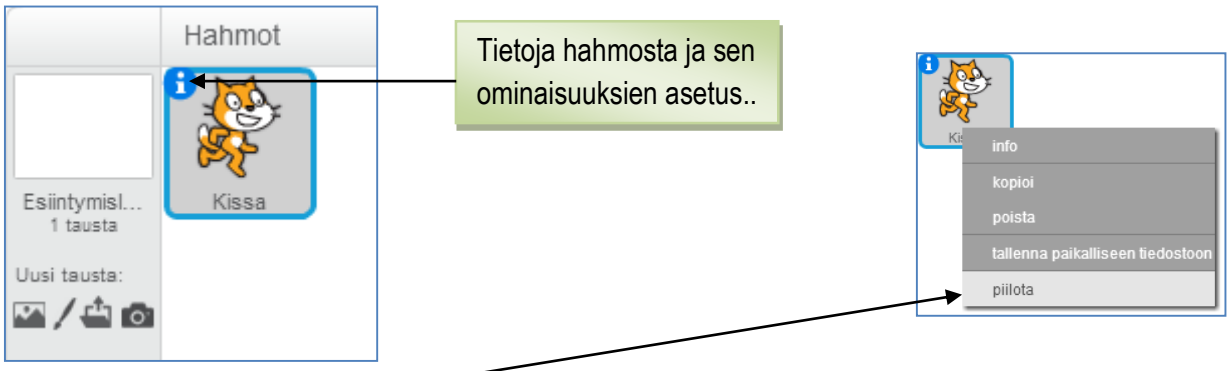
Kuten aiemmin jo todettiin, kaikki ohjelman suorituksen aikaiseen toimintaan liittyvät asiat esitetään näyttämöllä. Esimerkiksi jos olet tehnyt animaation, näet sen näyttämöllä. Näyttämön yläosassa on neljä tärkeää toimintoa, jotka käymme seuraavaksi läpi.




Ohjelman hahmot ja taustat esitetään näyttämöllä. Usein ohjelmaan kuuluu useita hahmoja, eivätkä kaikki ole välttämättä näkyvissä samanaikaisesti. Näyttämö on kaksiulotteinen taso. Hahmojen paikat näyttämöllä määräytyvät niiden x- ja y-koordinaattien perusteella. Kun liikutat hiiriosoitinta näyttämöllä, näet **hahmot-ikkunan** oikeassa alareunassa osoittimen koordinaatit, esim. x: -220, y: 170. Näyttämön **vasemman yläkulman** koordinaatit ovat **x:-240, y: 180** ja oikean **alakulman koordinaatit** ovat puolestaan **x: 240, y: -180**. Näyttämön keskipisteen koordinaatit ovat **x: 0, y: 0**. Ohjelmaa tehdessäsi voit ottaa näyttämöllä olevasta hahmosta kiinni hiiren vasemmalla painikkeella ja raahata sen haluamaasi kohtaan näyttämöä.

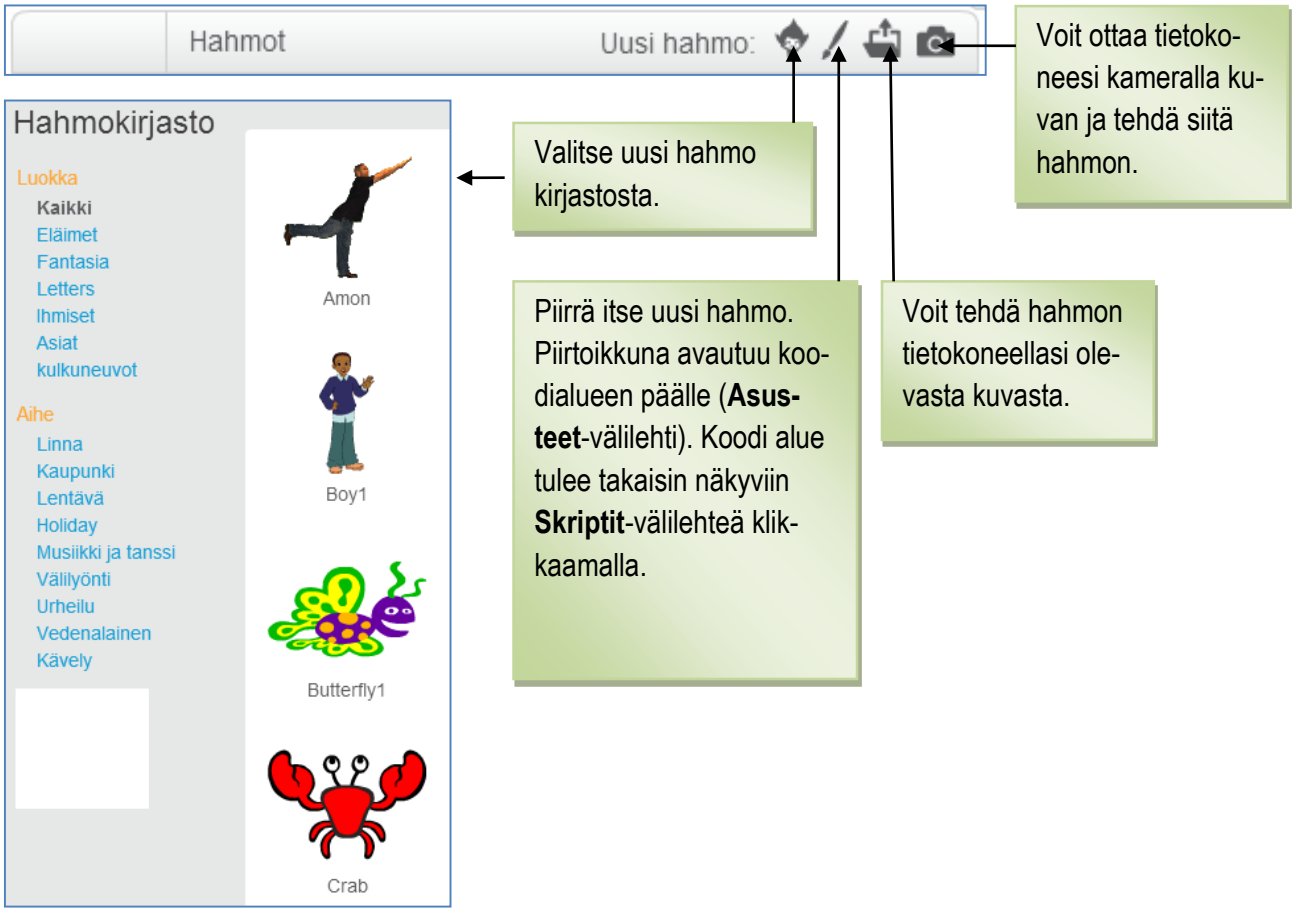
## HAHMOT-IKKUNA

**Hahmot-ikkuna** sijaitsee näyttämön alapuolella ja siinä suoritetaan hahmojen ja taustojen hallinta. Jos haluat lisätietoja hahmosta tai muuttaa joitakin sen asetuksia, voit klikata hahmon ympärillä olevassa sinisessä kehyksessä sijaitsevaa i-kirjaimen kuvaketta.

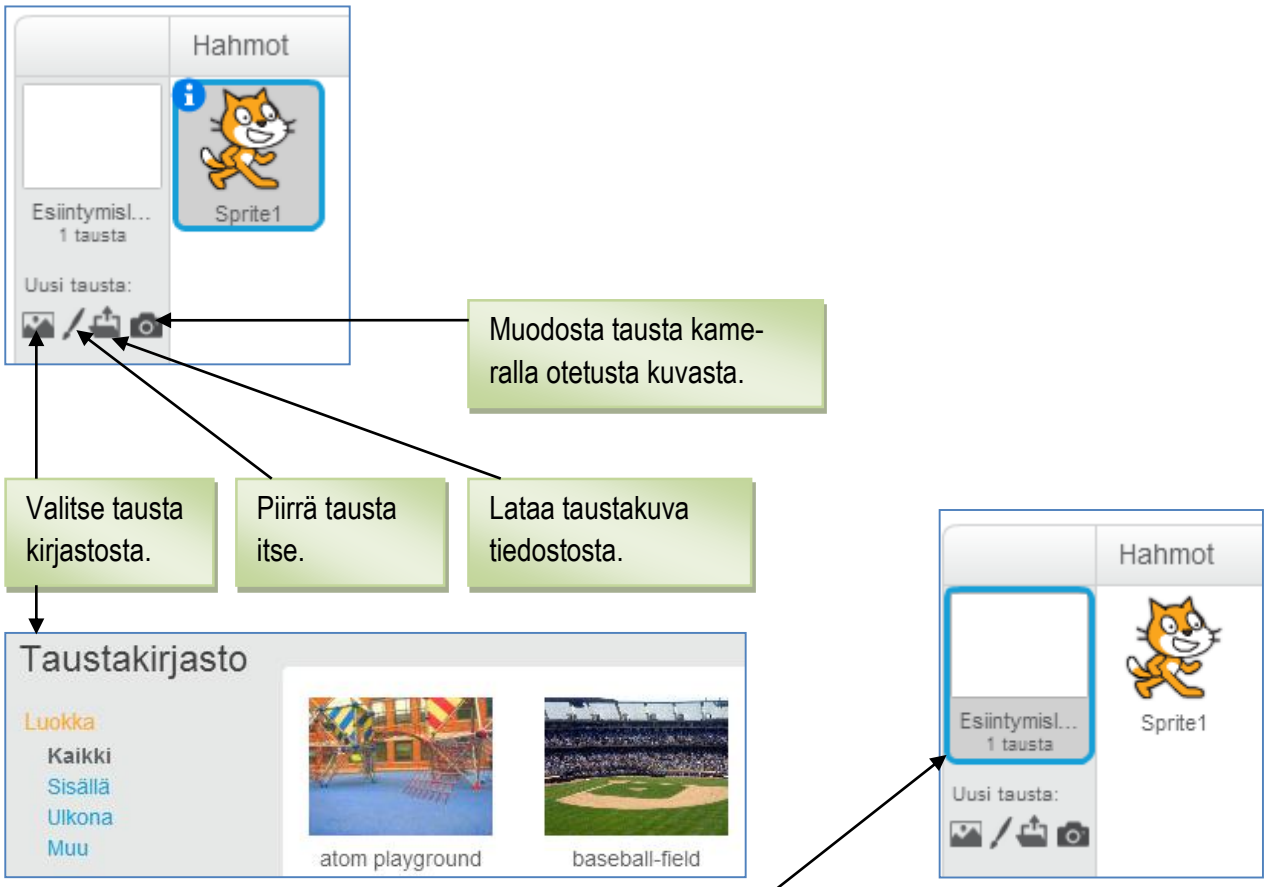



Jos haluat piilottaa hahmon, klikkaa sen kuvaketta Hahmot-ikkunassa ja valitse valikosta valinta **Piilota**.

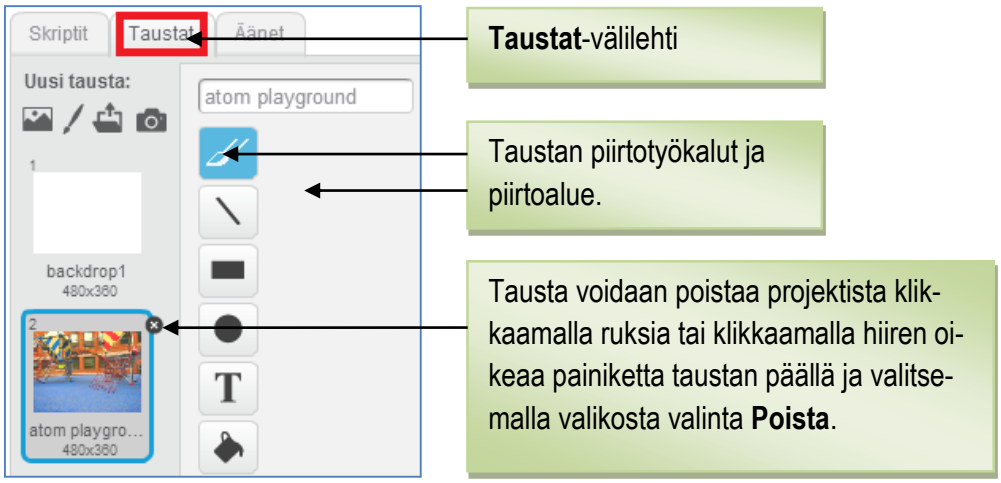
Hahmot-ikkunan oikeassa yläreunassa on kuvakkeet, joilla voit tuoda projektiisi uusia hahmoja. Voit valita uuden hahmon kirjastosta, piirtää sen itse, ladata tiedostosta tai muodostaa se ottamalla kuva tietokoneen kameralla.



Oletuksena näyttämön tausta on valkoinen. Taustan voi kuitenkin vaihtaa haluamakseen Hahmot-ikkunan vasemmasta reunasta. Tausta voidaan valita valmiista kirjastosta, piirtää itse, ladata tiedostosta tai muodostaa tietokoneen kameralla otetusta kuvasta. Aivan kuten hahmojakin, myös taustoja voidaan vaihtaa ohjelman suorituksen aikana.



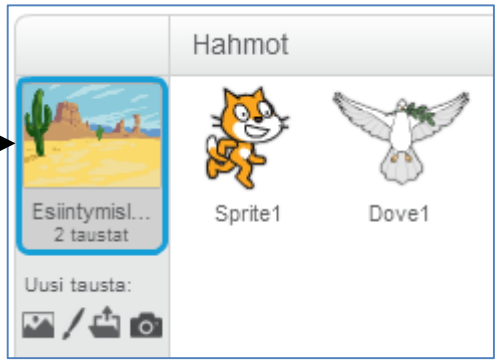

Kun Hahmot-ikkunassa näyttämön tausta on valittuna, voit klikata koodipalikat alueelta **Taustat**-välilehteä. Koodialue muuttuu taustanpiirtotyökaluksi, jossa voit piirtää haluamasi taustan. Taustakirjastosta tai kuvasta muodostettua taustaa on mahdollista muokata piirtämällä sen päälle.

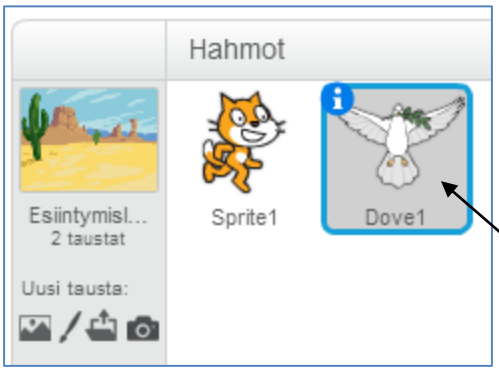


# YLEISKATSAUS KOODIPALIKOIHIN

Koodipalikat ovat tärkeä osa Scratchiä, sillä varsinainen toimintojen ohjelmointi suoritetaan niillä. Jokaisella hahmolla on oma itsenäinen, toisista hamoista ja taustasta riippumaton koodi, joka on muodostettu koodipalikoista. Taustoilla oleva koodi on puolestaan yhteinen kaikille taustoille. Hahmojen ja taustan koodit pystyvät kuitenkin kommunikoimaan keskenään, kuten myöhemmin tulet huomaamaan. Esimerkiksi pelien ohjelmoinnissa hahmojen välinen kommunikointi on tärkeä osa toiminnallisuutta.

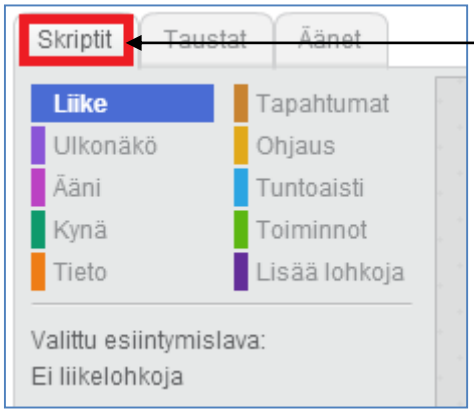
Kun Hahmot-ikkunassa on **tausta** valittuna, voit muodostaa koodipalikoista ohjelmakoodin taustalle. Tällöin koodialueella näkyy taustan koodi. **Huomaa** että taustalle ei ole olemassa kaikkia samoja koodipalikoita kuin hahmoille ja että koodi on yhteinen kaikille taustoille.




Kun valitset hahmon Hahmot-ikkunasta, voit muodostaa koodin kyseiselle hahmolle. Jokaisella hahmolla on oma itsenäinen ja toisista hahmoista riippumaton koodinsa.

Aloitetaan tutustuminen koodipalikoihin, jos ne eivät ole näkyvissä, klikkaa **koodipalikat-alueelta** välilehteä **Skriptit**.




Sekä hahmojen että taustan **koodipalikat** löytyvät aina **Skriptit-välilehdeltä**.

Koodipalikat on jaettu kymmeneen aihekohtaiseen osioon niiden käytön helpottamiseksi ja nopeuttamiseksi.

1. **Liike-osio**: Hahmon liikuttamiseen tarvittavat palikat.
2. **Ulkonäkö-osio**: Hahmojen ulkonäköön liittyvät palikat, myös puhe- ja ajatuskuplat.
3. **Ääni-osio**: Äänen tuottamiseksi tarvittavat palikat.
4. **Kynä-osio**: Palikat, joilla hahmot voivat piirtää viivoja ja jättää leimoja liikkuessaan näyttämöllä.
5. **Tieto-osio**: Muuttujien ja taulukoiden muodostus sekä niiden käsittely.
6. **Tapahtumat-osio**: Tapahtumat, joihin ohjelman halutaan reagoivan.
7. **Ohjaus-osio**: Odotus, toisto- ja ehtorakenteet.
8. **Tuntoaisti-osio**: Totuusarvo- ja ominaisuuspalikat, jotka kertovat mm. koskettaako hahmo jotain tai onko jotain ”tuntoaistein” havaittavaa tapahtunut (esim. liikettä kameran kuvassa).
9. **Toiminnot-osio**: Matemaattiset operaattorit, satunnaisluku, vertailu- ja loogiset operaattorit sekä muuttujien ja merkkijonojen yhdistäminen funktiopalikoilla.
10. **Lisää lohkoja -osio**: Voit tehdä omia aliohjelmia eli palikoita.

## KOODIPALIKOIDEN TYYPIT

Scratchsissä on neljä eri tyyppistä koodipalikkaa, jotka eroavat toisistaan niiden käyttötarkoituksen ja ulkonäön suhteen. Kaikista palikoista voidaan käyttää yleisnimeä **koodipalikat**. Alla olevassa luettelossa on kuitenkin kuvattu koodipalikat tarkemmin niiden tyyppien mukaisesti.

1. **Komentopalikka**
   Komentopalikan suoritus aiheuttaa jonkin toimenpiteen, joka yleensä myös voidaan havaita. Esimerkiksi komentopalikka ”**liiku <x> askelta**” liikuttaa hahmoa ja komentopalikka ”**sano <teksti> <x> sekunnin ajan**” tuottaa puhekuplan. Komentopalikoissa on usein mahdollisuus käyttää parametreja, jotka määrittävät, miten komento suoritetaan. Esimerkiksi palikassa ”**liiku <x> askelta**” **<x>** on parametri, joka määrittää kuinka monta askelta hahmon tulee liikkua. Palikassa ”**sano <teksti> <x> sekunnin ajan**” parametri **<teksti>** määrittää puhekuplaan tulevan tekstin ja parametri **<x>** puolestaan sen, miten kauan puhekupla on näkyvissä.

2. **Ominaisuus- / muuttuja- / funtiopalikka**

Nämä kolme palikkaa ovat ulkomuodoiltaan täysin samanlaisia. Ominaisuuspalikat voivat olla eri värisiä riippuen siitä, minkä osion palikasta on kysymys. Muuttujapalikat ovat aina oranssin värisiä. Funktiopalikat puolestaan ovat aina vihreitä.

Ominaisuuspalikasta voidaan lukea tietyn ominaisuuden arvo. Esimerkiksi palikasta ”**x-sijainti**” voidaan lukea hahmon x-koordinaatin arvo ja palikasta ”**<vuosi> nyt**” voidaan lukea nykyinen vuosiluku. Muuttujapalikasta voidaan lukea muuttujan arvo ja komentopalikkaa käyttämällä siihen voidaan tallentaa uusi arvo.

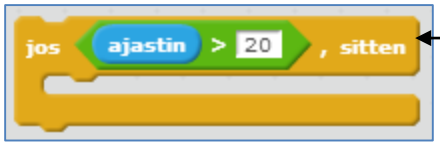

Ominaisuuspalikoita


Muuttujapalikka muuttujalle nimeltä **laskuri**.


Funktiopalikka. Käytettävä matemaattinen funktio valitaan valintalistasta ja funktiolle välitettävä luku annetaan parametrina. Palikka palauttaa funktion tuloksen.

Yhteistä näillä kaikille tämän ryhmän palikoille on se, että niitä ei koskaan käytetä yksinään, vaan aina jonkin muun koodipalikan yhteydessä. Esimerkiksi ehto- ja toistorakenteen ehdossa voidaan tutkia, onko ominaisuuden tai muuttujan arvo yhtä suuri, pienempi tai suurempi kuin tietty raja-arvo.


Ominaisuuspalikkaa ”**<vuosi> nyt**” käytetään komentopalikan ”**sano <teksti> <x> sekunnin ajan**” parametrina.




Ehtorakenteen ehdossa käytetään ominaisuuspalikkaa **ajastin** ja vertailuoperaattoria ”**suurempi kuin**” tutkimaan, onko ominaisuuden **ajastin** arvo suurempi kuin **20**. Jos **ajastimen** arvo on **suurempi kuin** 20, ehdon totuusarvo on **tosi** ja silloin suoritetaan ehtorakenteen **sitten-osa**.


Toistorakenteen ehdossa käytetään muuttujapalikkaa **laskuri** ja vertailuoperaattoria ”**yhtä suuri kuin**” tutkimaan, onko muuttujan **laskuri** arvo yhtä suuri kuin 20. Toistorakenteen toisto päättyy, kun ehto muuttuu arvoon **tosi**, eli muuttujan **laskuri** arvo on **20**.


Komentopalikalla ”**aseta <laskuri> arvoon <1>**” asetetaan muuttujan **laskuri** arvoksi luku **1**.

3. **Totuusarvopalikka**

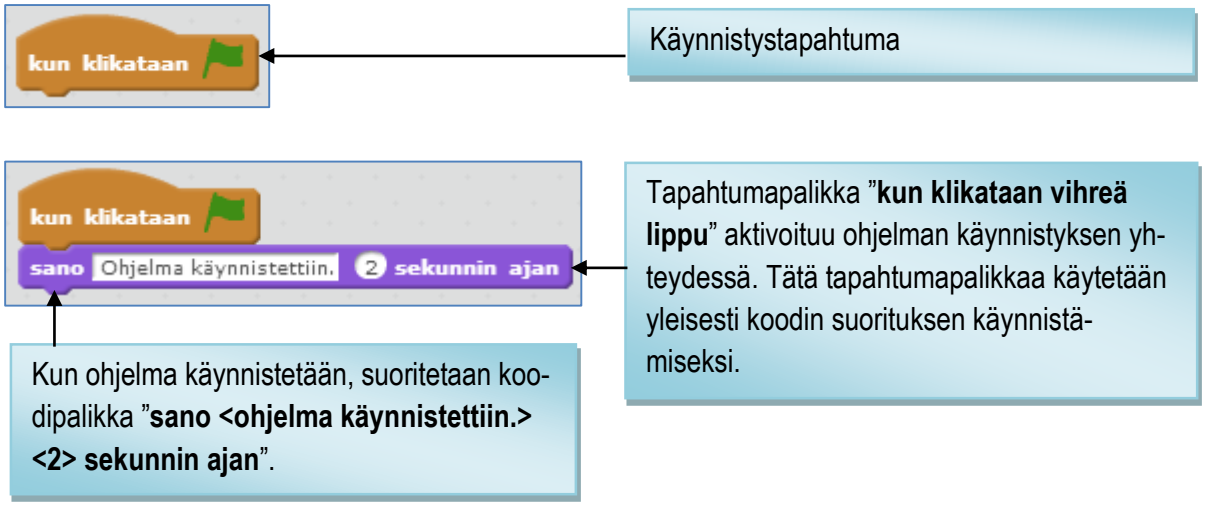
Kaikki totuusarvopalikat palauttavat totuusarvon joka on joko **tosi** tai **epätosi**. Totuusarvopalikat eivät voi esiintyä yksinään, vaan niitä käytetään ehto- ja toistorakenteiden ehdoissa. Esimerkiksi totuusarvopalikka ”**koskettaako <reuna>**” palauttaa arvon **tosi**, jos hahmo koskettaa näyttämön reunaa. Jos hahmo ei kosketa reunaa, se palauttaa arvon **epätosi**.


Totuusarvopalikka ”**koskettaako <reuna>**”. Kosketuksen **kohde** voidaan valita valintalistasta. Muita mahdollisia kosketuksen kohteita ovat hiiriosoitin ja toinen hahmo.


Ehtorakenteen ehdossa tutkitaan, koskettaako hahmo reunaa. Ehto on **tosi**, jos hahmo koskettaa reunaa ja silloin suoritetaan ehtorakenteen **sitten-osaan** sijoitetut koodipalikat.

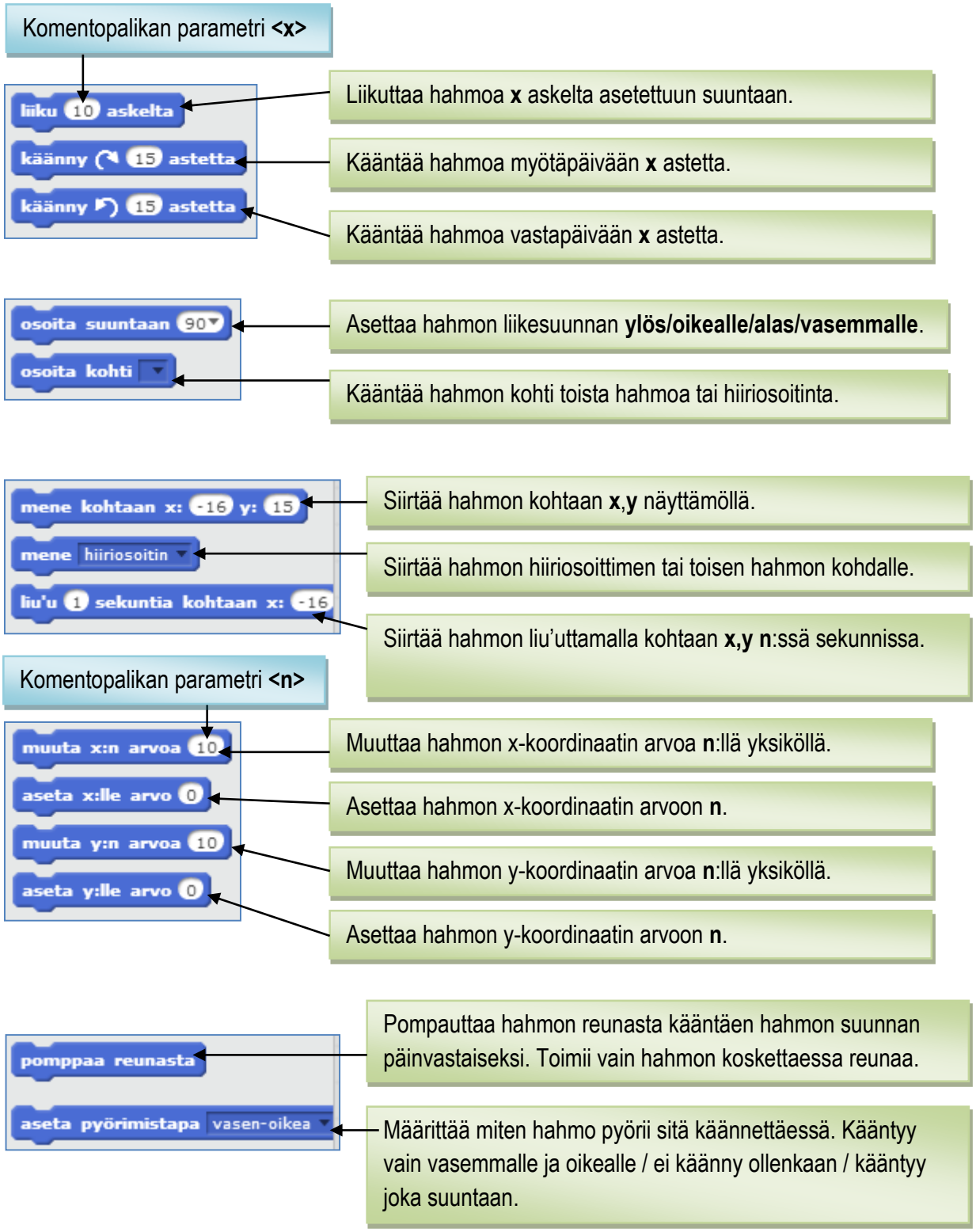
4. **Tapahtumapalikka**

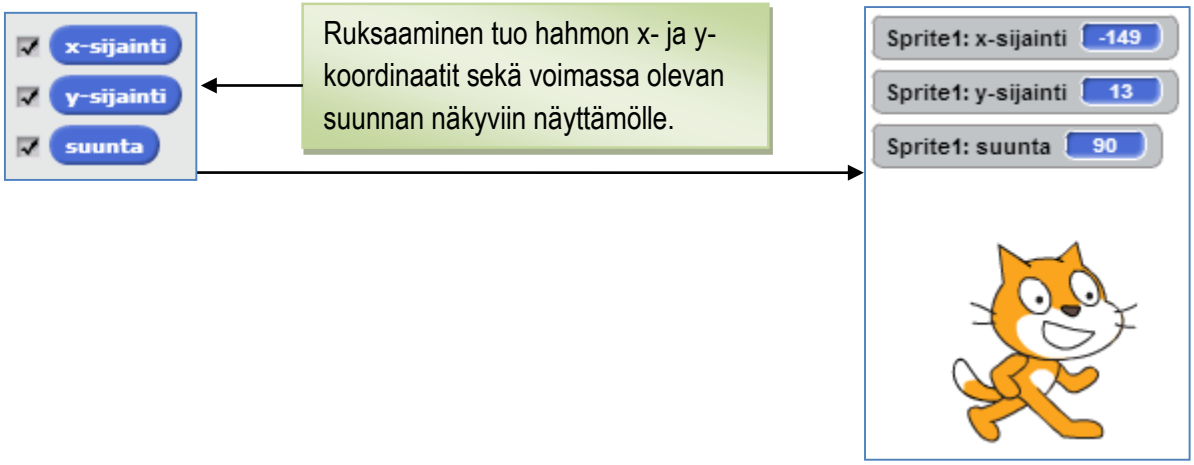
   Tapahtumapalikat reagoivat erilaisiin tapahtumiin, kuten vihreän lipun klikkaamiseen, näppäimen painamiseen, hahmon klikkaamisen, viestin vastaanottoon tai ajastimen arvoon. Kaikki tapahtumapalikat löytyvät **Tapahtumat-osiosta** ja niillä on hyvin tärkeä merkitys. **Kaikki ohjelmat käynnistetään jollain tapahtumalla**. Koodin suorituksen käynnistäminen on aina tarpeen sitoa kiinni johonkin tapahtumaan. Hyvin yleinen käynnistystapahtuma on ”**kun klikataan vihreä lippu**”. Tämä tapahtuma aktivoituu silloin, kun ohjelma käynnistetään virherän lipun kuvaketta klikkaamalla. Tällöin suoritetaan kaikki tämän tapahtumapalikan alle sijoitetut koodipalikat.



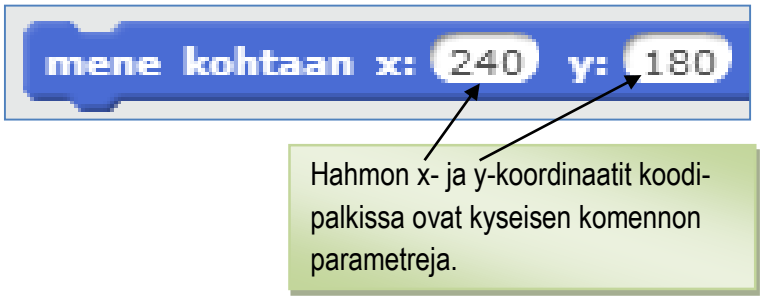
## Liike-osio

Liike-osiosta löytyvillä komentopalikoilla aikaansaadaan kaikki näyttämöllä olevien hahmojen liike. Jotkut palikat siirtävät hahmoa välittömästi ja toiset puolestaan muuttavat hahmon liikesuuntaa.



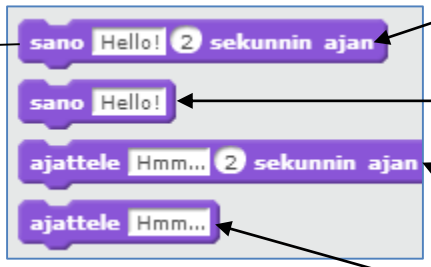



Voit kokeilla yksittäisen koodipalikan suorituksen vaikutusta hahmoon tuplaklikkaamalla haluamaasi palikkaa. Jos palikassa on asetettavia parametreja, voit asettaa ne ennen klikkausta. Voit kokeilla esimerkiksi klikata palikkaa ”**Liiku 10 askelta**” ja näet kissan liikahtavan näyttämöllä. Etsi seuraavaksi koodipalikka ”**Mene kohtaan x: <koordinaatti> y: <koordinaatti>**”, anna x:n arvoksi 240 ja y:n arvoksi 180. Arvot annetaan klikkaamalla sinisessä koodipalkissa olevia valkoisia kohtia. Tuplaklikkaa tämän jälkeen koodipalikkaa ja näet kissan siirtyvän aivan näyttämön oikeaan yläreunaan.



## Ulkonäkö-osio

Näillä koodipalikoilla muutetaan hahmojen ja taustan ulkonäköä sekä ohjelmoidaan hahmo esittämään puhe- ja ajatuskuplia.




Hahmo ”sanoo” puhekuplalla parametrina annetun **tekstin**. Teksti on näkyvissä **x** sekuntia.

Sama kuin edellä, mutta puhekupla jää näkyviin.

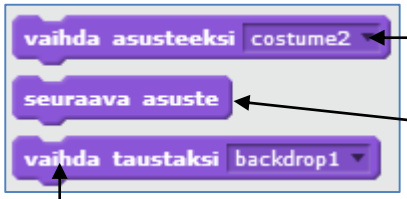
Hahmo ”ajattelee” ajatuskuplalla parametrina annetun **tekstin**. Teksti on näkyvissä **x** sekuntia.

Sama kuin edellä, mutta ajatus jää näkyviin.


Komentopalikan vaikutus kissahahmoon.


Tuo hahmon näkyviin näyttämölle, jos se on ollut piilotettuna.

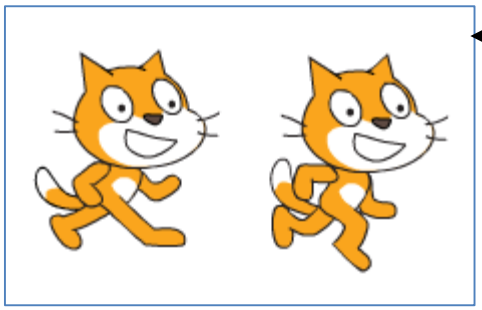
Piilottaminen poistaa hahmon näkyvistä näyttämöltä.




Useimmilla piirroshahmoilla on yleensä kaksi asustetta ja niitä voidaan luoda itse lisää. Tällä komennolla vaihdetaan hahmolle tietty asuste.

Vaihtaa hahmolle seuraavan asusteen ”asustelistasta”. Asusteita vaihtamalla ja hahmoa samalla liikuttamalla luodaan esimerkiksi kävelyä tai lentämistä esittävä animaatio.

Vaihtaa näyttämön taustaksi halutun taustan tai edellisen / seuraavan taustan. Haluttu tausta tulee ensin olla lisättynä projektiin (**Taustat/uusi tausta**).



Kuvassa on kissan kaksi asustetta. Kissan hahmo on niissä eri asennoissa ja näiden asusteiden vaihto liikkumisen yhteydessä muodostaa kävelyanimaation.


Muuttaa hahmoa erilaisilla tehosteilla määrällä **x**. Mahdollisia tehosteita ovat väri / kalansilmä / pyörre / pikselöi / mosaiikki / kirkkaus / haamu.

Asettaa jonkin em. tehosteen arvoon **x**.

Poistaa kaikki em. graafiset tehosteet ja palauttaa hahmon perustilaan.


Muuttaa hahmon kokoa määrällä **x**. Positiivinen luku kasvattaa hahmoa ja negatiivinen luku pienentää sitä.

Asettaa hahmon koon arvoon **x** prosenttia. 100% on nykyinen koko, alle 100% pienentää ja yli 100% suurentaa hahmoa.

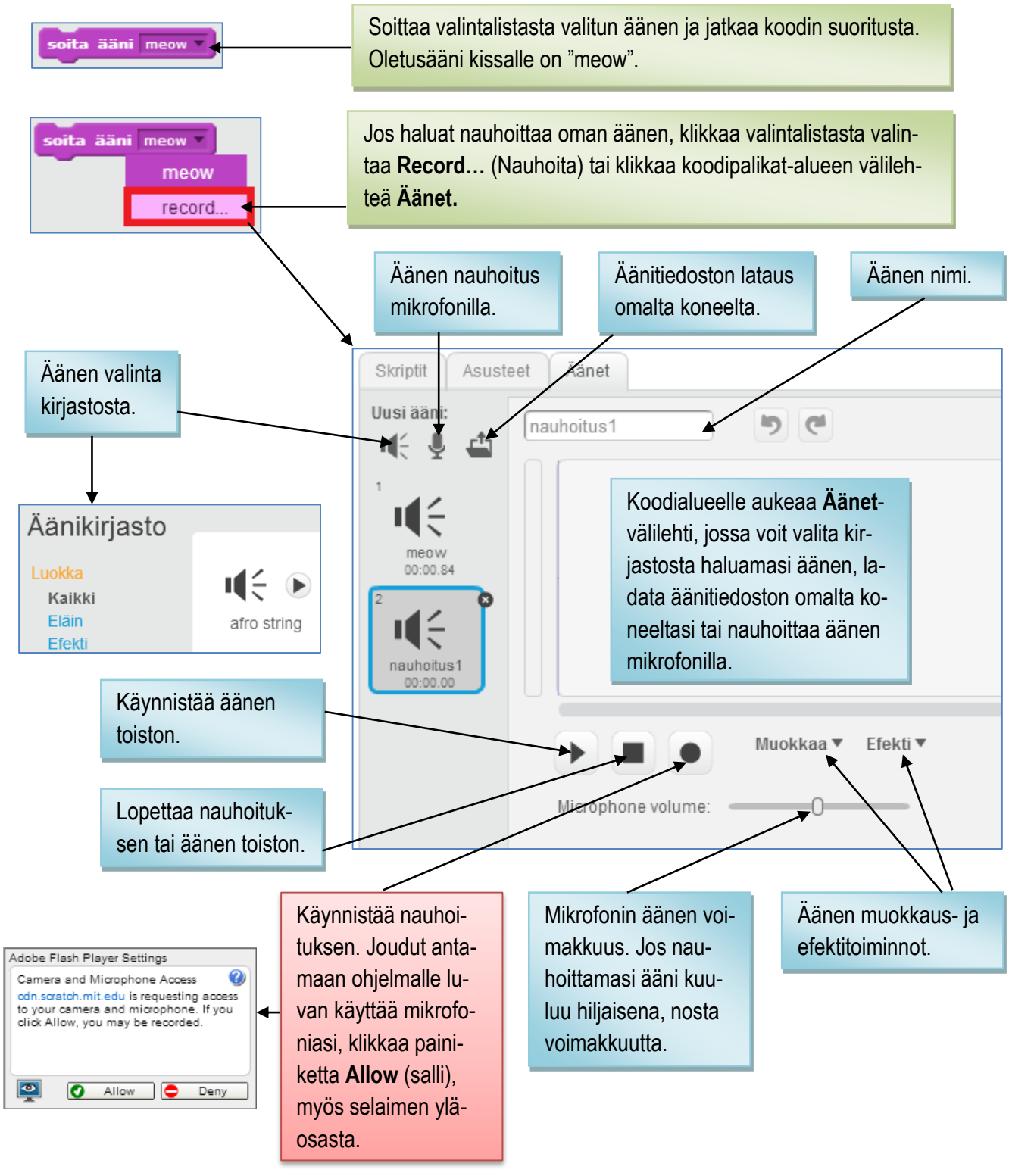

Siirtää hahmon kaikkien muiden hahmojen päälle, ko. hahmo näkyy siis päällimmäisenä.

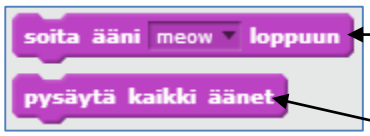
Siirtää hahmoa taaksepäin suhteessa muihin hahmoihin. Tällöin hahmo jää toisten hahmojen taakse.


Valintojen ruksaaminen tuo näyttämölle näkyviin hahmon asusteen numeron, taustan nimen ja hahmon koon.

## ÄÄNI-OSIO

Äänien tuottamiseen on kaksi erilaista mahdollisuutta. Yksinkertaisin menetelmä on soittaa koodipalikalla tietty äänitiedosto. Esimerkiksi hahmoilla on valmiina yksi hahmolle sopiva äänitiedosto. Voit myös ladata äänitiedoston omalta koneeltasi tai tallentaa sen suoraan mikrofonin kautta. Toinen menetelmä äänien tuottamiseksi on soittaa jokin nuotti valitulla instrumentilla.



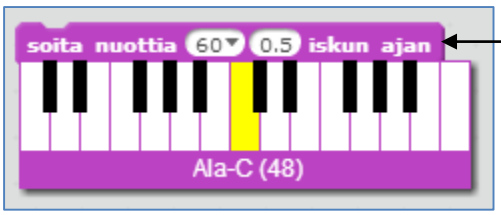



Soittaa valitun äänen **mutta** odottaa äänen loppumista, ennen kuin jatkaa koodin suoritusta. Muutoin sama kuin komentopalikka "**soita ääni <x>**".

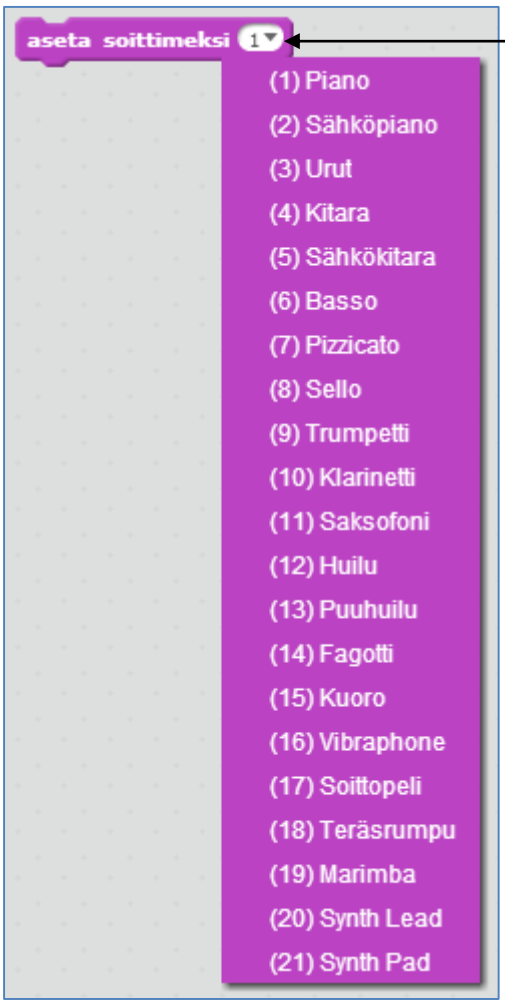
Pysäyttää kaikki käynnissä olevat äänet.


Soittaa valittua rumpua yhden kerran. Odottaa soittamisen jälkeen **<x> iskun ajan** ennen kuin koodin suoritusta jatketaan. Iskunaika riippuu asetetusta temposta.

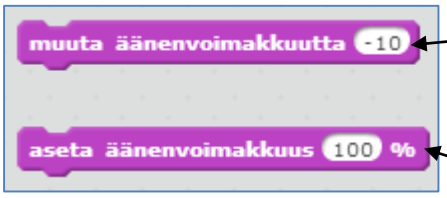

Tauko **<x>** iskua. Tauon pituus riippuu temposta. Esim. jos tempo on 60 bpm niin 1 isku = 1 sekunti.




Soittaa valintalistasta valittua **nuottia <x> <n>** iskun ajan. Odottaa iskun ajan ennen kuin jatkaa koodin suoritusta.

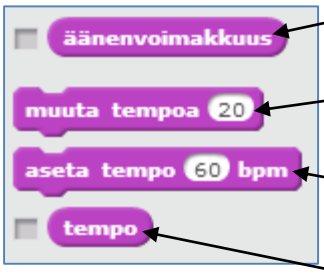



Valintalistasta valitaan **soitin**, jolla edellisen komennon nuotti soitetaan.




Muuttaa hahmon äänen voimakkuutta **<x> yksikköä** alueella 0-100. Positiivinen luku kasvattaa voimakkuutta ja negatiivinen alentaa sitä.

Muuttaa hahmon äänen voimakkuuden arvoon **<x> prosenttia** (0-100).




Ruksaaminen tuo hahmon äänen voimakkuuden näkyviin näyttämölle.

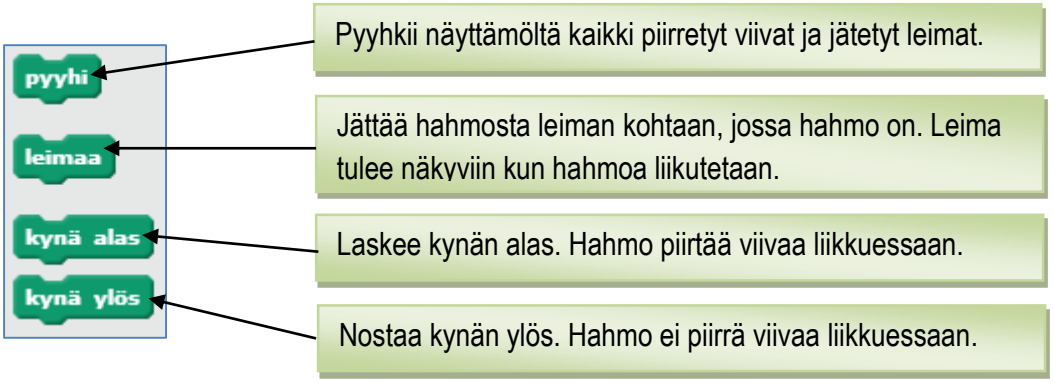
Muuttaa tempoa **<x> bpm**, positiivinen tai negatiivinen.

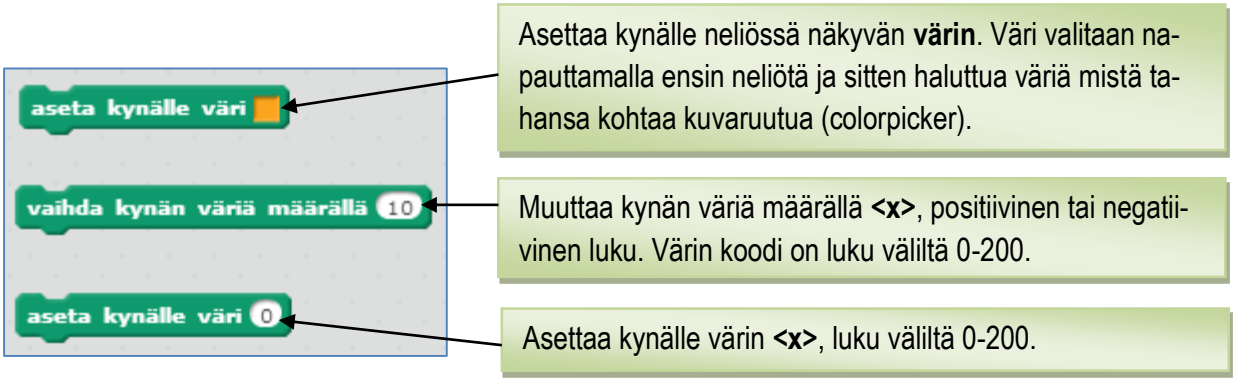
Asettaa tempon arvoon **<x> bpm**.

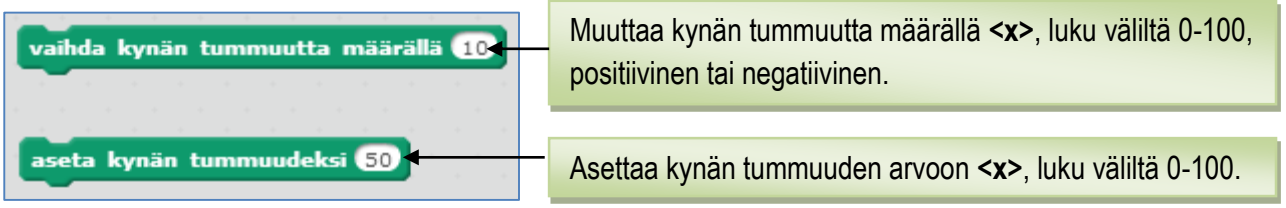
Ruksaaminen tuo hahmon tempon näkyviin näyttämölle.

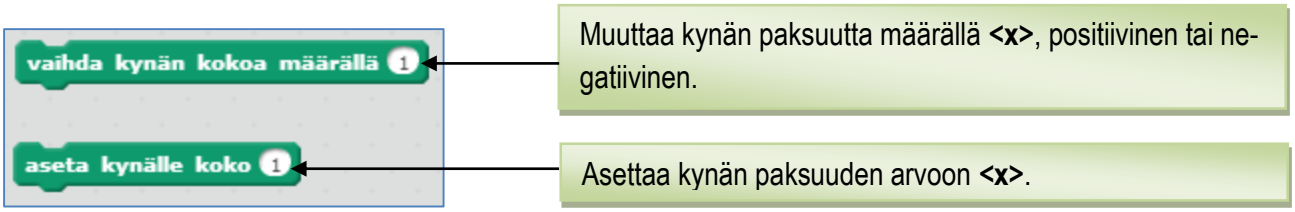
## Kynä-osio

Näillä koodipalikoilla näyttämöllä liikkuvat hahmot voivat piirtää viivaa perässään tai jättää leimoja, eli kopioita hahmosta.



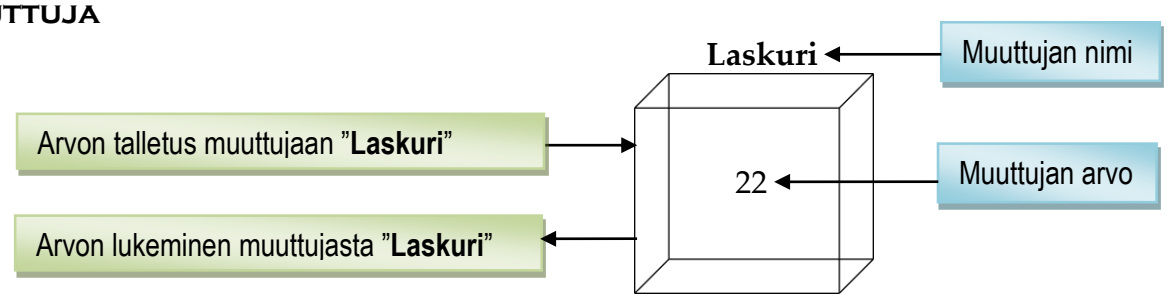



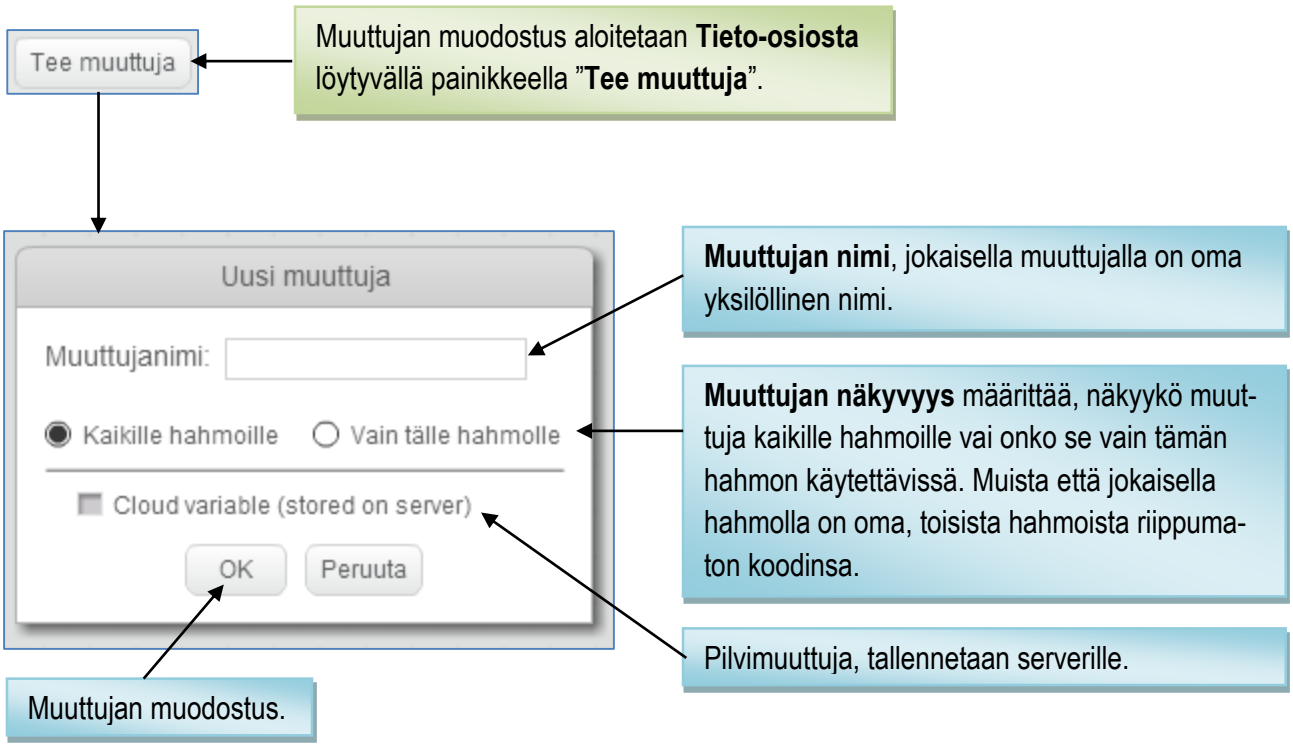
## Tieto-osio

Hyvin usein hiemankin laajemmissa ohjelmissa tarvitaan muuttujia. Muuttujia käytetään ohjelman tarvitsemien tietojen varastointiin ja monesti ne esiintyvät ehto- ja toistorakenteiden ehtojen yhteydessä. Esimerkiksi ehtorakenteen ehdossa voidaan tutkia, onko muuttujalla tietty arvo ja tehdä toimenpiteitä sen perusteella. Toistorakenteissa puolestaan toistojen lukumäärä voidaan määrittää muuttujalla. Ehto- ja toistorakenteisiin palataan myöhemmin.

Muuttuja voidaan ajatella laatikoksi, jolla on nimi. Tähän laatikkoon voidaan tallettaa vain yksi asia, jokin arvo. **Jokaisella muuttujalla on oma yksilöivä nimi, kahta saman nimistä muuttujaa ei voi olla olemassa**. Muuttujalle voidaan antaa arvo ja muuttujasta voidaan lukea siihen tallennettu arvo useita kertoja ohjelman suorituksen aikana.

### Muuttuja



Alla olevissa kuvissa on luotu muuttuja nimeltä ”**laskuri**” ja käsitellään sitä komentopalikoilla.


Valinnan ruksaaminen tuo muuttujan nimen ja arvon näkyviin näyttämölle ja ruksin poistaminen piilottaa sen. Muuttuja voidaan poistaa klikkaamalla hiiren oikeaa painiketta muuttujan nimen päällä (oranssi alue) ja valitsemalla valikosta valinta **Poista muuttuja**. Samasta valikosta on mahdollista myös nimetä muuttuja uudelleen valinnalla **Uudelleennimeä muuttuja**.


Asettaa valintalistasta valitun muuttujan **<laskuri>** arvoon **<0>.**

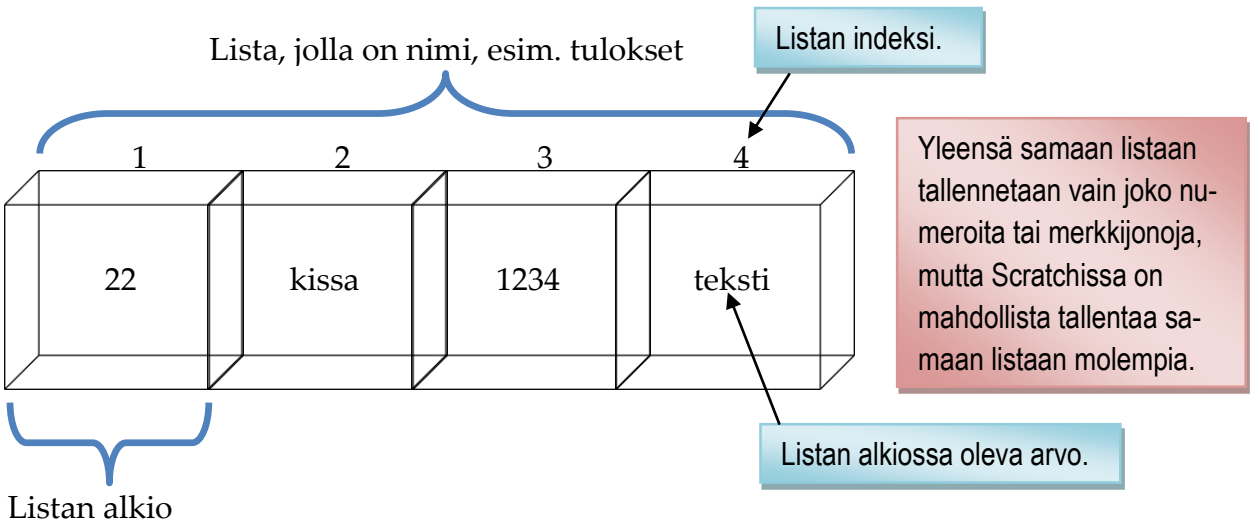
Muuttaa valintalistasta valitun muuttujan **<laskuri>** arvoa **<1>**:llä. Muutos voi olla positiivinen tai negatiivinen.


Tuo valintalistasta valitun muuttujan **<laskuri>** näkyviin näyttämölle.

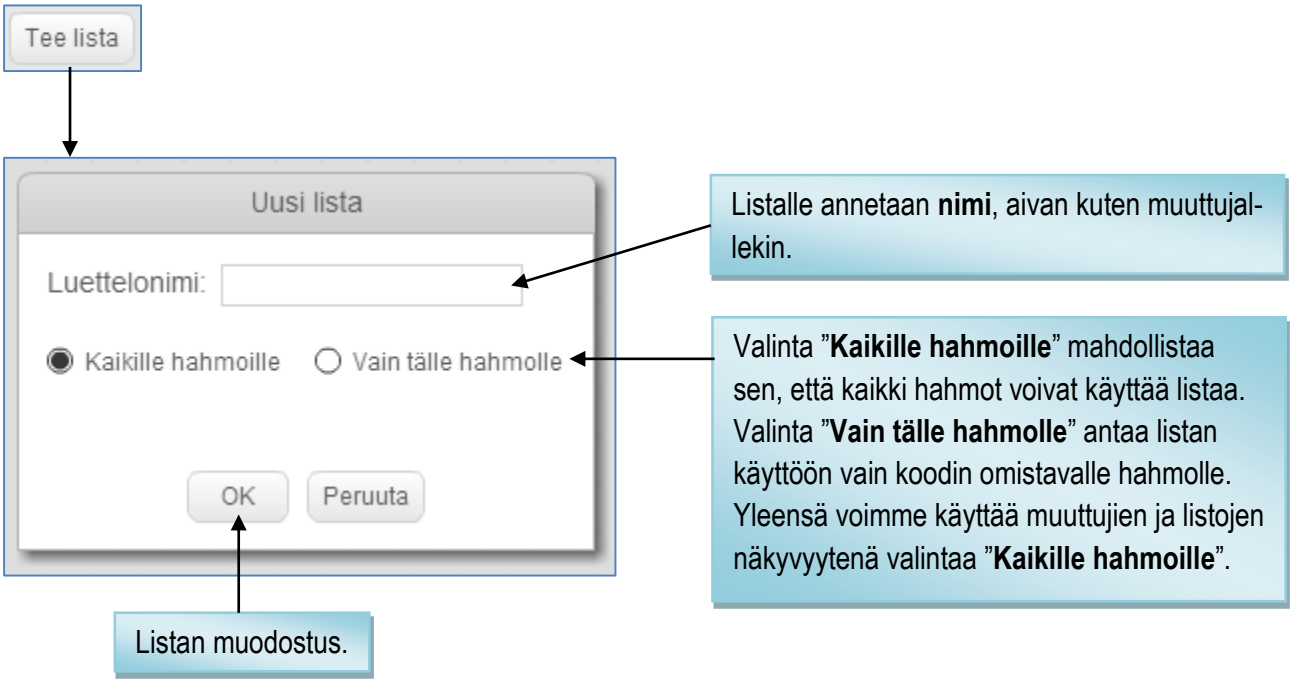
Poistaa valintalistasta valitun muuttujan **<laskuri>** näkyvistä näyttämöltä.

## Lista

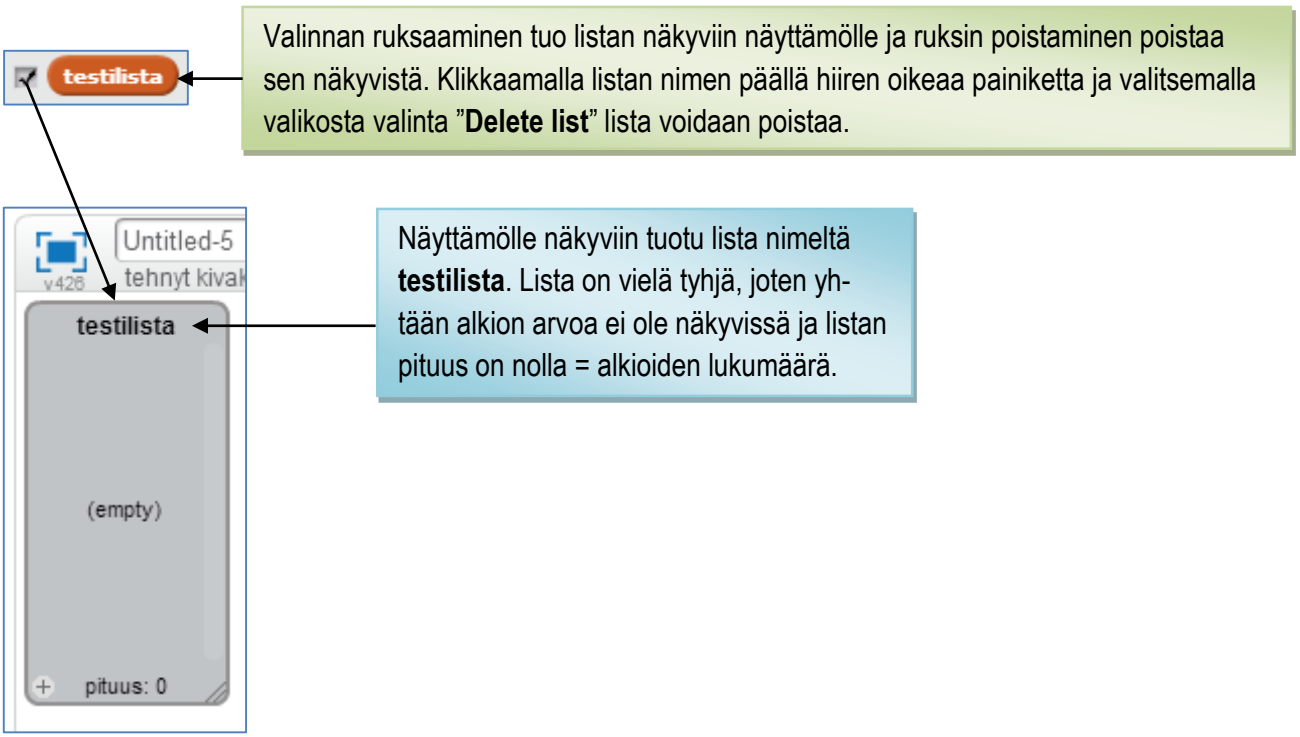
Ohjelmoinnissa muuttuja on hyvin kätevä ja niitä käytetään yleisesti. Muuttujalla on kuitenkin se heikko puoli, että yhteen muuttujaan voidaan tallettaa kerralla vain yksi arvo. Lista on puolestaan tietorakenne, johon voidaan tallentaa useita arvoja. Listoja käytetään, kun samaan asiaan liittyy paljon tallennettavaa tietoa. Listan voidaan ajatella muodostuvan toistensa perään liitetyistä laatikoista, joihin jokaiseen voidaan tallettaa arvo. Listalla on vain yksi nimi, mutta jokaisella laatikolla on oma järjestysnumero eli indeksi, jolla viitataan haluttuun kohtaan eli alkioon listassa.

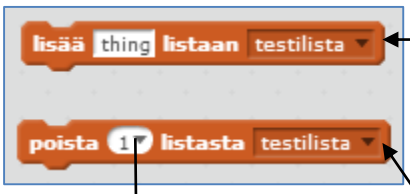


Lista muodostetaan painikkeesta ”**Tee lista**”.



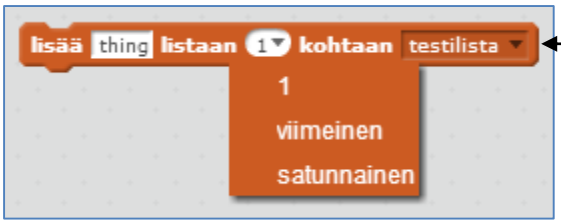

Seuraavissa kuvissa on muodostettu lista nimeltä **testilista** ja näytetään, miten listaa voidaan käsitellä komentopalikoilla.






Lisää listaan alkion ja antaa sille arvoksi parametrin **<thing>**. Lisäys kohdistuu valintalistasta valittuun listaan. **<Thing>** voi olla numero, merkkijono tai muuttujan arvo. Lisäys muodostaa uuden alkion listan loppuun.


Poistaa **ensimmäisen / viimeisen / kaikki** alkiot listasta. Poistettava kohde valitaan valintalistasta, samoin lista, johon poisto kohdistuu.



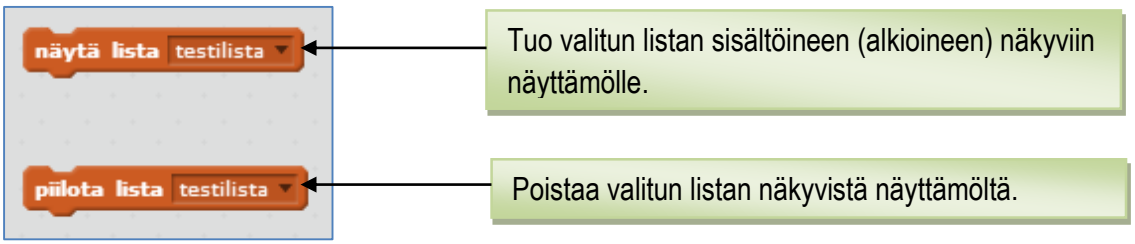

Lisää alkion arvolla **<thing>**, valittuun kohtaan listaa (**ensimmäinen / viimeinen / satunnainen**). Lisäys kohdistuu valintalistasta valittuun listaan.


Vaihtaa listan **ensimmäisen / viimeisen / satunnaisen** alkion arvon valintalistasta valittuun listaan. Parametri **<thing>** on uusi arvo alkiolle.


Tämä on **ominaisuuspalikka** ja se palauttaa listan **ensimmäisen / viimeisen / satunnaisen** alkion arvon valintalistasta valitusta listasta. Paluuarvona saadaan alkion sisältämä arvo. Ominaisuuspalikkaa ei voida käyttää yksinään, vaan se esiintyy jonkin toisen palikan osana.


Myös tämä on **ominaisuuspalikka** ja sillä voidaan lukea valintalistasta valitun listan pituus, eli siinä olevien alkioiden määrä = paluuarvo.


Tämä on **totuusarvopalikka**, joka palauttaa arvon **tosi** (true) tai **epätosi** (false). Jos valintalistasta valitun listan jollain alkiolla on arvo **<thing>**, paluuarvo on **tosi**. Jos arvoa **<thing>** ei löydy yhdestäkään listan alkiosta, paluuarvo on **epätosi**.

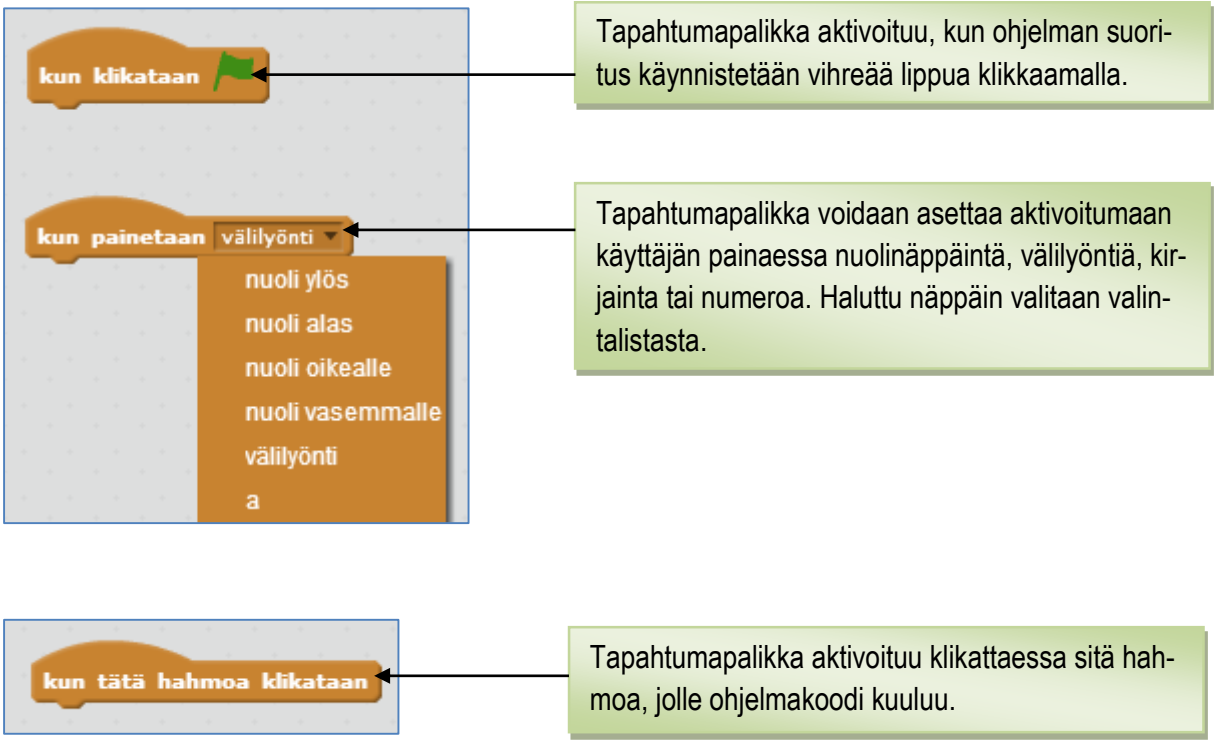



## Tapahtumat-osio

Tapahtumat voivat olla käyttäjän tai ohjelmakoodin itsenäisesti aikaansaamia. Puhuttaessa tapahtumapohjaisesta ohjelmoinnista tarkoitetaan sitä, että aina määrätyn tapahtuman tapahtuessa suoritetaan ohjelmassa tietty koodi. **Tapahtuma ja suoritettava koodi on siis sidottu toisiinsa**. Nykyinen Windows-ohjelmointi perustuu tapahtumiin ja niiden käsittelemiseksi muodostettuun ohjelmakoodiin.

Esimerkiksi yksinkertainen tapahtuma voisi olla se, että käyttäjä painaa välilyöntiä. Tähän tapahtumaan on mahdollista reagoida ohjelmakoodilla. Ensin tapahtuma ”napataan” tapahtuman koodipalikalla ja sitten sen alapuolelle sijoitetaan halutut toiminnot suorittavat koodipalikat.

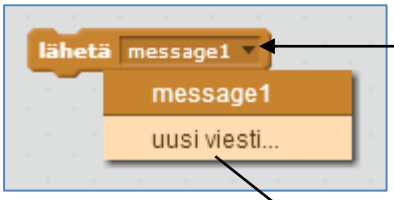
**Tapahtumat-osion koodipalikat ovat tärkeitä, sillä niitä käytetään kaikissa ohjelmissa koodin suorituksen käynnistäjinä**.



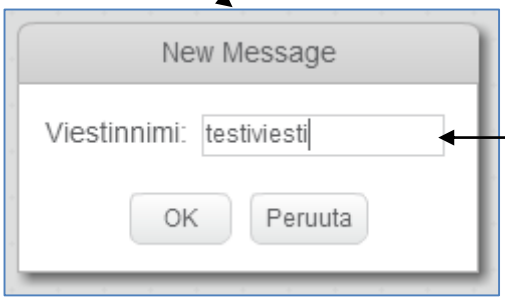

Tapahtumapalikka aktivoituu, kun näyttämön tausta vaihtuu valintalistasta valituksi taustaksi.


Tapahtumapalikka voidaan asettaa aktivoitumaan, kun **äänen voimakkuus**, **ajastin** tai **videoliike** ylittää asetetun **rajan <x>**. Äänen voimakkuus tulkitaan mikrofoniin tulevasta äänestä ja sillä voidaan tehdä ääniohjaus ohjelmaan. Ajastin aloittaa toimintansa ohjelman käynnistyksen yhteydessä ja se voidaan myöhemmin nollata ohjelmallisesti. Videoliikkeen ohjelma tunnistaa tietokoneen kameran kuvaamasta videokuvasta ja siinä tutkitaan vain hahmon kohdalla havaittavaa liikettä. Tämä mahdollistaa liikeohjauksen tekemisen ohjelmaan. Anna ohjelmalle tarvittaessa lupa käyttää tietokoneesi mikrofonia ja kameraa painamalla painiketta ”**Salli**” (Allow).

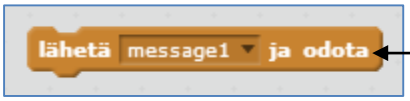

Jokaisella hahmolla on itsenäinen, toisista hahmoista riippumaton koodi. Hahmojen toiminta voidaan kuitenkin kytkeä yhteen **viesteillä**. Hahmo voi lähettää kaikille toisille hahmoille ja taustalle viestin. Tämä tapahtumapalikka aktivoituu, kun hahmo vastaanottaa valintalistasta valitun **nimetyn viestin**.




Viestin lähettäminen suoritetaan komentopalikalla ”**lähetä <viesti>**”. Uusi viesti muodostetaan valitsemalla valintalistasta valinta ”**uusi viesti…**” Viesti lähetetään kaikille hahmoille ja myös taustalle. Viestin lähetyksen tarkoituksena on yleensä ohjata muut hahmot tai tausta tekemään jotain.




Viestillä ei ole muuta tunnistetietoa kuin **viestinnimi**, joka syötetään tekstikenttää ja klikataan painiketta OK.



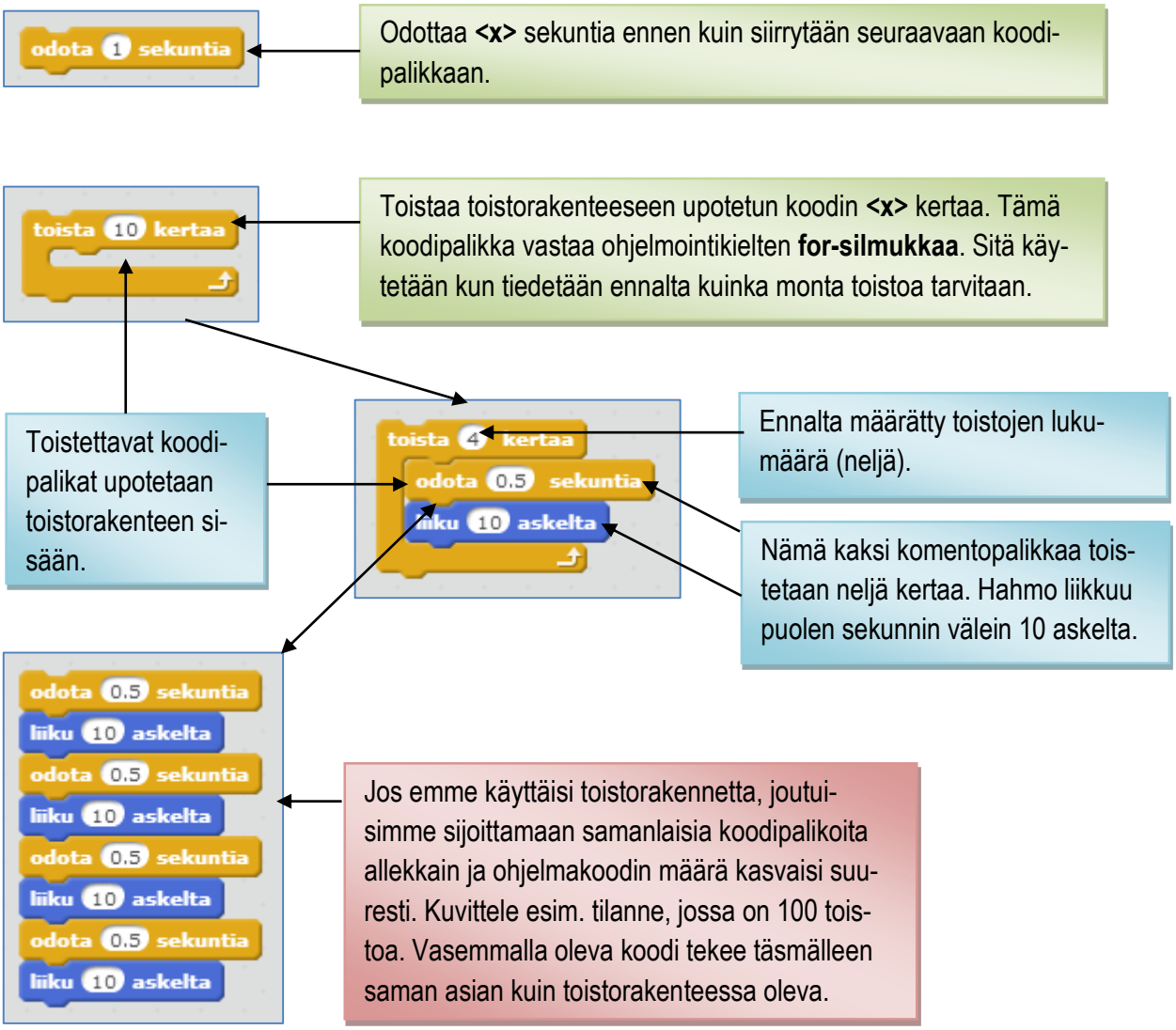

Tämä komentopalikka lähettää viestin samoin kuin edellinenkin, mutta jää odottamaan, että kaikki viestin vastaanottajat ovat suorittaneet viestin vastaanottavissa tapahtumapalikoissa olevat ohjelmakoodit, ennen kuin suoritusta jatketaan tämän palikan jälkeen.

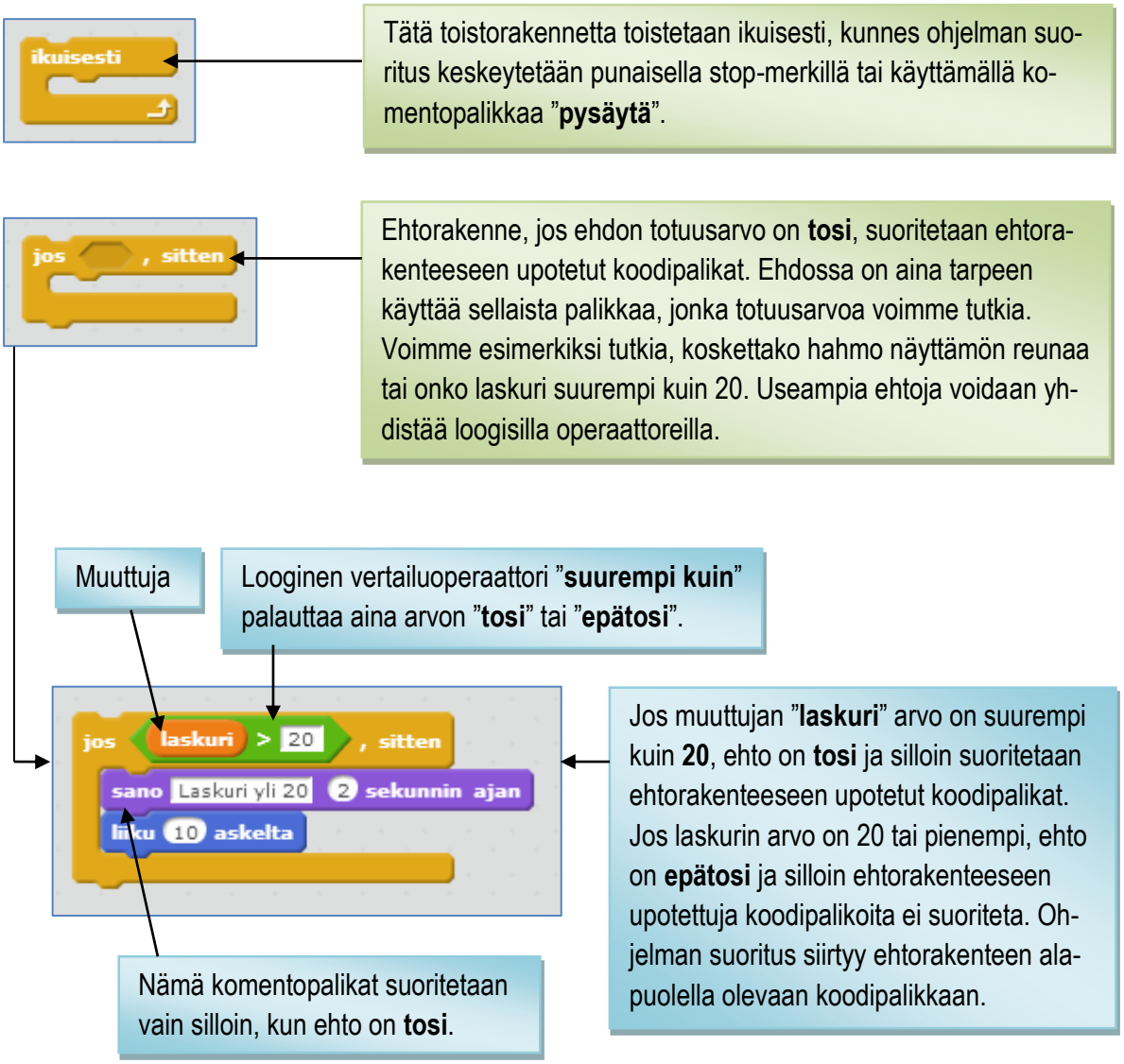
## Ohjaus-osio

Toisto- ja ehtorakenteen ovat tämän osion keskeisiä elementtejä. Ne ovat yleisiä kaikissa ohjelmointikielissä ja löytyvät myös ammattilaisten käyttämistä kielistä. Toistorakenteella voidaan toistaa yhtä tai useampaa koodipalikkaa joko ennalta valittu määrä tai sitten niin kauan kuin toiston ehto on voimassa. Sekä toisto- että ehtorakenteissa käytetään yleisesti muuttujia, vertailuoperaattoreita ja loogisia operaattoreita yhdistämään ehtoja. Ilman muuttujia tai ominaisuus-/totuusarvopalikoita niiden käyttö ei ole kovinkaan mielekästä.

Ehtorakenteessa tutkitaan, onko jokin tietty ehto **tosi** vai **epätosi** (true/false). Ehtorakenteita on kaksi erilaista, ensimmäinen sisältää ehdon ja vain yhden koodilohkon, joka suoritetaan jos ehto on tosi. Toinen ehtorakenne sisältää ehdon ja kaksi koodilohkoa, ensimmäinen suoritetaan jos ehto on tosi ja toinen jos se on epätosi.

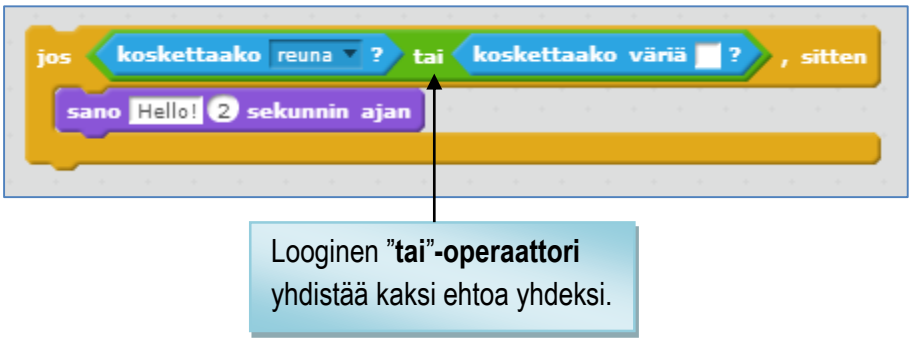
**Ohjaus-osion** koodipalikoiden kanssa tarvitaan yleensä myös useita muita koodipalikoita, mutta tässä muodostamme niihin ensin vain yleisen katsauksen. Perehdymme myöhemmin niihin perusteellisesti, sillä ne ovat hyvin oleellinen osa ohjelmointia. Niitä käyttämällä ohjelmakoodista voidaan tehdä lyhyempää, selkeämpää ja helpommin ylläpidettävää.





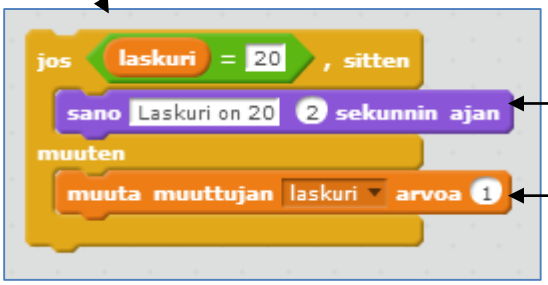

Ehdon toiminta perustuu totuusarvon tutkimiseen. Totuusarvo voidaan muodostaa esimerkiksi vertailuoperaattorilla ja silloin yleensä tutkitaan aina jonkin muuttujan tai ominaisuuden arvoa. Yhteen ehtoon voidaan liittää useita osaehtoja käyttämällä loogisia operaattoreita ”**ja**” (and) sekä ”**tai**” (or). Scratchissä on myös sellaisia totuusarvopalikoita, jotka palauttavat totuusarvon ja niitä voidaan käyttää ehdoissa sellaisenaan. Tällaisia ovat esimerkiksi palikat ”**koskettaako <reuna>**”, ”**koskettaako väriä <x>**”, ”**onko näppäintä <x> painettu**” ja ”**onko hiiren nappi painettu**”. Nämä totuusarvopalikat löytyvät **Tuntoaisti-osiosta**.

Alla on esitetty esimerkki siitä, miten loogisella ”**tai**”**-operaattorilla** voidaan yhdistää kaksi ehtoa yhdeksi ehdoksi. Koko ehto on **tosi**, jos hahmo koskettaa reunaa tai valkoista väriä.



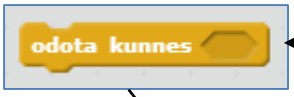

Looginen ”**tai**”**-operaattori** yhdistää kaksi ehtoa yhdeksi.


Tämä ehtorakenne sisältää kaksi vaihtoehtoista koodilohkoa. Itse ehto on täysin samanlainen kuin edellisessäkin, mutta ehdon totuusarvon suhteen ohjelmalla on nyt kaksi mahdollisuutta toimia. Jos ehto on **tosi**, suoritetaan ”**sitten**”**-osaan** upotetut koodipalikat. Jos ehto on **epätos**i, suoritetaan puolestaan ”**muuten**”**-osaan** upotetut palikat.

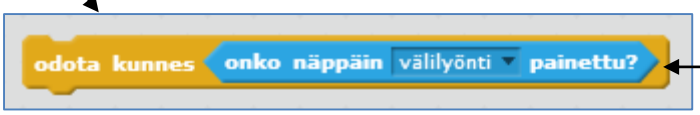



Suoritetaan vain silloin, kun laskuri = 20 (ehto = **tosi**).

Suoritetaan silloin, kun laskuri on erisuuri kuin 20 (ehto on **epätosi**).

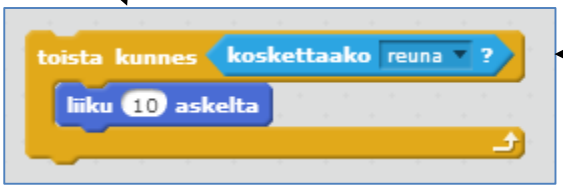



Asettaa ohjelman odotustilaan, kunnes annettu ehto muuttuu **todeksi**.

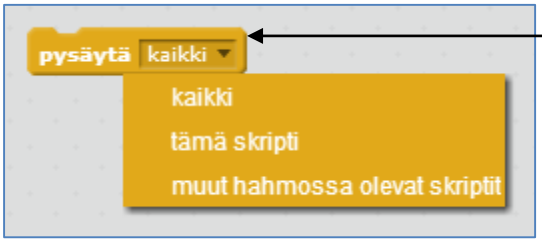



Odottaa, kunnes käyttäjä painaa välilyöntiä. Koodin suoritus jatkuu vasta tämän jälkeen.


Tätä toistorakennetta käytetään, kun ei etukäteen tiedetä, kuinka monta kertaa toistorakenne on tarpeen toistaa. Suorituskertojen lukumäärä määräytyy ehdon totuusarvon mukaan. Rakennetta toistetaan kunnes ehdon totuusarvo on **tosi** (ts. toistetaan niin kauan kuin ehdon totuusarvo on **epätosi**.)




Totuusarvopalikka ”**koskettaako <reuna>**” palauttaa arvon **tosi** hahmon koskettaessa reunaa. Toistorakenteessa siis liikutetaan hahmoa 10 askelta kerrallaan niin kauan, kunnes se koskettaa reunaa, jolloin totuusarvopalikka palauttaa arvon **tosi**.




Pysäyttää ohjelmallisesti joko kaikkien hahmojen ja taustan ohjelmakoodin suorituksen, tai pelkästään kyseisen koodin suorituksen, tai kaikki muut hahmossa olevat skriptit. Valinta ”**kaikki**” vastaa punaisen stop-merkin painamista.

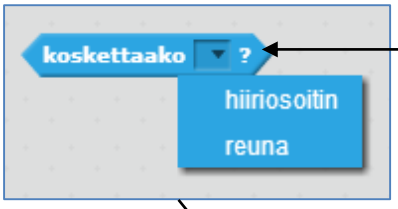

Tämä tapahtuma aktivoituu, kun hahmosta muodostetaan klooni. Tapahtuman ”nappaaminen” mahdollistaa kloonin ohjaamisen ohjelmakoodilla.


Muodostaa kloonin, eli kopion valintalistasta valitusta hahmosta. Klooni ilmestyy näyttämölle suoraan hahmon alle. Jotta klooni tulisi näkyviin, täytyy hahmoa tai sen kloonia liikuttaa.

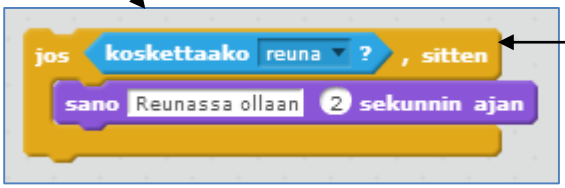

Kun tätä komentoa käytetään kloonin ohjelmakoodissa, se poistaa kyseisen kloonin näyttämöltä. Jos komentoa käytetään hahmon koodissa, se poistaa kaikki kloonit näyttämöltä.

## Tuntotaisti-osio

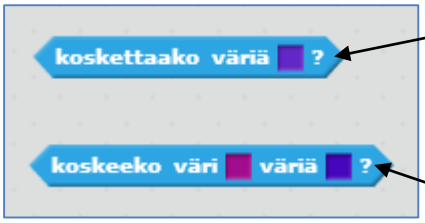
Useimmat tuntoaisti-osion koodipalikat ovat ominaisuus- ja totuusarvopalikoita. Ominaisuuspalikoilla voidaan selvittää esimerkiksi kaikki hahmon oleelliset ominaisuudet sekä päivämäärä ja kellonaika. Totuusarvopalikoilla puolestaan voidaan mm. selvittää koskettaako hahmo reunaa tai jotain väriä sekä onko jotain tiettyä näppäintä tai hiiren näppäintä painettu. Sekä ominaisuus- että totuusarvopalikoita käytetään yleisesti ehto- ja toistorakenteissa.




Palauttaa totuusarvon **tosi**, jos hahmo koskettaa hiiriosoitinta, näyttämön reunaa tai toista hahmoa. Kosketuksen kohde valitaan valintalistasta.

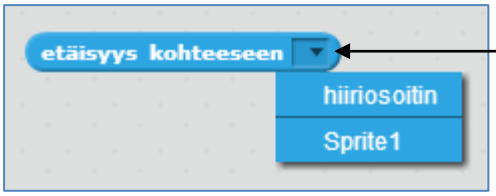



Ehtorakenteessa tutkitaan, koskettaako hahmo näyttämön reunaa. Jos koskettaa, ehdon arvo on **tosi** ja silloin suoritetaan ”**sano…**”-koodipalikka.

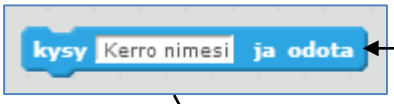



Palauttaa totuusarvon **tosi**, jos hahmo koskettaa neliössä olevaa väriä. Väri voi olla toisessa hahmossa tai näyttämön taustassa. ”**Väri**” valitaan klikkaamalla hiirellä ensin neliötä ja sitten haluttua väriä jostain kohtaa kuvaruutua (colorpicker).

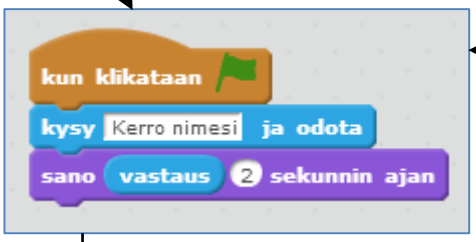
Ensimmäinen ”**väri**” on hahmossa oleva väri ja toinen joko toisessa hahmossa tai taustassa oleva väri. Palauttaa totuusarvon **tosi**, jos värit koskettavat toisiaan.




Ominaisuus, josta voidaan lukea kyseisen ohjelmakoodin omistavan hahmon ”**etäisyys**” toiseen hahmoon tai hiiriosoittimeen. Kohde, johon hahmon etäisyys mitataan, valitaan valintalistasta.

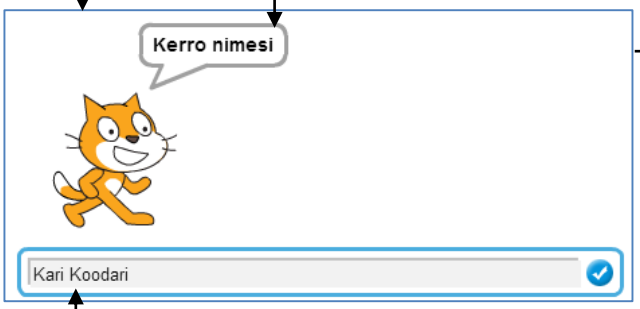



Komento näyttää käyttäjälle parametrina annetun tekstin ja antaa käyttäjälle mahdollisuuden kirjoittaa vastaus tekstikenttään. Ohjelman suoritus jatkuu vasta, kun käyttäjä on syöttänyt vastauksensa. Vastaus tallentuu ominaisuuteen ”**vastaus**”, josta se voidaan myöhemmin ohjelmallisesti lukea.




Koodi kysyy käyttäjältä nimeä ja hänen syöttämänsä merkkijono tallentuu ominaisuuteen ”**vastaus**”. Lopuksi käyttäjän syöttämä vastaus näytetään hahmon puhekuplalla.

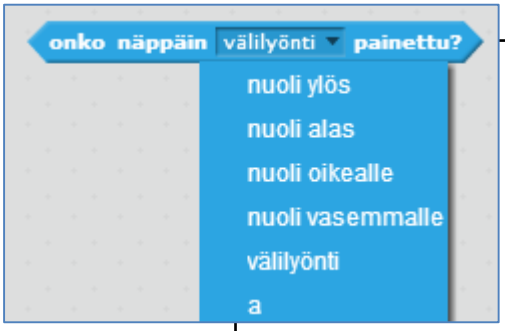
Hahmo esittää käyttäjälle kysymyksen.



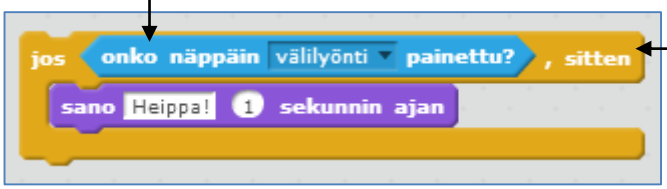


Tekstikenttä, johon käyttäjä voi kirjoittaa vastauksensa. Lopuksi painetaan Enter tai klikataan hiirellä sinistä kuvaketta.

Käyttäjän syöttämä merkkijono tallentuu ominaisuuteen ”**vastaus**”, josta se voidaan lukea ja esim. näyttää puhekuplalla.




Palauttaa totuusarvon **tosi**, jos käyttäjä on painanut valintalistasta valittua **näppäintä**. Tällä voidaan toteuttaa nopea reagointi käyttäjän painamiin näppäimiin.




Ehtorakenteen ehtona käytetään totuusarvopalikkaa ”**onko näppäin <välilyönti> painettu?**” Hahmo sanoo ”Heippa!”, jos käyttäjä on painanut välilyöntiä.


Palauttaa totuusarvon **tosi**, jos käyttäjä on painanut hiiren vasenta painiketta.


Ominaisuus, josta voidaan lukea hiiriosoittimen **x-koordinaatti** näyttämöllä.

Ominaisuus, josta voidaan lukea hiiriosoittimen **y-koordinaatti** näyttämöllä.


Ominaisuus, josta voidaan lukea tietokoneen mikrofonista tulevan **äänen voimakkuus** asteikolla 1-100. Ominaisuus mahdollistaa ääniohjauksen käytön ohjelmissa. Valinnan ruksaaminen tuo voimakkuuden näkyviin näyttämölle. Ohjelma kysyy lupaa käyttää tietokoneesi mikrofonia, vastaa ”**Allow**” (salli).


Esimerkki täydellisestä ohjelmasta. Ohjelma käynnistyy vihreää lippua klikkaamalla. Tämän jälkeen alkaa ikuinen toistorakenne. Ehtorakenteen ehdossa tutkitaan, onko mikrofonista tulevan äänen voimakkuus suurempi kuin 40 ja jos on, niin hahmo sanoo ”Kuulen ääntä”.


Liike tai suunta.

Hahmo tai näyttämö (esiintymislava)

Ominaisuus, josta voidaan lukea kuinka paljon **liikettä** tai **suuntaa** on tietokoneen kameran ottamassa videokuvassa. Ominaisuuden käyttö edellyttää luonnollisesti tietokoneeseen liitettyä kameraa. Videoliikkeen määrää ja suuntaa voidaan tutkia joko pelkästään hahmon kohdalla tai koko näyttämön alueella. Valinnan ruksaaminen tuo videoliikkeen määrän näkyviin näyttämölle. Ohjelma kysyy lupaa käyttää tietokoneesi kameraa, vastaa ”**Allow**” (salli).


Jos tietokoneen kameran ottaman videokuvan liikkeen määrä on yli 40, hahmo sanoo ”Näen liikettä”.

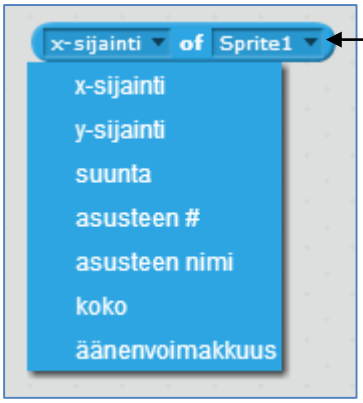

Komento kääntää valintalistan valinnan mukaan tietokoneen kameran **päälle**, **pois päältä** tai **päälle horisontaalisesti käännettynä**. Käännettäessä kamera päälle ohjelma kysyy käyttäjältä luvan, vastaa ”**Allow**” (salli).


Tietokoneen kameran ottama kuva näytetään näyttämön taustalla. Tällä komennolla asetetaan **videokuvan läpinäkyvyys** prosenttilukuna 0-100. Suurempi luku tekee videosta enemmän läpinäkyvän.

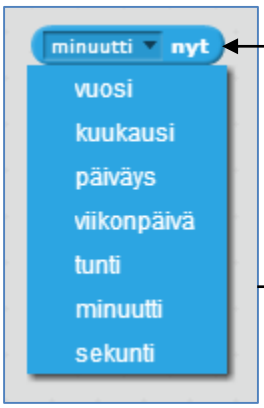

Ominaisuudesta voidaan lukea **ajastimen** arvo. Ajastimen arvo lähtee liikkeelle nollasta ohjelman käynnistyksen yhteydessä, mutta se voidaan nollata myös ohjelmallisesti. Valinnan ruksaaminen tuo ajastimen arvon näkyviin näyttämölle.


Nollaa ajastimen ohjelmallisesti.

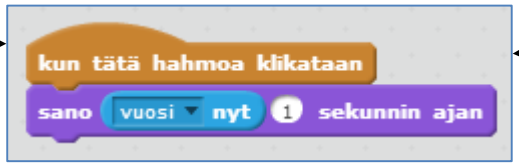

Ohjelman käynnistyttyä alkaa ikuinen silmukka. Ehtorakenteen ehdossa tutkitaan, onko ajastimen arvo yhtä suuri kuin 5. Jos on, hahmo sanoo ”5 sekuntia”. Tämän jälkeen nollataan ajastin ohjelmallisesti ja sama toistuu viiden sekunnin päästä.




Jälkimmäisestä valintalistasta valitaan, halutaanko lukea hahmon (Sprite) vai näyttämön (Stage, esiintymislava) ominaisuuksia. Hahmon ominaisuuksista voidaan lukea **hahmon x-ja y-koordinaatti, suunta, asusteen numero, asusteen nimi, koko ja äänenvoimakkuus**. Näyttämön luettavia ominaisuuksia ovat puolestaan **taustan numero ja nimi sekä äänenvoimakkuus**.




Ominaisuudesta voidaan lukea tämänhetkinen **vuosi, kuukausi, päivä, viikonpäivä, tunti, minuutti ja sekunti** valintalistan valinnan mukaisesti.




Hahmon klikkaus kertoo tämänhetkisen vuosiluvun.


Ominaisuudesta voidaan lukea, kuinka monta päivää on kulunut vuoden 2000 vaihtumisen jälkeen tähän päivään mennessä.

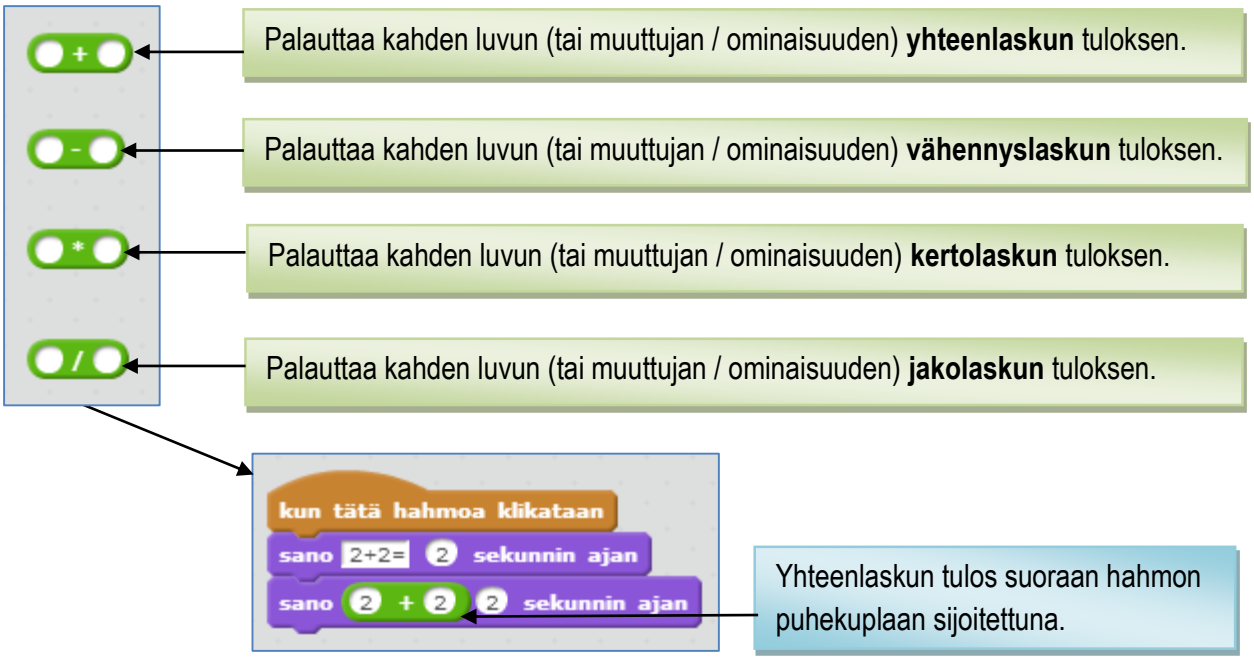

Ominaisuudesta voidaan lukea kirjautuneen käyttäjän käyttäjänimi.

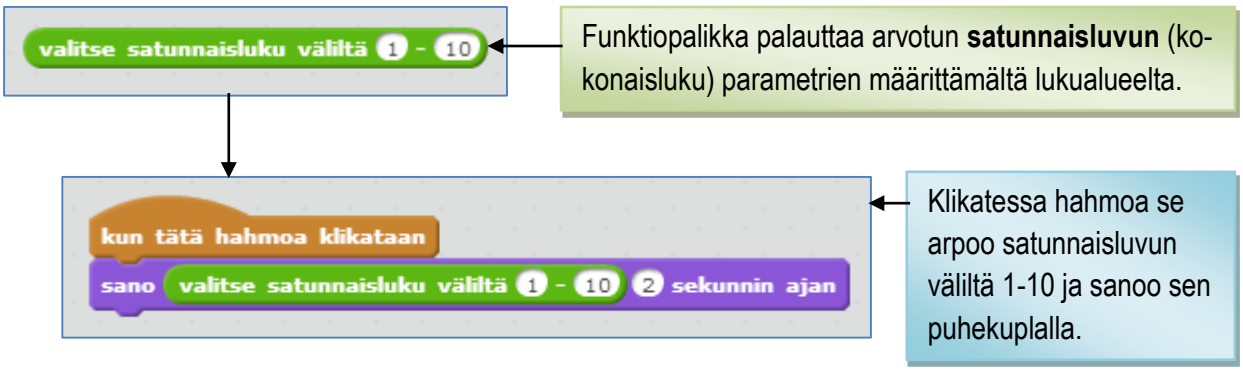
## Toiminnot-osio

Toiminnot-osion koodipalikoista löytyy mm. kolmenlaisia operaattoreita: matemaattiset, vertailu- ja loogiset operaattorit. Näiden lisäksi käytettävissä on funktiopalikoita, joilla voidaan toteuttaa satunnaisluvun arvonta, muuttujien yhdistäminen, kirjaimen poiminta merkkijonosta, muuttujan pituuden selvitys, jakojäännös sekä desimaaliluvun pyöristys kokonaisluvuksi. **Mitään Toiminnot-osion koodipalikoista ei voida käyttää itsenäisesti, vaan ne upotetaan aina jonkin toisen koodipalikan sisään**. Esimerkiksi matemaattisilla operaattoreilla voidaan laskea uusi arvo muuttujalle ja vertailuoperaattoreita käytetään puolestaan ehto- ja toistorakenteen ehdoissa.

### Matemaattiset operaattorit

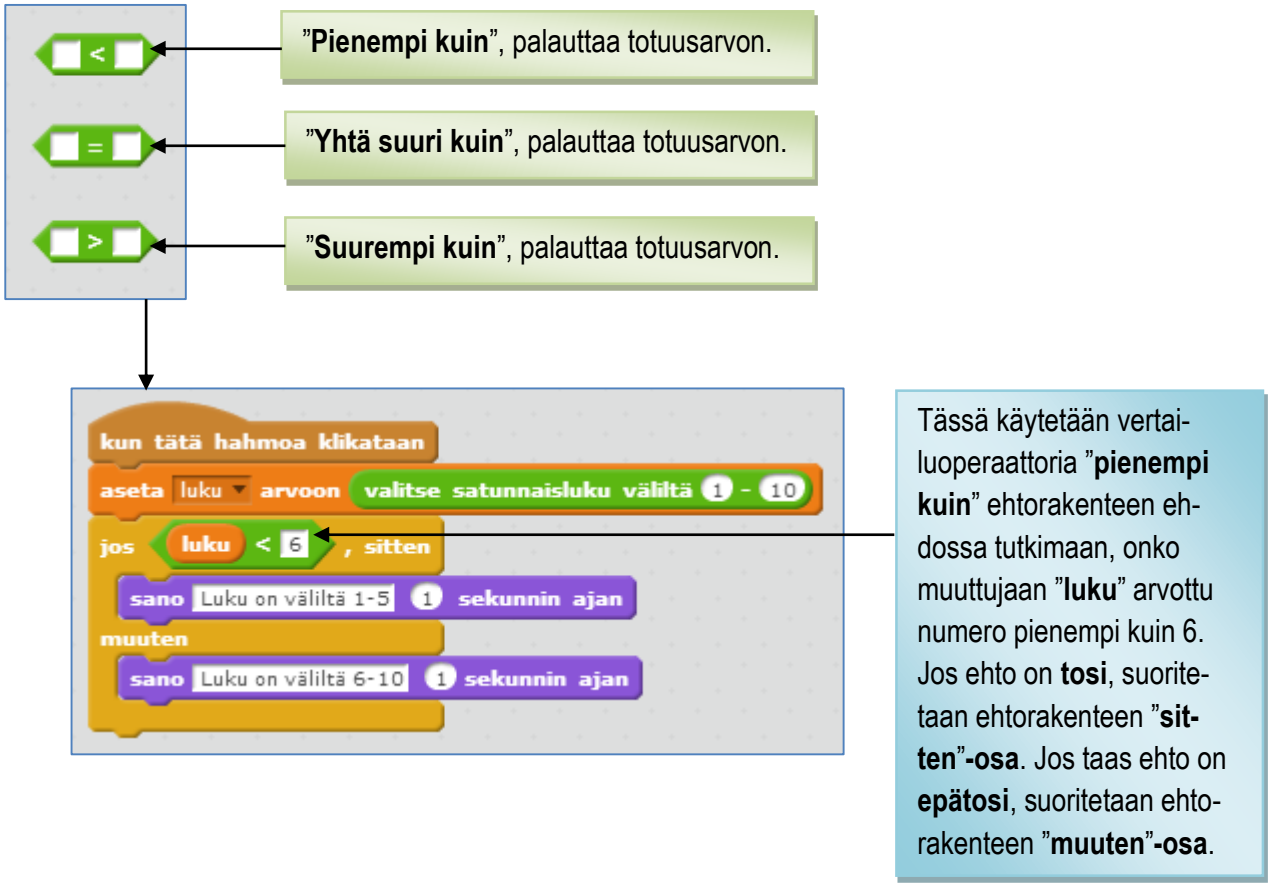
Matemaattisia operaattoreita käytetään luvuille sekä numeroarvoisille muuttujille ja ominaisuuksille. Matemaattiset operaattorit palauttavat aina numeroarvon.



### Vertailuoperaattorit

Vertailuoperaattoreilla tutkitaan, onko jonkin numeerisen muuttujan tai ominaisuuden arvo alle, yli tai yhtä suuri kuin tietty raja-arvo. Vertailun ensimmäinen parametri on tällöin tutkittava kohde ja toinen parametri raja-arvo, esimerkiksi **laskuri < 20**. Vertailuoperaattorit palauttavat aina totuusarvon **tosi** tai **epätosi**.



## Loogiset operaattorit

Loogisia operaattoreita käytetään aina ehdoissa yhdistämään useampia osaehtoja yhdeksi ehtolauseeksi. Esimerkiksi jos haluat yhdellä ehtolauseella selvittää, koskettaako hahmo toista hahmoa ja reunaa, voit käyttää silloin loogista **”ja”-operaattoria**.

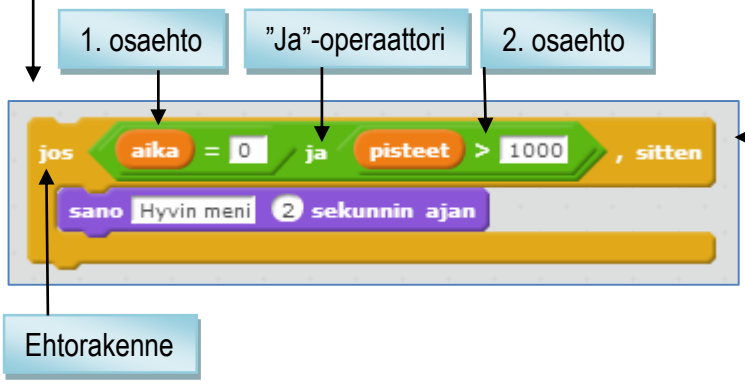
Ehdot muodostetaan aina vertailuoperaattoreilla tai totuusarvon palauttavilla totuusarvopalikoilla ja ne sijoitetaan loogisen operaattorin parametreiksi. Loogisten operaattoreiden käyttö vähentää tarvittavan ohjelmakoodin määrää ja mahdollistaa helpommin ylläpidettävän koodin muodostuksen. Ehtoja käytetään ehto- ja toistorakenteissa, joten tässä kytkeytyy yhteen kolme asiaa: Ehto-/toistorakenne, looginen operaattori ja vertailuoperaattori/totuusarvon palauttava totuusarvopalikka.

**”ja”-operaattori**


Looginen **”ja”-operaattori** (and), palauttaa totuusarvon **tosi**, jos molemmat osaehdot ovat totta.

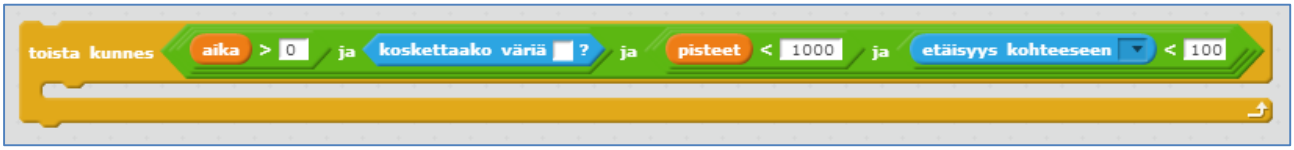
**”Ja”-operaattorin totuustaulu:**

| 1. parametri | 2. parametri | Palautus |
|---|---|---|
| tosi | tosi | tosi |
| tosi | epätosi | epätosi |
| epätosi | tosi | epätosi |
| epätosi | epätosi | epätosi |




Ehtorakenteessa käytetään **”ja”-operaattoria** yhdistämään kaksi osaehtoa yhdeksi ehtolauseeksi. Ensimmäisessä osaehdossa tutkitaan vertailuoperaattorilla, onko muuttuja **aika = 0**. Toisessa osaehdossa tutkitaan, onko muuttuja **pisteet > 1000**. Osaehdoissa tutkitaan muuttujien arvoja. Jos molempien osaehtojen totuusarvo on **tosi**, hahmo sanoo ”Hyvin meni”.

”Ja”-operaattorilla voidaan yhdistää useampiakin ehtoja kuin kaksi. Alla on esimerkki toistorakenteesta, jonka ehtoon on yhdistetty ”ja”-operaattorilla yhteensä neljä osaehtoa.

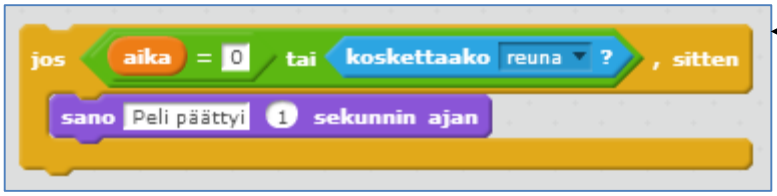


”tai”-operaattori


Looginen ”**tai**”**-operaattori** (or), palauttaa totuusarvon **tosi**, jos molemmat tai toinen osaehdoista on totta.

”Tai”-operaattorin totuustaulu:

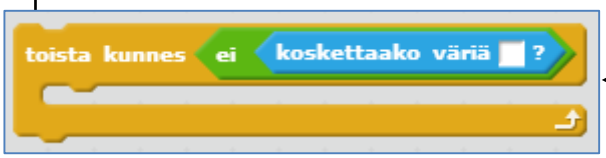
| 1. parametri | 2. parametri | Palautus |
|---|---|---|
| tosi | tosi | tosi |
| tosi | epätosi | tosi |
| epätosi | tosi | tosi |
| epätosi | epätosi | epätosi |




Ehtorakenteessa käytetään ”**tai”-operaattoria** yhdistämään kaksi osaehtoa. Jos muuttuja ”**aika**” on nolla **tai** hahmo koskettaa näyttämön reunaa (totuusarvopalikka), suoritetaan ehtorakenteeseen upotettu koodipalikka. Riittää siis, että jompikumpi osaehdoista saa totuusarvon **tosi**.

”ei”-operaattori


Looginen ”**ei**”**-operaattori** (not), muuttaa totuusarvon **tosi** arvoksi **epätosi** ja vastaavasti arvon **epätosi** arvoksi **tosi**.




Totuusarvopalikka ”**koskettaako väriä <valkoinen>?**” palauttaa totuusarvon **tosi** hahmon koskettaessa valkoista väriä. **Ei-operaattori** kuitenkin muuntaa tämän arvon päinvastaiseksi. Tämän vuoksi toistorakennetta toistetaan niin kauan, kun hahmo koskettaa valkoista väriä. Toistorakenteen ”**Toista kunnes**” suoritus päättyy, kun ehdon arvo on **tosi**.

Ei-operaattorin totuustaulu:

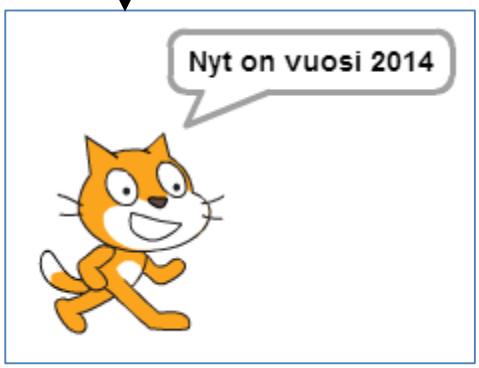
| parametri | Palautus |
|---|---|
| tosi | epätosi |
| epätosi | tosi |

## Toiminnot-osion funktiopalikoita


Palauttaa merkkijonon, jossa on **yhdistetty** kaksi merkkijonoa yhdeksi jonoksi. Yhdistettävät merkkijonot annetaan parametreina ja toinen tai molemmat voivat olla myös muuttujia tai ominaisuuksia.


Yhdistetään merkkijono ”**Pisteet**” ja muuttujan ”**pisteet**” arvo yhdeksi merkkijonoksi ja sanotaan se puhekuplalla.


Yhdistetään merkkijono ”**Nyt on vuosi**” ja **ominaisuudesta** ”**vuosi**” luettu nykyinen vuosiluku yhdeksi merkkijonoksi ja sanotaan se puhekuplalla.

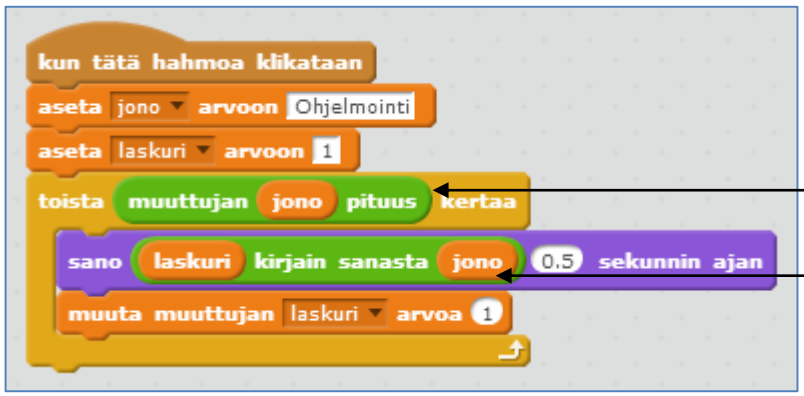




Palauttaa **<n>:nen kirjaimen** parametrina annetusta **sanasta** tai **muuttujan arvosta**. Viereinen esimerkki palauttaa kirjaimen ”w”.


Palauttaa parametrina annetun **merkkijonon pituuden**. Parametri voi olla myös merkkijonomuuttuja. Viereinen esimerkki palauttaa luvun 5.

Esimerkki kahden em. funktiopalikan käytöstä.

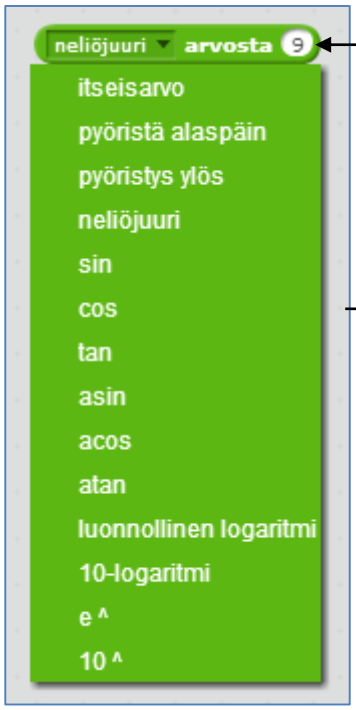



Toistorakenteen toistojen lukumäärä = muuttujan ”**jono**” pituus (merkkien määrä).

Poimitaan yksi merkki kerrallaan muuttujasta **jono** ja sanotaan se puhekuplalla.


Palauttaa kahden luvun jakojäännöksen. Ensimmäinen parametri on jaettava ja toinen jakaja.


**Pyöristää** parametrina annetun desimaaliluvun ja palauttaa sen lähimpänä kokonaislukuna.



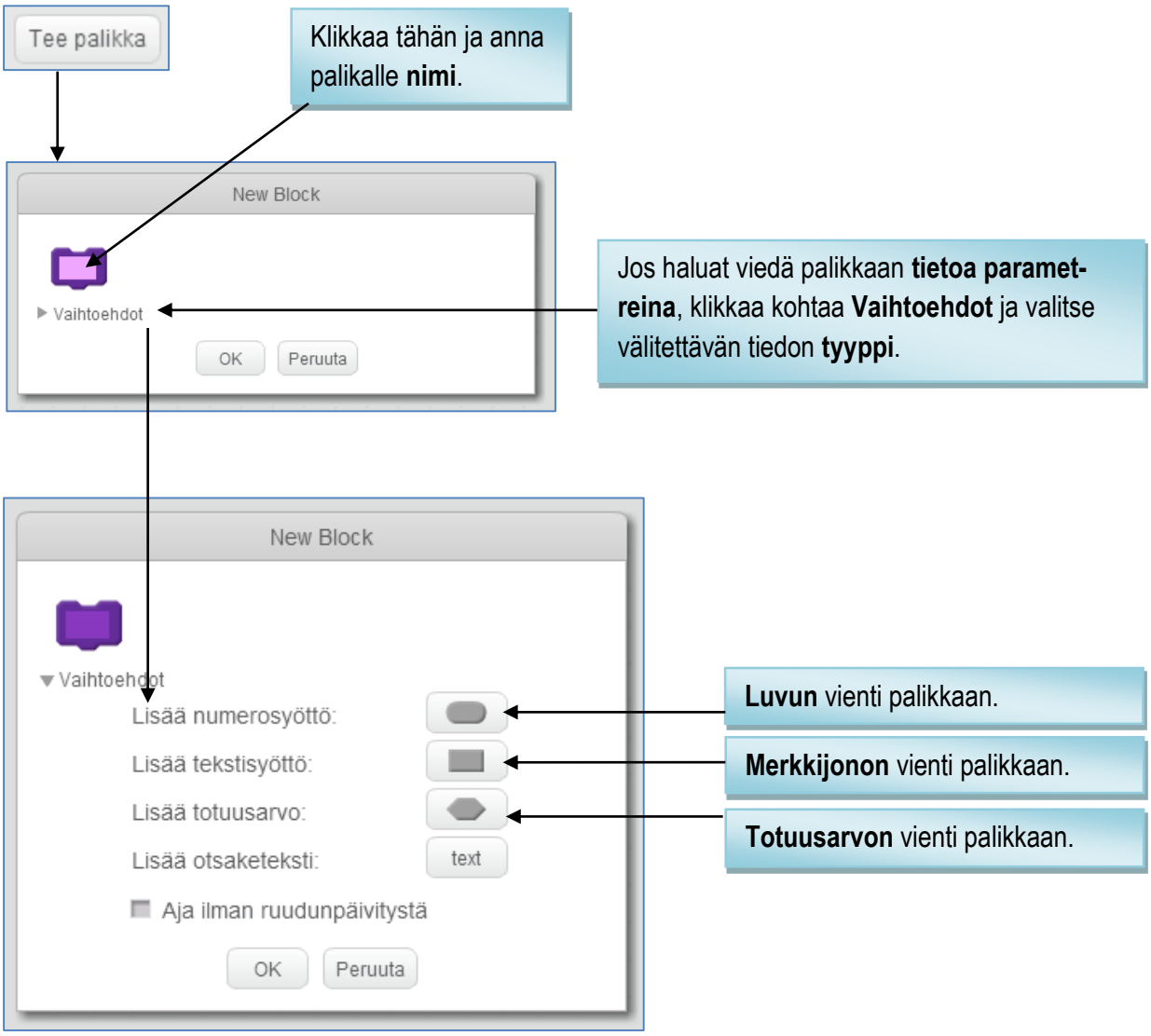

Valintalistasta valitaan haluttu **matemaattinen funktio** ja parametriksi annetaan luku tai muuttuja, jolle funktio suoritetaan. Paluuarvona on funktion tulos.


Asetetaan muuttujan ”**tulos**” arvoksi luvun 25 neliöjuuri ja sanotaan tulos puhekuplalla.

## Lisää lohkoja-osio

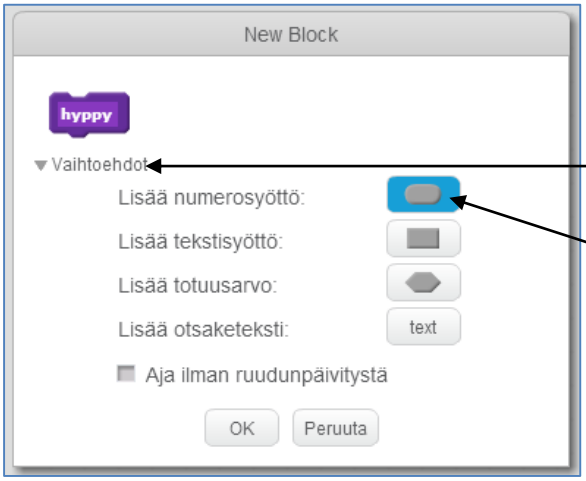
Tästä osiosta löytyvällä painikkeella ”**Tee palikka**” voit tehdä itse omia palikoita, jotka sisältävät haluamiasi koodipalikoita. Palikat ovat luonteeltaan aliohjelmia. Ammattilaisten käyttämissä ohjelmointikielissä koko ohjelma jaetaan pieniin osiin, joita kutsutaan aliohjelmiksi (tai funktioiksi, jotka voivat myös palauttaa arvon). Aliohjelmat tekevät ohjelman rakenteesta huomattavasti selkeämmän. Lisäksi ne lyhentävät tarvittavan ohjelmakoodin määrää oleellisesti. Yleisesti ottaen sellaisesta ohjelmakoodista, jota tarvitaan useammassa eri paikassa, kannattaa tehdä aliohjelma. Tällöin samaa koodia ei tarvitse toistaa ohjelmassa, vaan haluttu toiminto voidaan suorittaa käyttämällä aliohjelmaa. Joskus aliohjelmille on tarpeen välittää tietoa, jota se käyttää omassa toiminnassaan. Tämä tieto voidaan välittää aliohjelman parametreilla. Scratchissä aliohjelmia kutsutaan palikoiksi, joten puhumme jatkossa vain palikoista.

Palikan muodostus aloitetaan klikkaamalla painiketta ”**Tee palikka**”.



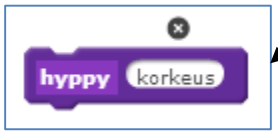
Alla olevissa kuvissa näytetään, miten tehdään sellainen palikka, johon viedään parametrina numeromuotoinen tieto. Palikan käyttötarkoitus on hypäyttää hahmoa ylöspäin parametrin määrittämän askelmäärän verran.


1. Annetaan palikalle nimeksi ”**hyppy**”.



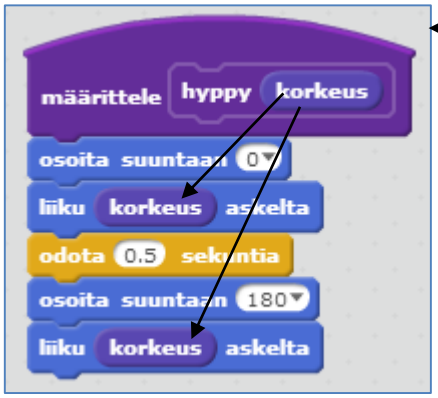

2. Klikataan auki kohta ”**Vaihtoehdot**”.

3. Klikataan kuvaketta ”**Lisää numerosyöttö**”.


4. Palikan nimen perään muodostuu parametri, jonka nimi on aluksi **number1**, kirjoitetaan tähän uudeksi nimeksi ”**korkeus**” ja painetaan Enter tai klikataan painiketta OK.


5. Nyt meillä on ”**hyppy**”-niminen palikka, johon voidaan viedä tietoa ”**korkeus**”-nimisellä parametrilla. Tämän palikan alle voimme liittää muita koodipalikoita ja käyttää niissä ”**korkeus**”**-parametrin** arvoa aivan kuten muuttujaa.




6. Palikkaan on lisätty koodi, joka liikuttaa hahmoa parametrin ”**korkeus**” osoittaman määrän ylöspäin, odottaa puoli sekuntia ja siirtää hahmoa saman verran alaspäin. Parametri ”**korkeus**” raahataan hiirellä ylimmästä palikasta niihin alapuolella oleviin palikoihin, joissa sitä halutaan käyttää.

Palikan hyöty tulee esille ennen kaikkea silloin, kun samaa ohjelmakoodia tarvitaan useammassa paikassa. Voimme esimerkiksi ohjelmoida kaksi erillistä näppäintä aiheuttamaan sen, että hahmo hyppää ylöspäin. Välilyönnistä hahmo ohjelmoidaan hyppäämään 100 askelta ylöspäin ja näppäimestä ”nuoli ylöspäin” 200 askelta.


7. Tämä on ns. pääohjelma, jossa ohjelman suoritus pyörii ikuisessa toistorakenteessa. Käyttäjän painaessa välilyöntiä ohjelman suoritus siirtyy palikkaan ”**hyppy**” ja parametrin ”**korkeus**” arvoksi tulee **100**. Painettaessa näppäintä ”nuoli ylöspäin” toiminta on muuten samanlaista, mutta parametrin ”**korkeus**” arvoksi tulee nyt **200**.


”**Hyppy**”-palikka löytyy osiosta ”**Lisää lohkoja**”. Palikka raahataan hiirellä vetämällä haluttuun kohtaan pääohjelmaa ja annetaan parametrille haluttu arvo.

# Ohjelmoinnin perusteita

Tässä kappaleessa tutustutaan Scratch-ohjelmointiin laatimalla pieniä esimerkkiohjelmia. Laadittavissa ohjelmissa tuodaan esiin Scratch-ohjelmoinnin keskeiset periaatteet. Esimerkkien ohjelmakoodeista näet, miten erilaisia koodipalikoita käytetään ja miten niitä voidaan yhdistellä. Erilaisia ohjelmia voidaan laatia rajaton määrä ja tämän kappaleen tarkoituksena on antaa riittävät valmiudet omien ohjelmien suunnittelemiseksi ja toteuttamiseksi. Ensimmäiset ohjelmaesimerkit käydään läpi hyvin perusteellisesti. Tämän jälkeen oletetaan, että tiedät ohjelman muodostuksen keskeiset periaatteet ja sen, mistä mitkäkin koodipalikat löytyvät. Myöhemmät esimerkit ovat vähän enemmän pelkistettyjä kuvauksia ohjelmista. Ennen varsinaisen ohjelmoinnin aloitusta on hyvä määritellä kirjallisesti mitä ohjelman halutaan tekevän. Tätä vaihetta kutsutaan ohjelman määrittelyksi ja myöhempien ohjelmaesimerkkien alussa onkin aina lyhyt kirjallinen määritys ohjelmista.

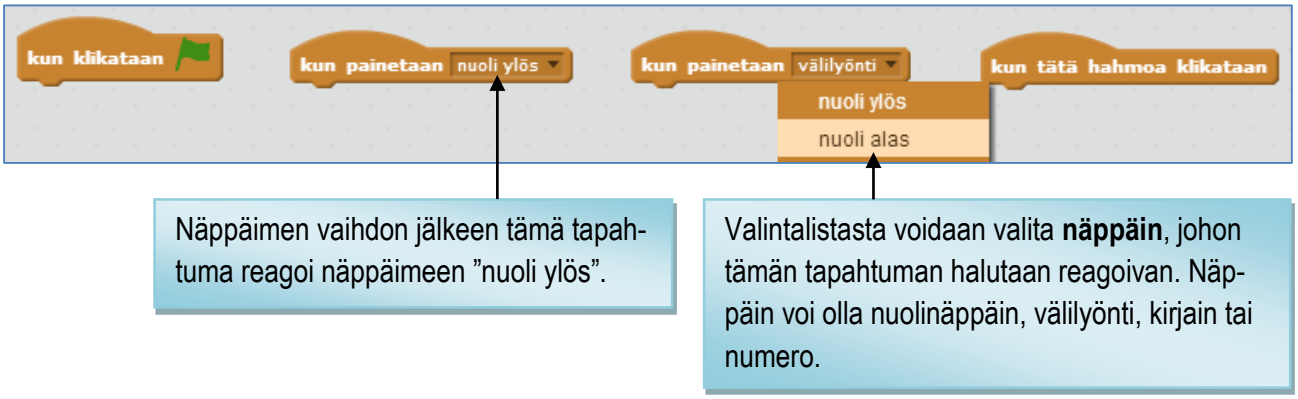
## Tapahtumiin reagoiminen

Ohjelmakoodin suoritus alkaa aina jostain tapahtumasta. Hyvin usein tapahtuman aiheuttaa käyttäjä, esimerkiksi käynnistämällä ohjelman vihreästä lipusta, painamalla jotain näppäintä tai klikkaamalla hiirellä. Nämä tapahtumat on mahdollista ”napata” **Tapahtumat-osiosta** löytyvillä tapahtumapalikoilla. Tapahtumapalikka ”**kun klikataan vihreä lippu**” on hyvin keskeinen tapahtumakäsittelijä, sillä se aktivoituu aina käyttäjän käynnistäessä ohjelman. Tämän tapahtumapalikan alle voidaan lisätä muut palikat, jotka halutaan suorittaa ohjelman käynnistyksen yhteydessä.

Tehdään ohjelma, joka reagoi käynnistykseen, hahmon klikkaamiseen hiirellä sekä ”nuoli ylös”- ja ”nuoli alas”-näppäinten painalluksiin. Havaittuaan ohjelman käynnistyksen hahmo sanoo ”Ohjelma käynnistettiin” ja klikattaessa hahmoa hiirellä se sanoo ”Mua klikaattiin”. Näppäin ”nuoli ylöspäin” suurentaa hahmon kokoa ja näppäin ”nuoli alaspäin” pienentää sitä”.

Uudessa projektissa on kissahahmo jo valmiina, joten meidän ei tarvitse lisätä hahmoa. Raahataan ensin koodialueelle ne tapahtumapalikat, jotka ”nappaavat” tarvittavat tapahtumat. Mene ”**Tapahtumat**”**-osioon** ja raahaa koodialueelle palikat ”**kun klikataan vihreä lippu**”, kaksi kappaletta ”**kun painetaan <välilyönti>”** ja ”**kun tätä hahmoa klikataan**”.

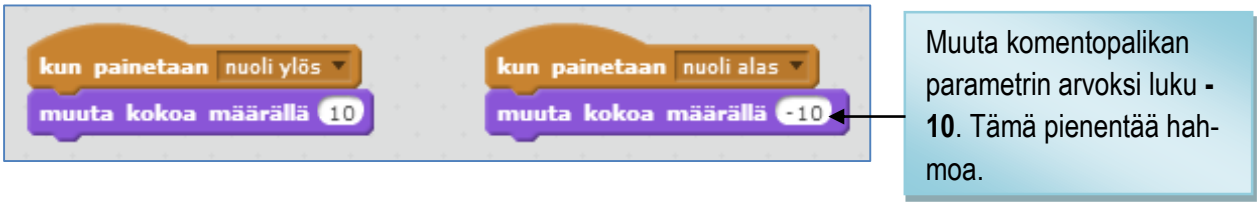
Tapahtumapalikoissa ”**kun painetaan <välilyönti>**” reagoidaan välilyönnin painamisiin. Haluamme näiden tapahtumien reagoivan kuitenkin näppäimiin ”nuoli ylös” ja ”nuoli alas”. Näppäin, johon tapahtuma reagoi, voidaan vaihtaa valitsemalla haluttu näppäin valintalistasta. Vaihdetaan näihin näppäimiksi ”nuoli ylös” ja ”nuoli alas”.



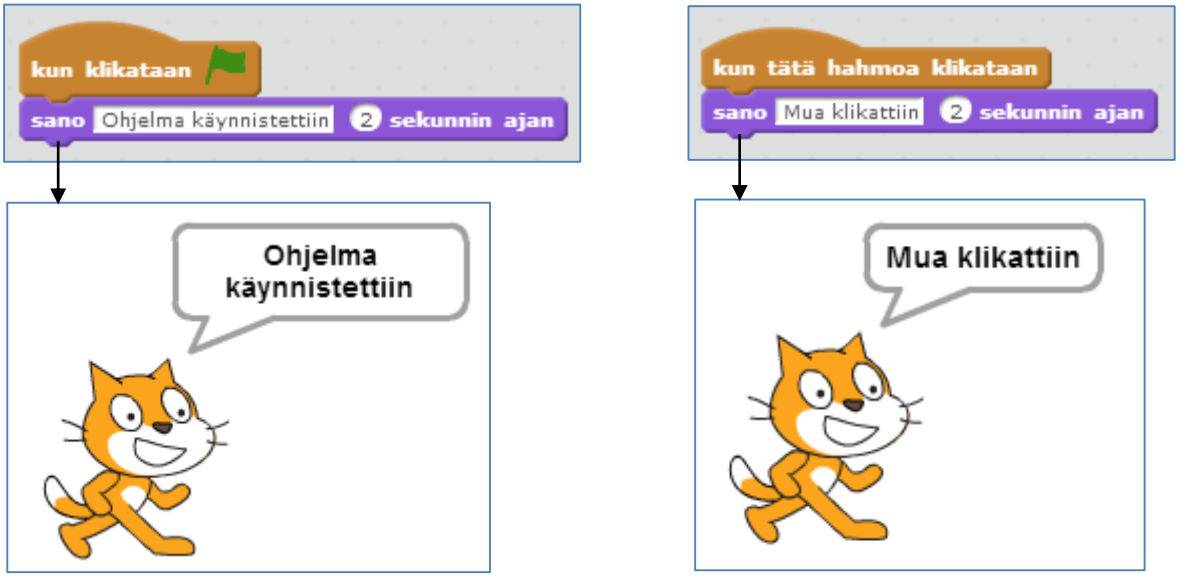

Nyt meillä on tapahtumat, mutta niissä ei ole vielä yhtään suoritettavaa koodipalikkaa, jotka tekisivät jotain. Se, miten ohjelma näihin tapahtumiin reagoi, riippuu täysin siitä, mitä koodipalikoita näiden tapahtumien alle sijoitamme. Siirry seuraavaksi ”**Ulkonäkö**”**-osioon**. Sieltä löydät komentopalikan ”**sano <hello!> <2> sekunnin ajan**”. **<hello!>** ja **<2>** ovat komentopalikan parametreja ja voit muuttaa ne sellaiseksi kuin haluat. Näitä parametreja muuttamalla voit ohjelmoida hahmon sanomaan puhekuplalla minkä tahansa tekstin ja puhekupla on näkyvissä haluamasi ajan. Raahaa tämä komentopalikka tapahtumien ”**kun klikataan vihreä lippu**” ja ”**kun tätä hahmoa klikataan**” alle seuraavan kuvan mukaisesti.


Muuta tekstin ”**Hello!**” tilalle vasemman puoleiseen palikkaan teksti ”**Ohjelma käynnistettiin**” ja oikean puoleiseen palikkaan ”**Mua klikattiin**”. Muuttaminen tehdään klikkaamalla hiirellä palikassa olevaa valkoista kohtaa, pyyhkimällä siinä oleva teksti pois ja kirjoittamalla uusi teksti.

Seuraavaksi sijoitetaan komentopalikat tapahtumiin ”**kun painetaan <nuoli ylös>**” ja ”**kun painetaan <nuoli alas>**”. Haluamme hahmon koon kasvavan ”nuoli ylös”-näppäimellä ja pienenevän ”nuoli alas”-näppäimellä. Raahaa molempien tapahtumien alle komentopalikka ”**muuta kokoa määrällä <10>**”. Arvo **10** on parametri, joka kertoo kuinka paljon kokoa halutaan muuttaa. Positiivinen luku kasvattaa kokoa ja negatiivinen pienentää sitä. Muuta tapahtuman ”**kun painetaan <nuoli alas>**” alla olevan komentopalikan parametrin arvoksi luku **-10**.




Ohjelma on nyt valmis, käynnistä se klikkaamalla ohjelmassa vihreän lipun kuvaketta. Käynnistyksen jälkeen suoritetaan tapahtumassa ”**kun klikataan vihreä lippu**” oleva komentopalikka. Ja kun klikkaat hahmoa hiirellä, suoritetaan tapahtumassa ”**kun tätä hahmoa klikataan”** oleva komentopalikka.




Ohjelma reagoi myös näppäimiin ”nuoli ylös” ja ”nuoli alas” suurentamalla ja pienentämällä hahmoa. Tapahtumaa ”**kun klikataan vihreä lippu**” käytetään hahmoissa yleisesti ohjelmakoodin suorituksen aloitustapahtumana. Muista, että jokaisella hahmolla on oma itsenäinen koodinsa ja jokaisen hahmon koodin suoritus tulee aloittaa jollain tapahtumalla. Yhdellä hahmolla voi olla myös useita eri tapahtumia ja niiden aktivoitumisen johdosta suoritettavia skriptejä.

## HAHMON LIIKEANIMAATIO

Monilla hahmoilla on kaksi asustetta, joiden vaihtaminen yhdessä samanaikaisen liikuttamisen kanssa muodostaa liikeanimaation. Esimerkiksi kissahahmon asusteen vaihtaminen liikkeen yhteydessä muodostaa kävelyanimaation.

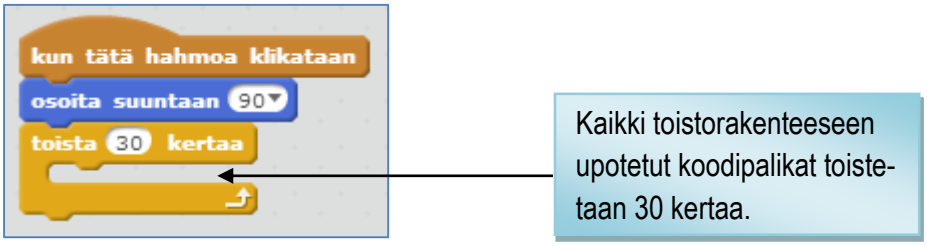
Siirrä kissahahmo hiirellä raahaamalla näyttämön vasempaan reunaan. Tehdään ohjelma, joka laittaa kissan kävelemään kun sen hahmoa klikataan hiirellä. Aloitamme ”**Tapahtumat**”**-osiosta** löytyvällä tapahtumalla ”**kun tätä hahmoa klikataan**”.


Klikattaessa kissahahmoa hiirellä, se lähtee kävelemään näyttämön oikeaa reunaa kohti, kääntyy ja palaa takaisin samaan kohtaan. Lopuksi kissa sanoo puhekuplalla tekstin ”Lenkki tehty”.

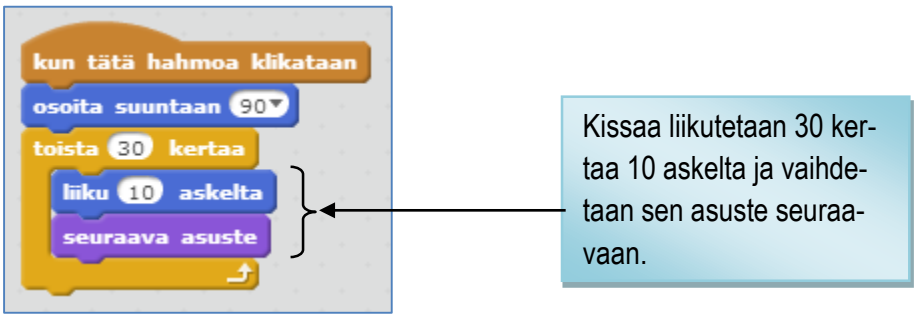
Jotta kävelyanimaatiosta tulisi mielekäs, kissaa tulee liikuttaa pienin askelin ja vaihtaa asustetta jokaisella liikutuskerralla. Tämän toteuttamiseksi tarvitsemme toistorakennetta. Voimme esimerkiksi liikuttaa kissa kerralla 10 askelta, vaihtaa asuste ja toistaa molemmat toimenpiteet 30 kertaa.

Asetetaan alkusi kissan liikesuunta oikealle, komentopalikka ”**osoita suuntaan <90>**” löytyy ”**Liike**”**-osiosta**.

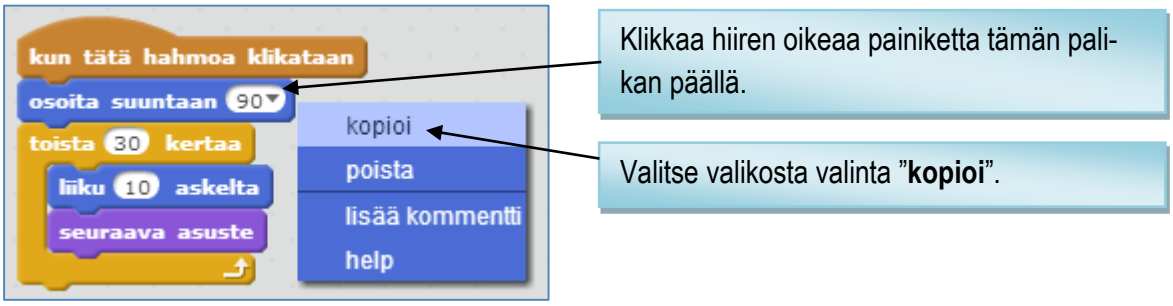

Seuraavaksi tarvitsemme toistorakenteen, löydät sen **Ohjaus-osiosta**. Tiedämme nyt etukäteen kuinka monta toistoa haluamme, joten käytämme toistorakennetta ”**toista <10> kertaa**” muuttamalla toistojen lukumäärän arvoon **30**.



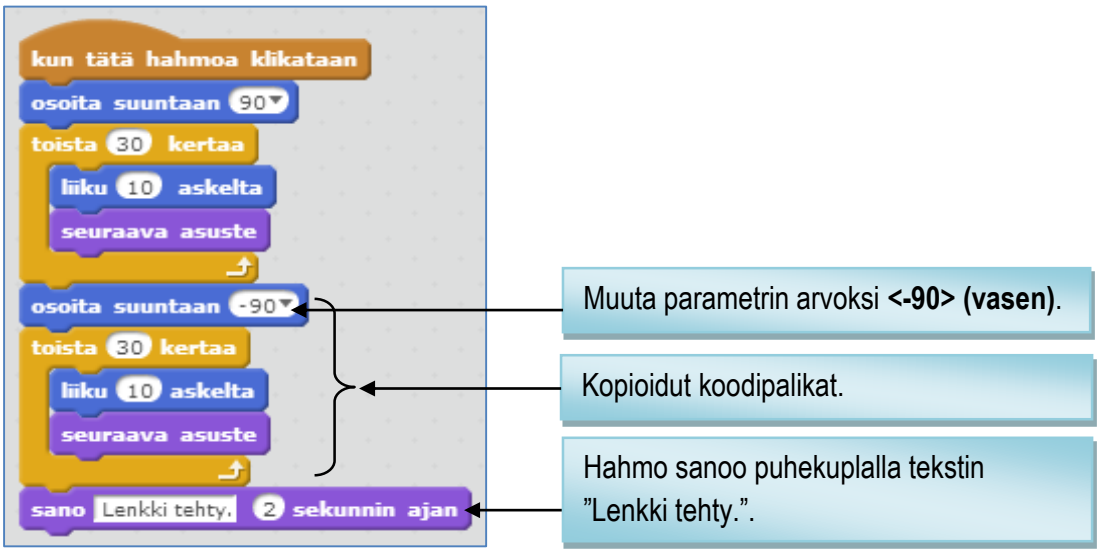
Upotetaan seuraavaksi tarvittavat komentopalikat toistorakenteeseen. Palikka ”**liiku <10> askelta**” löytyy **Liike-osiosta** ja palikka ”**seuraava asuste”** puolestaan **Ulkonäkö-osiosta**.



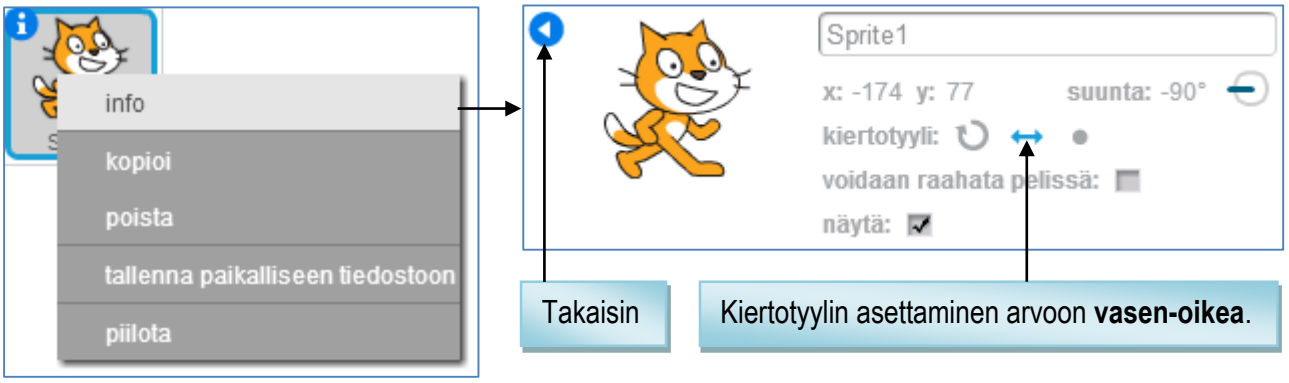

Toistorakenteen suoritus siirtää hahmoa yhteensä 300 askelta oikealle. Vasemmalle päin liikuttaminen on hyvin samanlaista, joten voit kopioida tähän tarvittavat palikat. Siirrä hiiriosoitin palikan ”**osoita suuntaan <90>**” päälle, klikkaa hiiren oikeaa painiketta ja valitse valikosta valinta ”**kopioi**”.




Liitä kopioidut palikat aiempien perään ja muuta ainoastaan palikan ”**osoita suuntaan <90>**” parametrin arvoksi **<-90> (vasemmalle**). Lisää viimeisen toistorakenteen alle ”**Ulkonäkö**”**-osiosta** löytyvä komentopalikka ”**sano <teksti> <2> sekunnin ajan**” ja vaihda tekstiksi ”Lenkki tehty.”.



Ohjelma on nyt valmis. Käynnistä se klikkaamalla kissahahmoa hiirellä. Kissa lähtee kävelemään oikealle, kääntyy, palaa takaisin ja sanoo ”Lenkki tehty”. Huomaat kuitenkin, että kävellessään vasemmalle kissa kääntyy ylösalaisin. Voit estää tämän asettamalla hahmon **kiertotyyliksi** valinnan ”**vasen-oikea**”. Tämä tehdään **Hahmot-ikkunassa**. Klikkaa hahmon kuvakkeen päällä hiiren oikeaa painiketta ja valitse valikosta valinta ”**Info**”. Kohdassa ”**kiertotyyli**” klikkaa kuvaketta, jossa on **vasemmalle ja oikealle osoittava nuoli**.



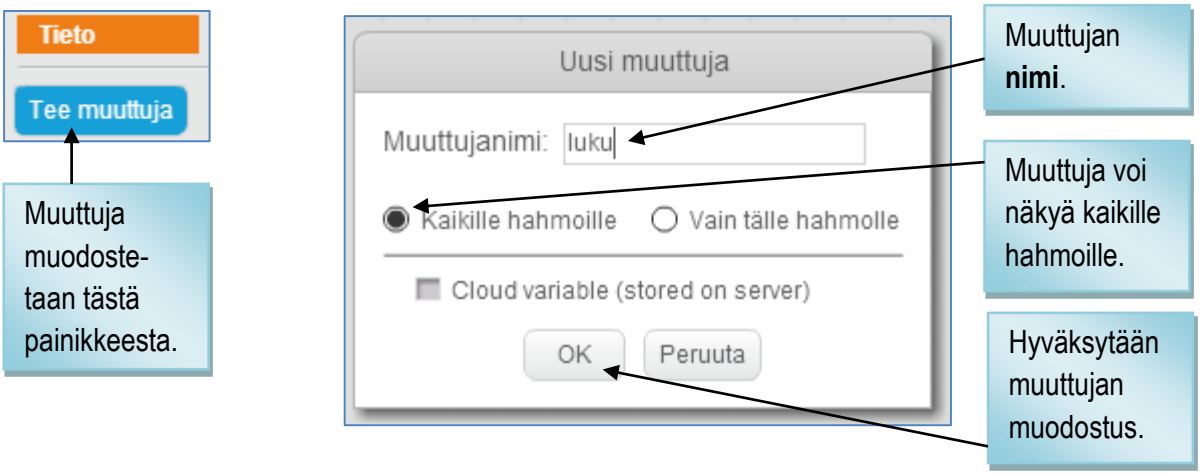

Tämän muutoksen jälkeen kissa pysyy oikein päin myös vasemmalle kävellessään.

## Muuttujan käyttö ja satunnaisluku

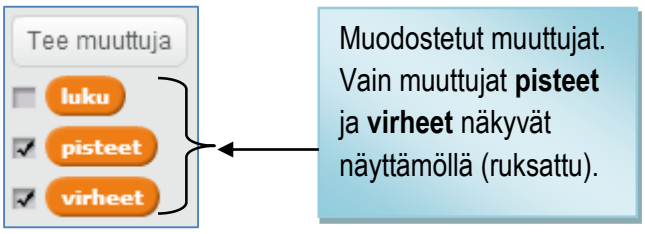
Tässä esimerkissä perehdytään muuttujan käyttöön, satunnaisluvun arvontaan, syötteen kysymiseen ja yhtäsuuruuden tutkimiseen ehtorakenteessa. Koko ohjelman toiminta perustuu toistorakenteeseen.

Ohjelmassa kissa-hahmo kysyy ”Arvaa mitä lukua 1-5 ajattelen”. Jos käyttäjä arvaa oikein, pisteet lisääntyvät kymmenellä. Jos hän arvaa väärin, virheet lisääntyvät yhdellä. Peli päättyy kun virheitä on viisi kappaletta.

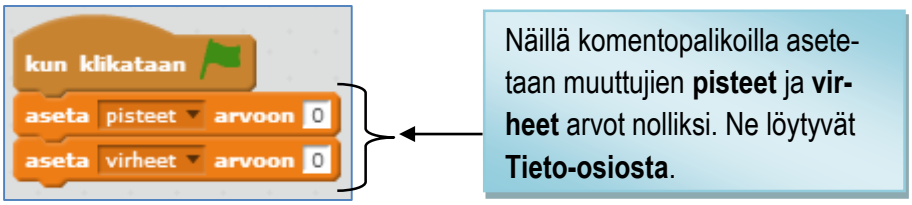
Ohjelmaan tarvitaan kolme muuttujaa **luku, pisteet** ja **virheet**. Muodosta ne **Tieto-osion** painikkeesta ”**Tee muuttuja**” ja anna muuttujille em. nimet. Muuttujat voivat olla näkyvissä kaikille hahmoille.



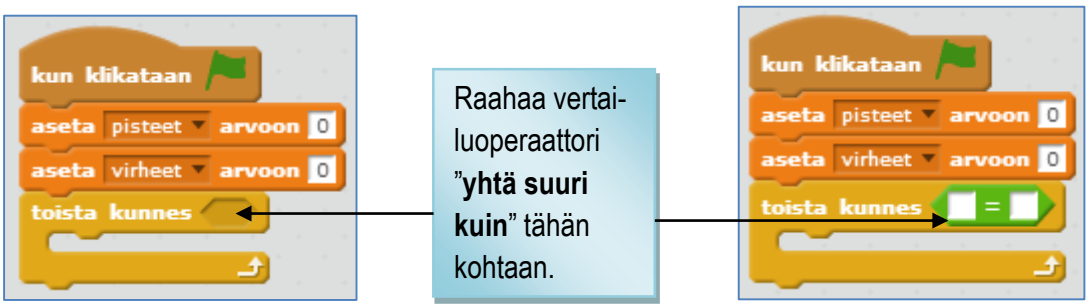

**Luku**-muuttujan arvoa ei haluta näkyviin näyttämölle, joten sen edestä poistetaan ruksi. Sen sijaan muuttujien **pisteet** ja **virheet** arvojen tulee näkyä näyttämöllä.



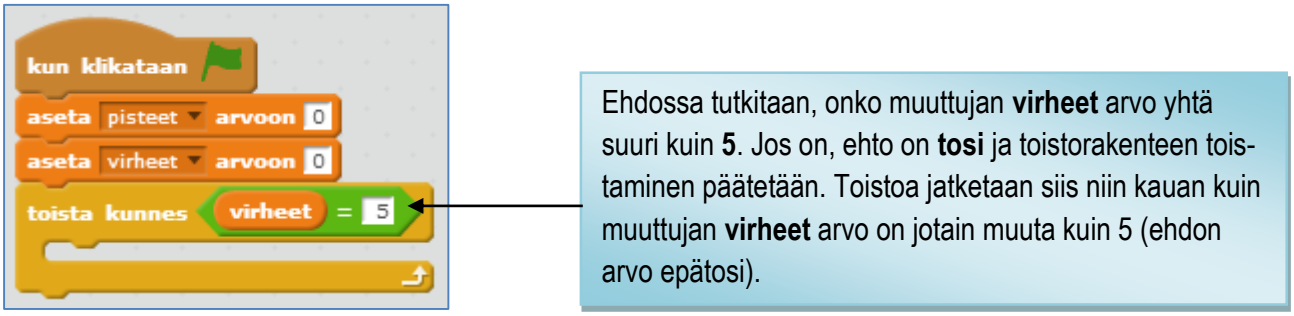

Heti käynnistyksen jälkeen koodissa asetetaan muuttujien **pisteet** ja **virheet** arvoksi nolla.



Seuraavaksi tarvitsemme toistorakennetta. Nyt emme voi tietää etukäteen, kuinka monta toistoa tarvitaan, sillä ohjelman on esitettävä käyttäjälle kysymyksiä niin kauan, kunnes käyttäjä on arvannut viisi kertaa väärin. Käytämme **Ohjaus-osiosta** löytyvää toistorakennetta ”**toista kunnes <ehto>**”. Tähän toistorakenteeseen voimme asettaa ehdon, joka rajoittaa toistojen lukumäärän. Sanallisesti ehtomme kuuluu ”**toista kunnes virheet = 5**”. Tarvitsemme ehdossa vertailuoperaattoria ”**yhtä suuri kuin**” (löytyy **Toiminnot-osiosta**) ja muuttujan **virheet** arvoa (löytyy **Tieto-osiosta**). Rahaa ensin kuitenkin toistorakenteen koodipalikka edellisten perään. Raahaa tämän jälkeen vertailuoperaattori ”**yhtä suuri kuin**” toistorakenteen ehtoon.



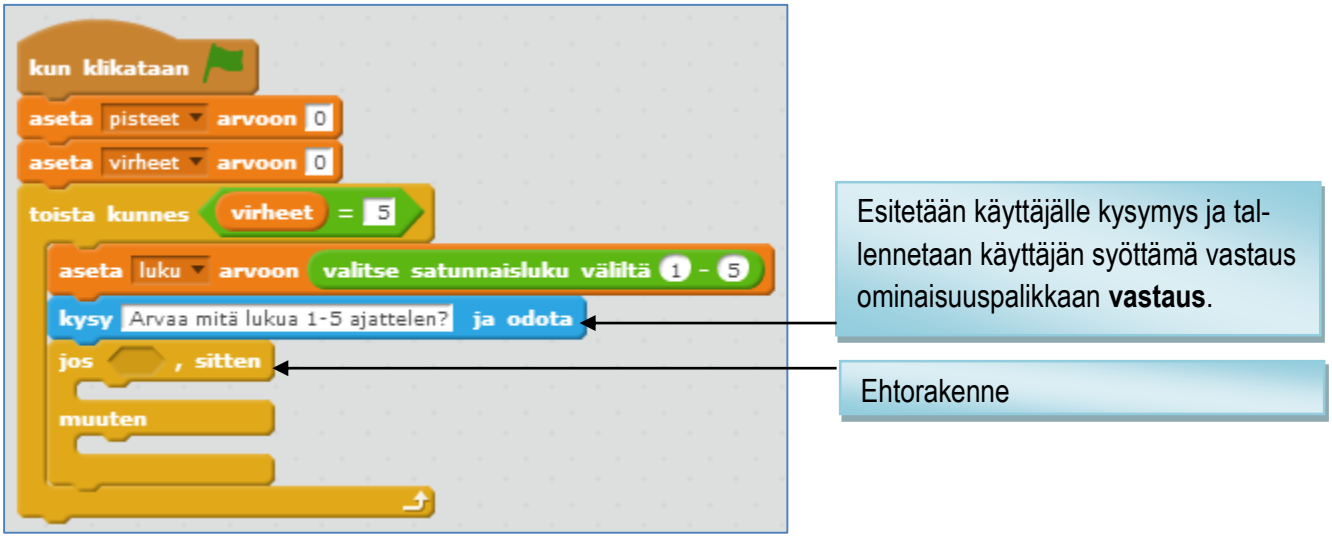

Raahaa **Tieto-osiosta** muuttujapalikka **virheet** vertailuoperaattorin ensimmäiseksi parametriksi ja syötä toiseksi parametriksi luku 5.




Seuraavaksi arvotaan satunnaisluku väliltä 1-5 ja sijoitetaan arvottu luku muuttujan **luku** arvoksi.

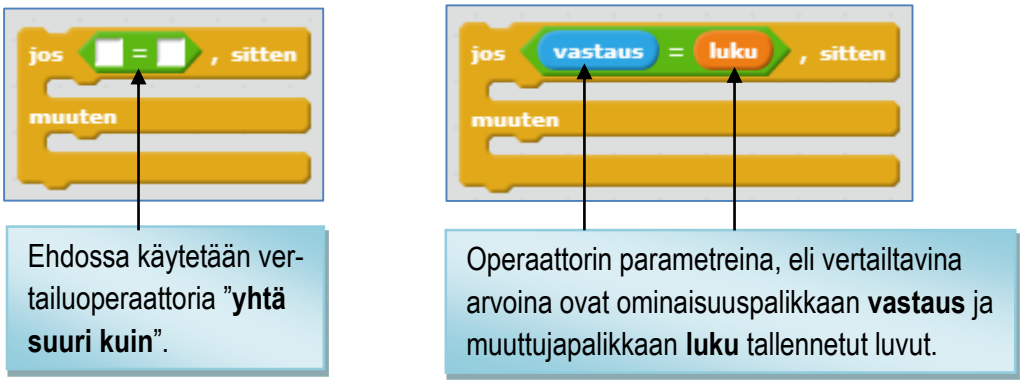
Sitten kysytään käyttäjältä ”Arvaa mitä lukua 1-5 ajattelen?” Käyttäjän tulee voida syöttää oma arvauksensa näppäimistöltä, joten käytämme tässä **Tuntoaisti-osiosta** löytyvää palikkaa ”**kysy <kysymys> ja odota**”. Tämä palikka esittää kysymyksen hahmon puhekuplassa ja näyttämön alareunaa tulee tekstikenttä, johon käyttäjä voi syöttää vastauksensa. Vastaus on luettavissa ominaisuuspalikasta **vastaus**. Käyttäjän vastauksen jälkeen tutkitaan, oliko vastaus yhtä suuri kuin arvottu luku. Jos oli, pisteet lisääntyvät kymmenellä. Jos ei ollut, virheet lisääntyvät yhdellä ja hahmo kertoo puhekuplalla ajattelemansa luvun.

Nyt tarvitsemme ehtorakennetta, jossa tutkitaan arvotun luvun ja käyttäjän syöttämän luvun yhtäsuuruutta vertailuoperaattorilla. Käytetään ehtorakennetta ”**jos <ehto>, sitten – muuten**”. Tämä koodipalikka löytyy **Ohjaus-osiosta**.

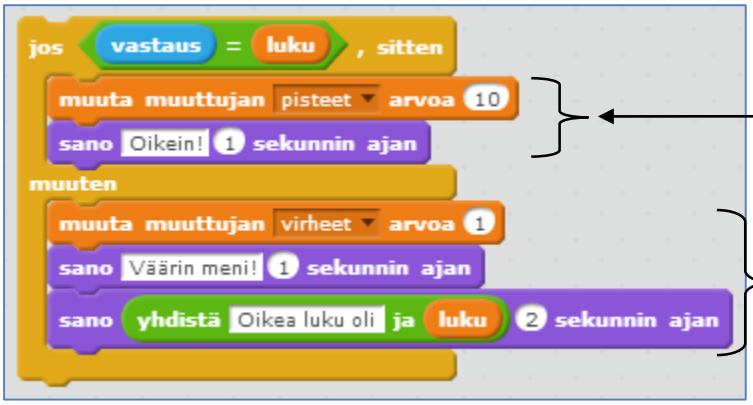



Ehtorakenteen ehdossa tutkitaan, onko ominaisuuden **vastaus** arvo yhtä suuri kuin muuttujan **luku** arvo. Jos yhtäsuuruus toteutuu, ehto on **tosi** ja silloin suoritetaan ehtorakenteen **sitten-osa**. Jos ehto on **epätosi**, suoritetaan vaihtoehtoinen **muuten-osa**.

Sijoita ehtoon ensin vertailuoperaattori ”**yhtä suuri kuin**”. Lisää sitten operaattorin parametreiksi ominaisuuspalikka **vastaus** ja muuttujapalikka **luku**.



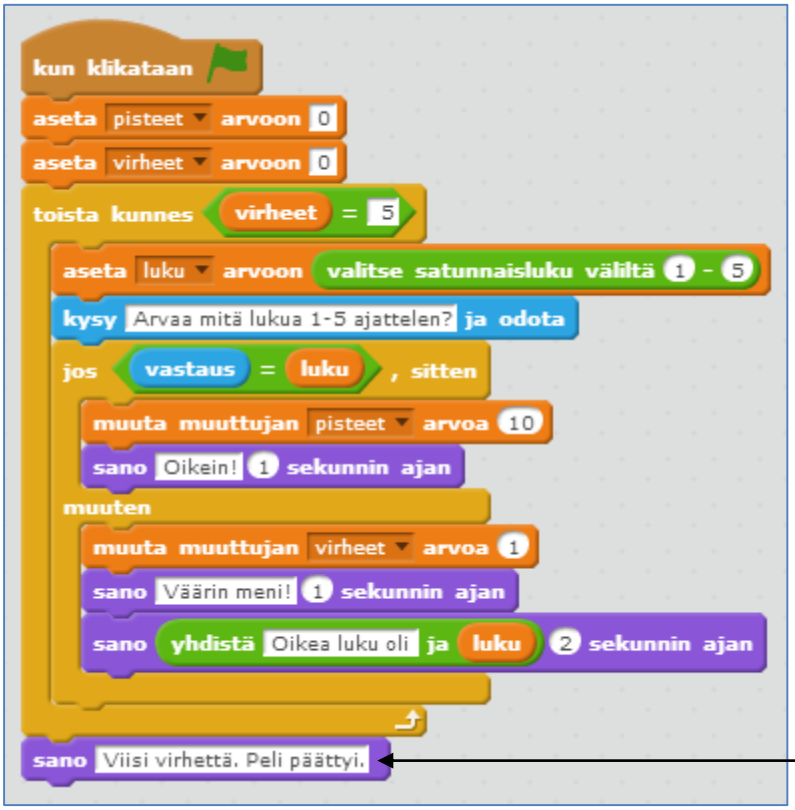
Ehtorakenteen **sitten-osa** suoritetaan aina silloin, kun ehto on **tosi**. Tässä tilanteessa se tarkoittaa sitä, että käyttäjä on arvannut luvun oikein. Tällöin kasvatetaan muuttujan **pisteet** arvoa kymmenellä ja näytetään puhekuplassa teksti ”Oikein!”. Ehtorakenteen vaihtoehtoinen **muuten-osa** suoritetaan aina silloin, kun ehto on **epätosi**. Tässä tilanteessa se tarkoittaa sitä, että käyttäjä on arvannut luvun väärin. Tällöin kasvatetaan muuttujan **virheet** arvoa yhdellä, sanotaan puhekuplalla teksti ”Väärin meni!” ja sanotaan vielä mikä oli oikea luku.




Ehtorakenteen **sitten-osassa** kasvatetaan muuttujan **pisteet** arvoa kymmenellä ja sanotaan teksti ”Oikein!”

Ehtorakenteen **muuten-osassa** kasvatetaan virheiden määrää yhdellä, sanotaan teksti ”Väärin meni!” ja lopuksi sanotaan vielä oikea luku.

Ohjelman kaikki koodipalikat on esitetty alla olevassa kuvassa.



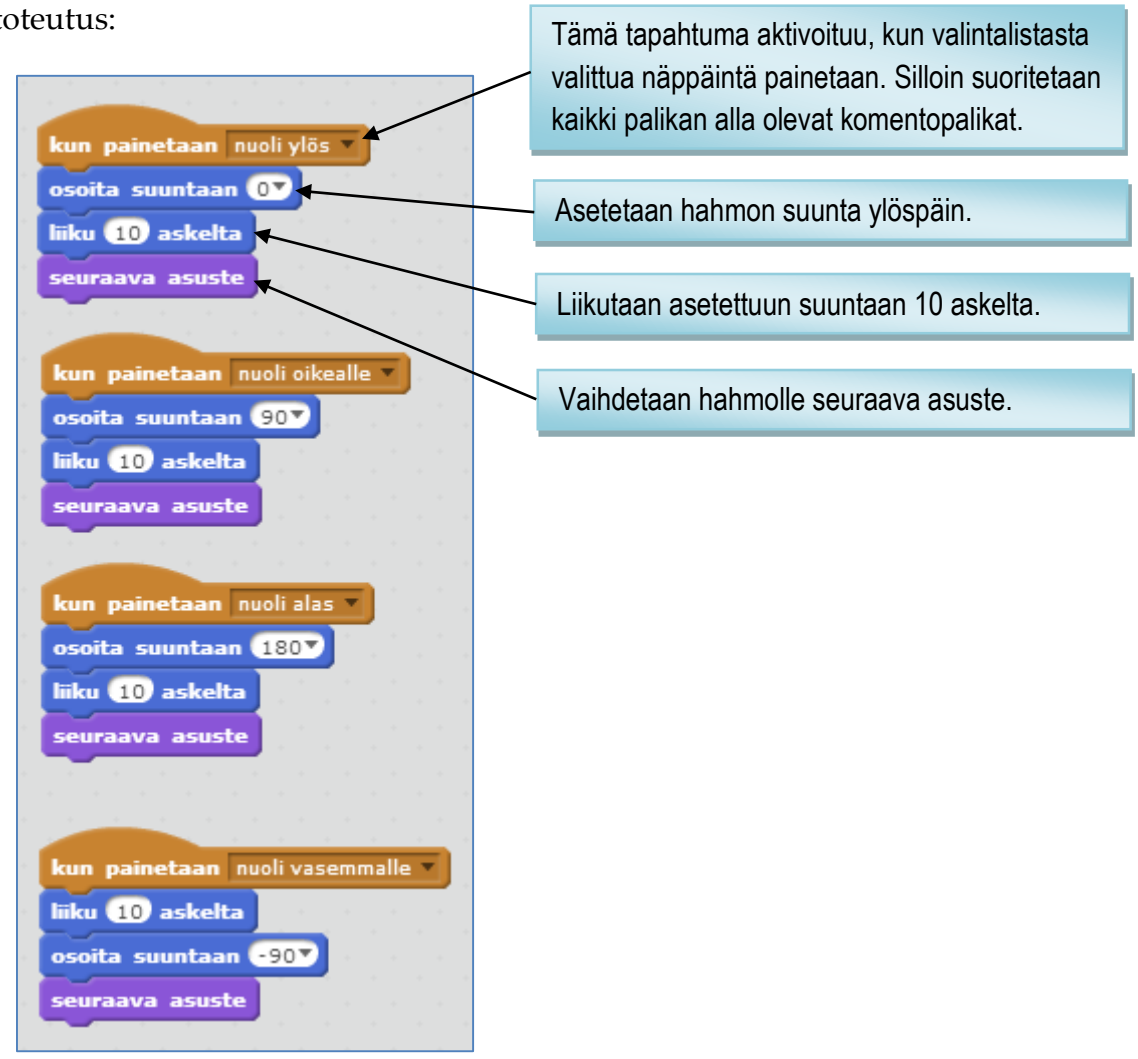

Toistorakenteen sisällä olevia koodipalikoita toistetaan niin kauan, kunnes muuttujan **virheet** arvo on 5. Sen jälkeen ohjelman suoritus siirtyy tähän palikkaan.

## Hahmon liikuttaminen nuolinäppäimillä

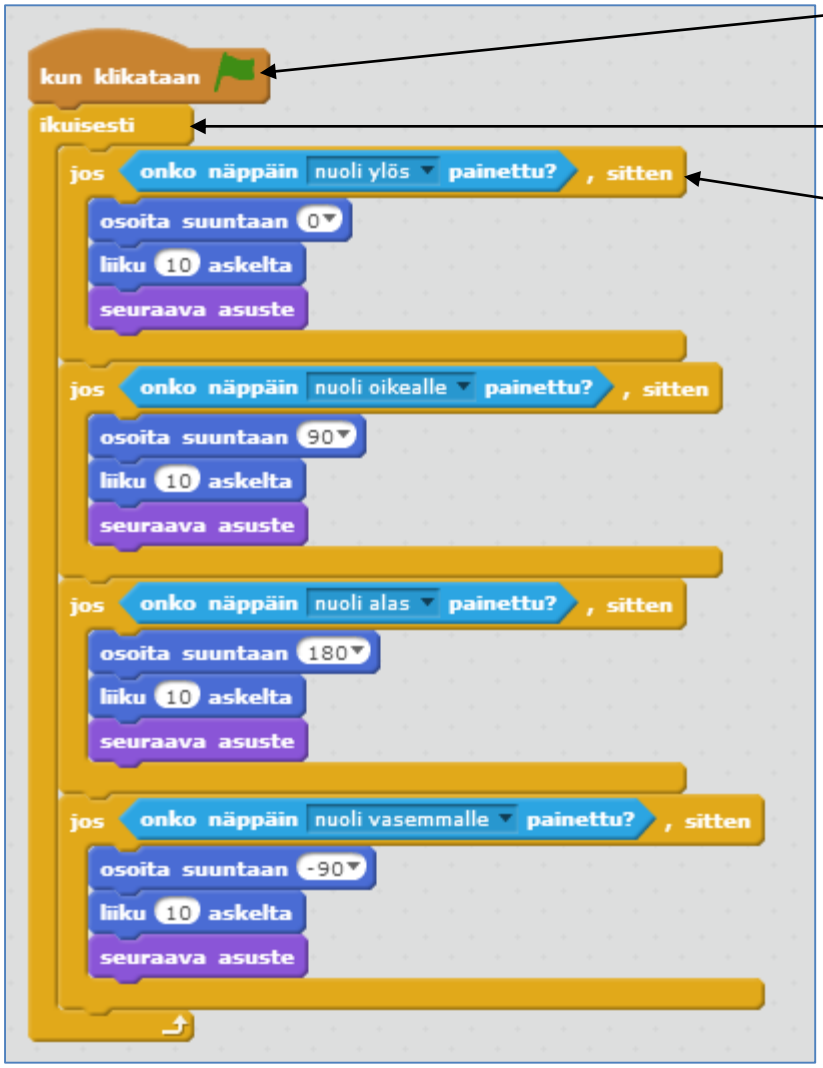
Ohjelman suunnittelu:

Näyttämöllä oleva kissa-hahmo liikkuu käyttäjän ohjaamana. Kissaa ohjataan nuolinäppäimillä haluttuun suuntaan. Liikkuessaan kissa vaihtaa asustetta, jolloin muodostuu kävelyä muistuttava animaatio.

Ohjelman toteutus:




Ohjelman samanlaiseen toimintaan voidaan päästä useilla erilaisilla toteutuksilla. Esimerkiksi nopeampi reagointi näppäinten painalluksiin voidaan toteuttaa käyttämällä totuusarvon palauttavaa palikkaa ”**onko näppäintä <näppäin> painettu**” Seuraavalla sivulla näet, miten samanlainen toiminta voidaan toteuttaa toisin.

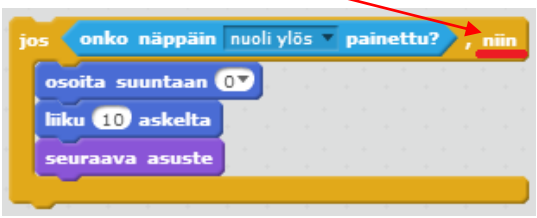



Tapahtuma aktivoituu kun ohjelman suoritus käynnistetään vihreästä lipusta.

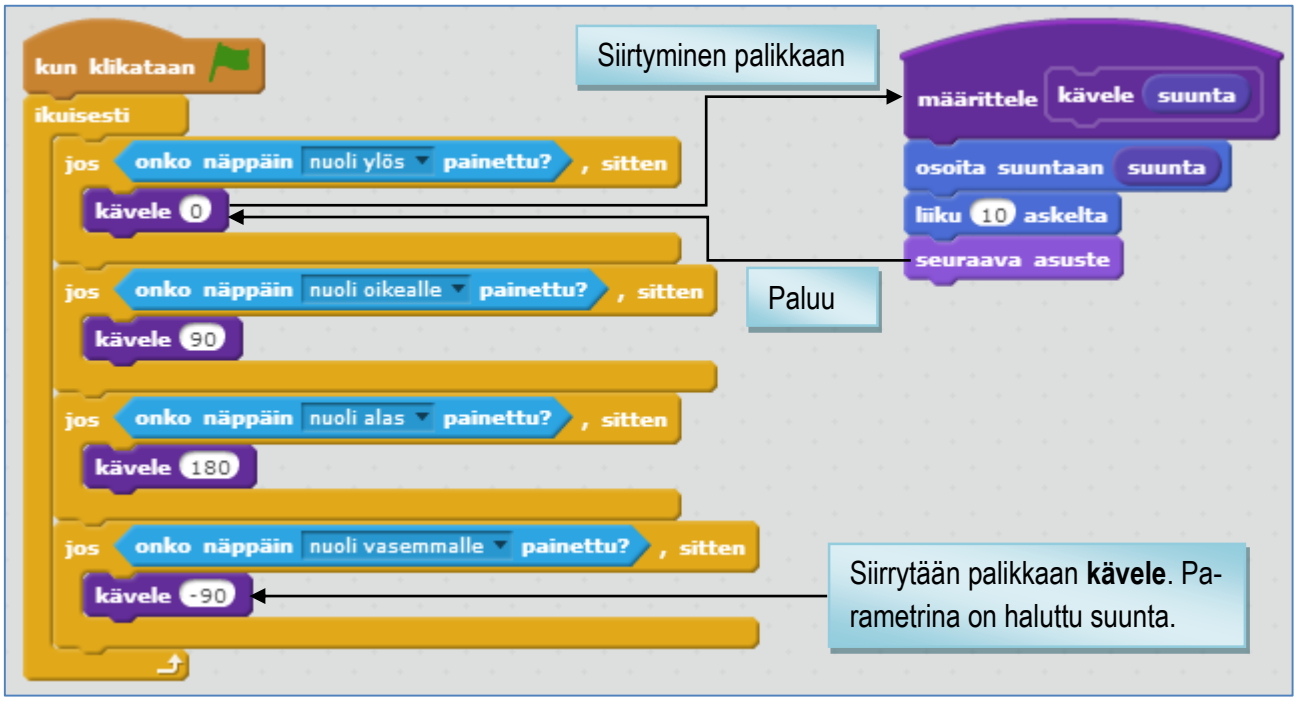
Ikuinen silmukka.

Ehtorakenteen, ehdossa tutkitaan onko näppäintä ”nuoli ylös” painettu. Jos on, ehto on **tosi** ja ehtorakenteen **sitten-osassa** olevat komentopalikat suoritetaan.

**Huom. Scratchin nykyisessä versiossa sitten-osan nimi on niin-osa.** Ehtorakenne on kuitenkin muuten täysin samanlainen.




Kuten huomaat, koodissa toistetaan aika paljon samoja palikoita. Voimme yksinkertaistaa ohjelmaa sijoittamalla toistettavat komentopalikat omaan palikkaan (aliohjelmaan) ja viemällä hahmon suunta parametrina palikkaan.



Kissa liikkuu nyt iloisesti näyttämöllä käyttäjän ohjaamana. Reunaan osuessaan se kuitenkin häviää melkein kokonaan näkyvistä. Lisätään **kävele**-palikkaan koodi, joka pompauttaa kissan reunasta takaisin näyttämölle, jos kissa koskettaa reunaa.


Ehdossa käytetään totuusarvopalikkaa ”**koskettaako <reuna>**”. Se palauttaa arvon **tosi**, jos hahmo koskettaa reunaa, muutoin paluuarvo on **epätos**i. Ehtorakenteeseen upotettu koodi suoritetaan vain silloin, kun ehto on **tosi**.

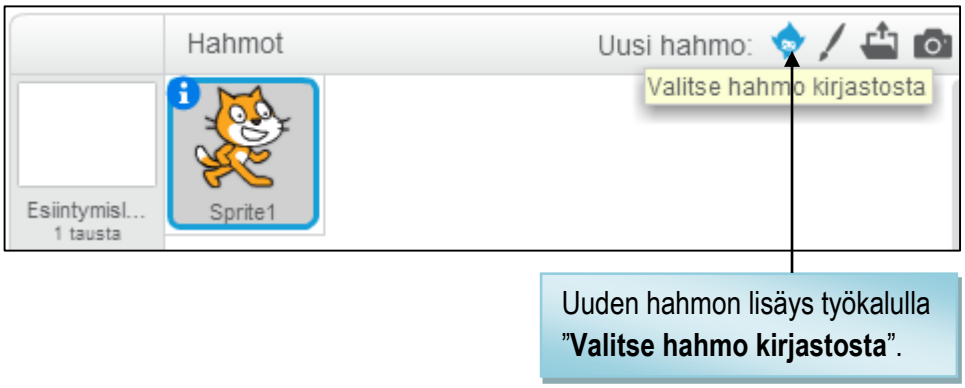
## Toisiaan koskettavat hahmot

Ohjelman suunnittelu:

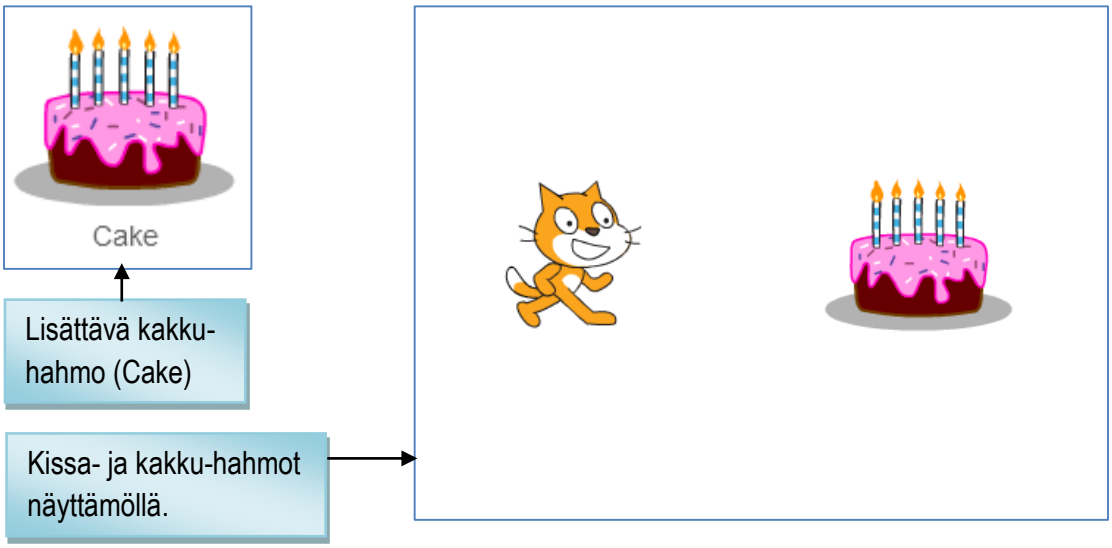
Käyttäjä ohjaa näyttämällä olevaa kissa-hahmoa nuolinäppäimillä haluamaansa suuntaan. Monissa ohjelmissa ja varsinkin peleissä on tarpeellista tietää, koskettavatko kaksi hahmoa toisiaan. Toisena hahmona ohjelmassa onkin kakku (Cake) ja kun kissa-hahmo koskettaa kakku-hahmoa se häviää näkyvistä. Tällöin kissa sanoo puhekuplalla tekstin ”Nam, olipa hyvää kakkua.” ja ohjelman suoritus päättyy tähän.

Ohjelman toteutus:

Uudessa projektissa kissa-hahmo onkin jo valmiina. Lisää kakku toiseksi hahmoksi **Hahmot-ikkunasta** löytyvällä ”**Valitse hahmo kirjastosta**”**-työkalulla**.

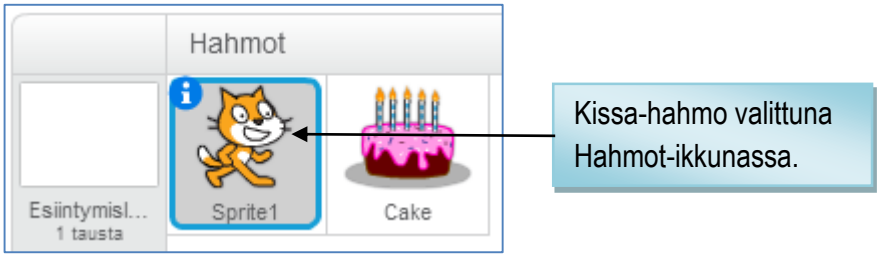



Valitse lisättäväksi hahmoksi kakku (Cake) tuplaklikkaamalla hahmon kuvaketta. Sijoita kissa- ja kakku-hahmot hiirellä raahaamalla haluamaasi kohtaan näyttämöä **niin, etteivät ne kosketa toisiaan**.



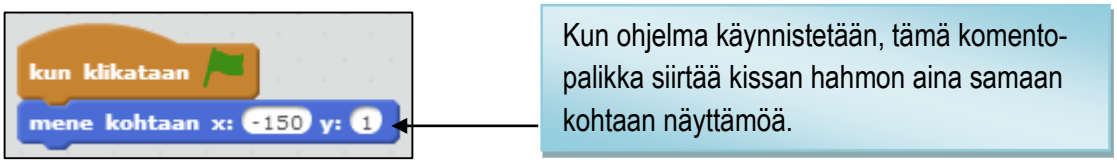
Kissa-hahmon koodipalikat:

Kissa liikuttamiseen käytettävät koodipalikat ovat sinulle jo tuttuja aiemmasta ohjelmaesimerkistä ja ne ovat pääosin samanlaisia myös tässä ohjelmassa. Klikkaa **Hahmot-ikkunassa kissa-hahmon kuvaketta**, jolloin se tulee valituksi ja pääset laatimaan sille koodipalikoita.

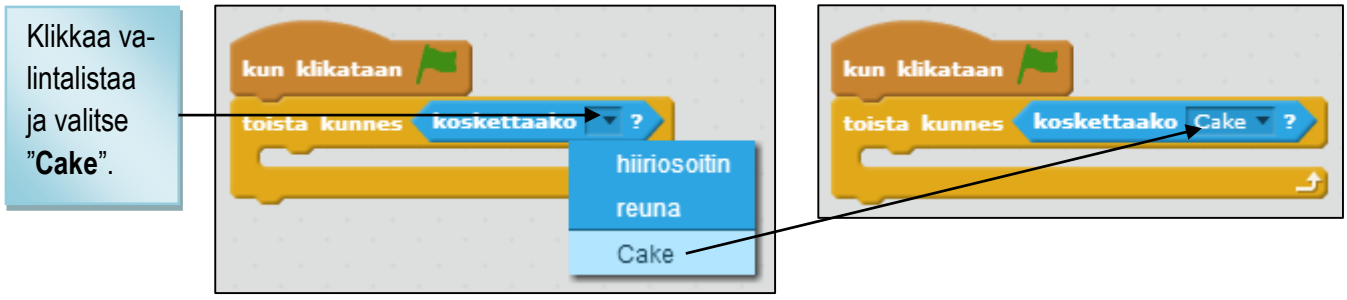



Ohjelman määrittelyssä sanotaan, että ohjelman suoritus loppuu, kun kissa koskettaa kakkua. Heti ohjelman käynnistyksen jälkeen kissa tulee siis siirtää kohtaan, jossa se ei kosketa kakkua. Tämä on tärkeää etenkin ohjelman toisella suorituskerralla, sillä ensimmäisen suorituskerran jälkeen kissa on jäänyt paikkaan, jossa se koskettaa kakkua. Tällöin ohjelman suoritus päättyisi heti käynnistyksen jälkeen.

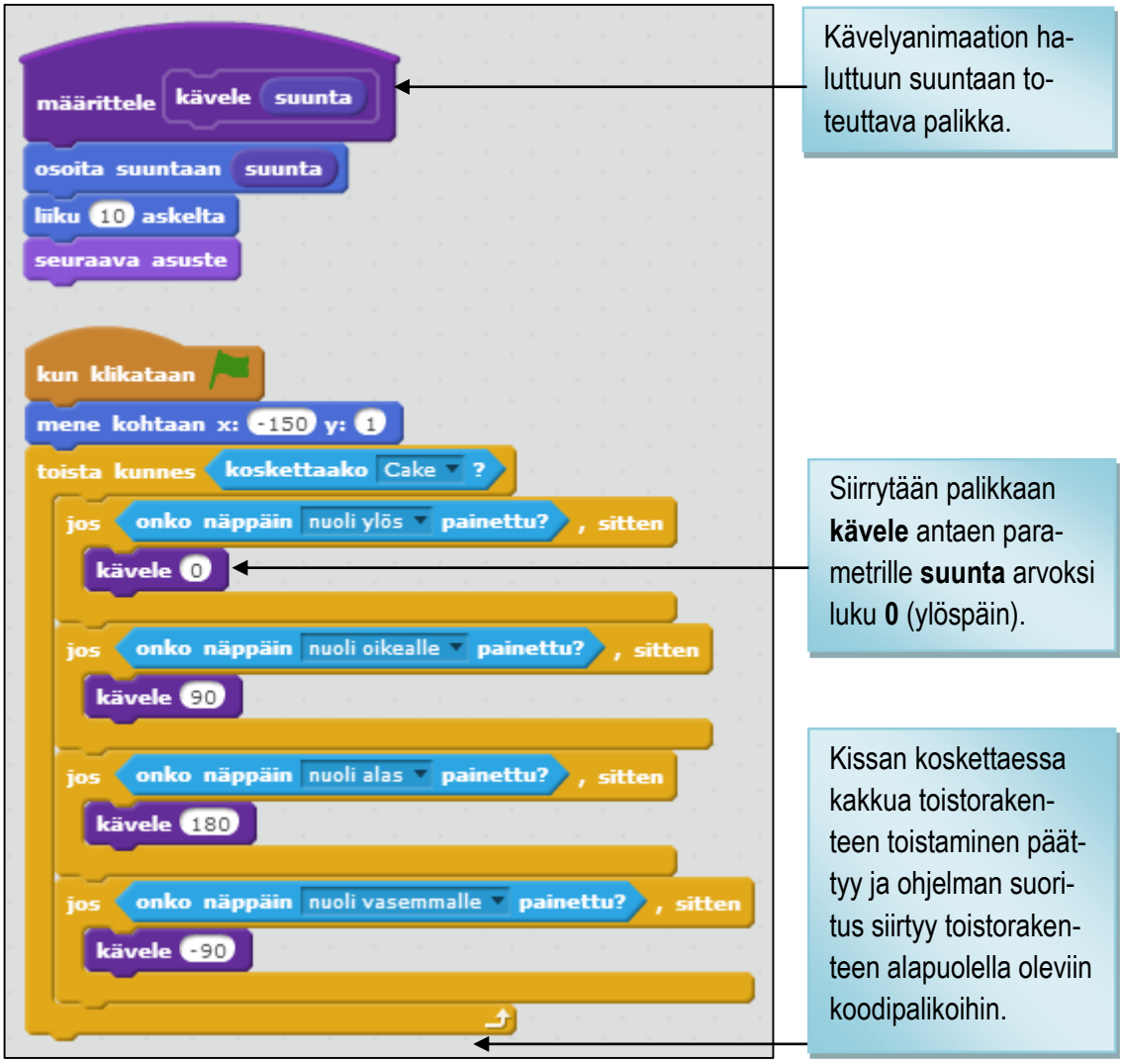
Kissa-hahmon paikka asetetaan **Liike-osiosta** löytyvällä komentopalikalla ”**mene kohtaa <x> <y>**”. Komentopalikan parametreina toimivat x- ja y-koordinaatit osoittavat kohtaan, jossa kissa tällä hetkellä on. Vedä tämä komentopalikka heti aluksi koodialueelle ja liitä sen yläpuolelle tapahtuma ”**kun klikataan vihreä lippu**”.




Nyt tarvitsemme toistorakenteen, jota toistetaan niin kauan, kunnes kissan ja kakun hahmot koskettavat toisiaan. Raahaa **Ohjaus-osiosta** toistorakenne ”**toista kunnes <ehto>**” edellisen palikan alle. Toistorakenteen ehdossa käytetään totuusarvopalikkaa ”**koskettaako <kohde>?**”. Tämä palikka löytyy **Tuntoaisti-osiosta**. Raahaa palikka sieltä toistorakenteen ehdoksi ja valitse valintalistasta valinta ”**Cake**”. Tämä tarkoittaa, että totuusarvopalikka palauttaa arvon **tosi** vain silloin, kun kissa koskettaa kakkua. Muutoin sen palauttama arvo on aina **epätosi**. Toistorakenteen toistaminen päättyy, kun ehto saa arvon **tosi**.




Kissan hahmon liikuttaminen on täysin samanlaista kuin aiemmassakin ohjelmaesimerkissä perustuen ”**kävele <suunta>**” –palikan (aliohjelman) käyttöön.




Kissa- ja kakku-hahmojen koskettaminen voidaan huomata kummankin hahmon koodissa, mutta kissan koodissa se on meillä jo valmiina toistorakenteen

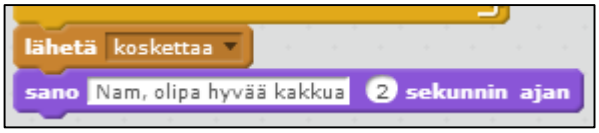
lopetusehtona. Kun siis ohjelman suoritus siirtyy toistorakenteen jälkeisiin koodipalikoihin, tiedetään kosketuksen tapahtuneen. Tällöin voimme kertoa kakku-hahmolle tapahtuneesta kosketuksesta.

Jokaisella hahmolla ja myös taustalla on itsenäinen ohjelmakoodi, joka toimii muiden hahmojen koodeista riippumatta. Miten kissa-hahmo voi siis kertoa kakku-hahmolle, että ne koskettavat toisiaan? Tietojen vaihto onnistuu esimerkiksi kaikille hahmoille näkyvien muuttujien kautta, mutta tällaisiin tilanteisiin on olemassa parempikin ratkaisu, nimittäin **viestin** lähetys ja vastaanotto. Viestien lähetys- ja vastaanotto on erityisen kätevä menetelmä hahmojen sekä hahmojen ja taustan väliseen kommunikaatioon.

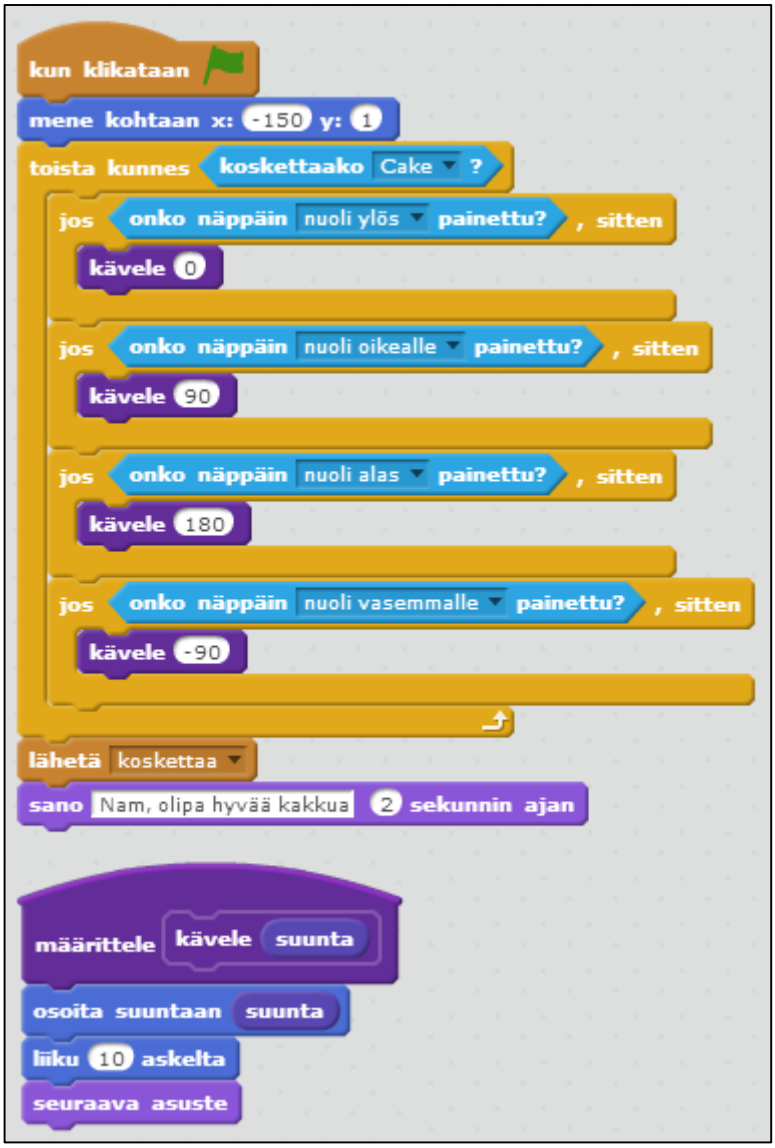
Seuraavaksi lisäämme kissa-hahmolle komentopalikan, joka lähettää kakku-hahmolle viestin. Viesti kertoo, että hahmot koskettavat toisiaan. Raahaa **Tapahtumat-osiosta** komentopalikka ”**lähetä <message1>**” toistorakenteen alapuolelle. Klikkaa valintalista auki ja valitse valinta ”**uusi viesti**”.


Tämän jälkeen pääset antamaan viestille **nimen**. Jokaisella viestillä on oma yksilöllinen nimensä, joka erottaa ne toisista viesteistä. Anna tälle viestille nimeksi **koskettaa** ja klikkaa painiketta OK.


Lisää vielä lopuksi kissa-hahmon koodiin puhekuplan tuottava komentopalikka.

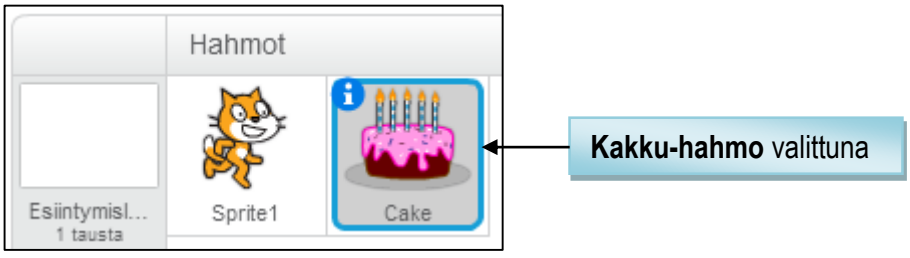


Kissa-hahmon koodi muodostuu kokonaisuudessaan alla olevista koodipalikoista.

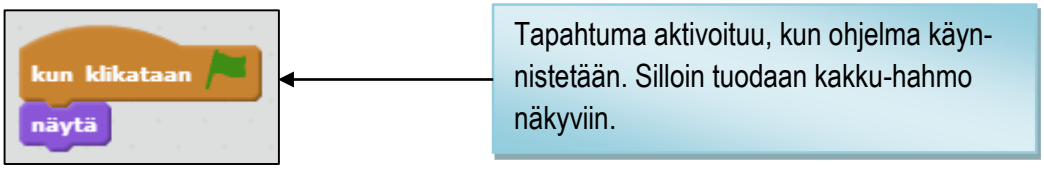



Kakku-hahmon koodipalikat:

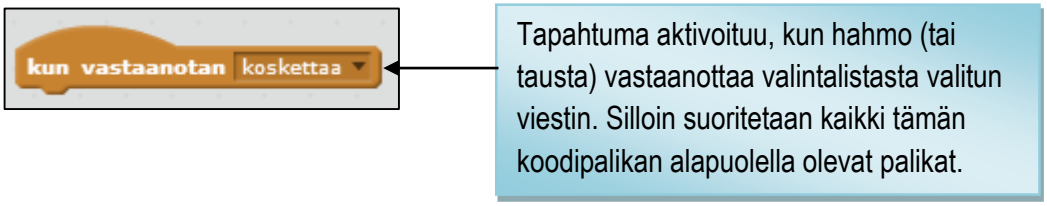
Valitse **Hahmot-ikkunasta kakku-hahmo** päästäksesi rakentamaan sille koodipalikoita.



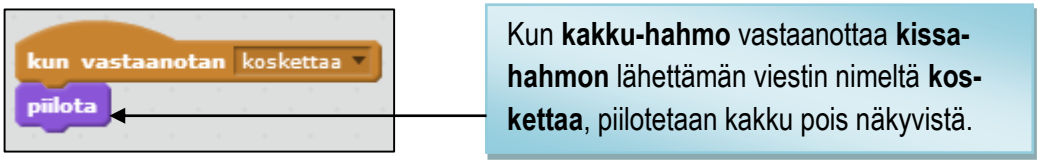
Lisää kakku-hahmon koodiin ensimmäiseksi tapahtumapalikka ”**kun klikataan vihreä lippu**”, joka ”nappaa” ohjelman käynnistystapahtuman. Lisää sen alle komentopalikka ”**näytä**”, joka löytyy **Ulkonäkö-osiosta**.




Viestin vastaanotto on myös tapahtuma ja **Tapahtumat-osiosta** löydät tapahtumapalikan ”**kun vastaanotan <koskettaa>**”. Tämä tapahtuma aktivoituu aina silloin, kun hahmo vastaanottaa valintalistasta valitun viestin. Ohjelmassa voi olla useita viestejä, mutta tässä meillä on vain yksi ja sen vuoksi viesti **koskettaa** on oletuksena valittuna.

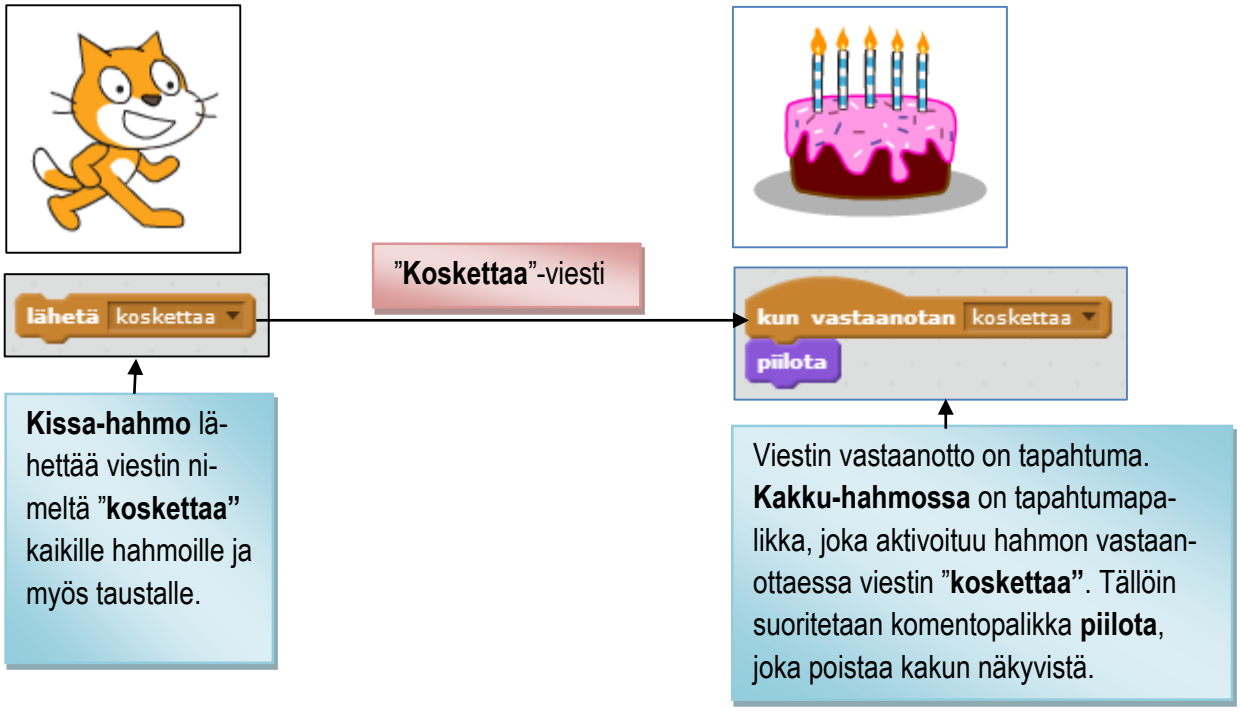



Ohjelman määrittelyssä totesimme, että kakku häipyy pois näkyvistä kissan koskettaessa sitä. Tämä tehdään käyttämällä **Ulkonäkö-osiosta** löytyvää komentopalikkaa ”**Piilota**”.




Alla on esitetty kaikki kakku-hahmon koodipalikat.

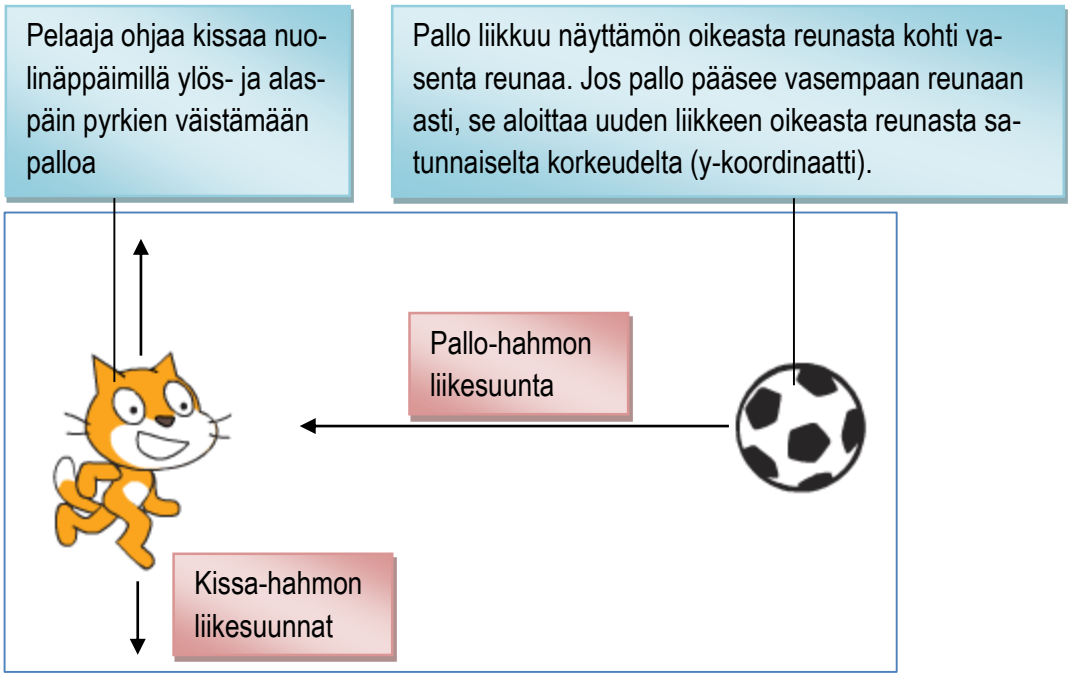
Katsotaan vielä tarkemmin viestin lähetykseen ja vastaanottamiseen perustuvaa kommunikaatiota hahmojen välillä.



# Väistelypeli

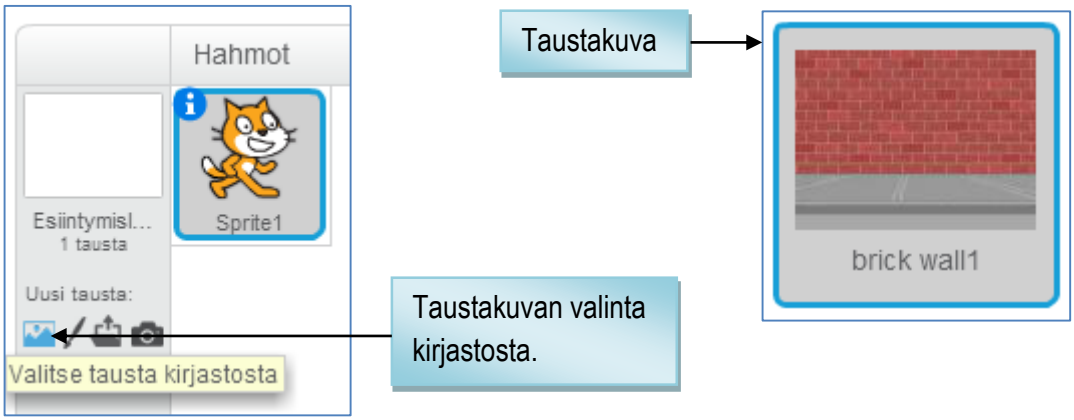
Ohjelman suunnittelu:

Näyttämöllä oleva kissa-hahmo liikkuu taustakuvan päällä nuolinäppäimillä ylös (”nuoli ylös”) ja alas (”nuoli alas”). Kissa on näyttämön vasemmassa reunassa. Näyttämön oikeassa reunassa on jalkapallo, joka lähtee pelin käynnistämisen jälkeen liikkumaan kohti vasenta reunaa. Pelaajan tulee ohjata kissaa niin, ettei jalkapallo osu kissaan. Jos jalkapallo osuu kissaan, peli päättyy ja pelaajalle kerrotaan hänen pistemääränsä. Jos kissa onnistuu väistämään jalkapallon, se jatkaa matkaansa, kunnes osuu näyttämön vasempaan reunaan. Tämän jälkeen jalkapallo siirtyy takaisin näyttämön oikeaan reunaan satunnaiseen y-koordinaatin kohtaan ja se lähtee jälleen liikkumaan kohti näyttämön vasenta reunaa. Jalkapallo liikkuu siis joka kerta eri korkeudella näyttämöä. Kun pelaajan ohjaama kissa-hahmo onnistuu väistämään jalkapallon, lisätään pelaajan pisteitä 50:llä. Pisteet näkyvät näyttämön oikeassa yläreunassa. Jalkapallon liike nopeutuu aina hieman neljän liikekerran jälkeen, näin pelin vaikeusaste kasvaa pelin edetessä.

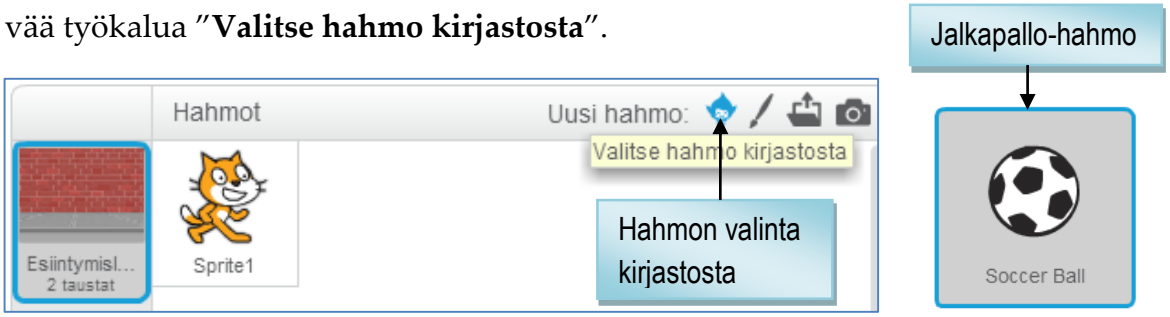


Ohjelman toteutus:

Lisää aluksi näyttämölle taustakuva käyttäen **Hahmot-ikkunan** työkalua ”**Valitse tausta kirjastosta**”. Valitse taustaksi punainen tiiliseinä (brick wall1).

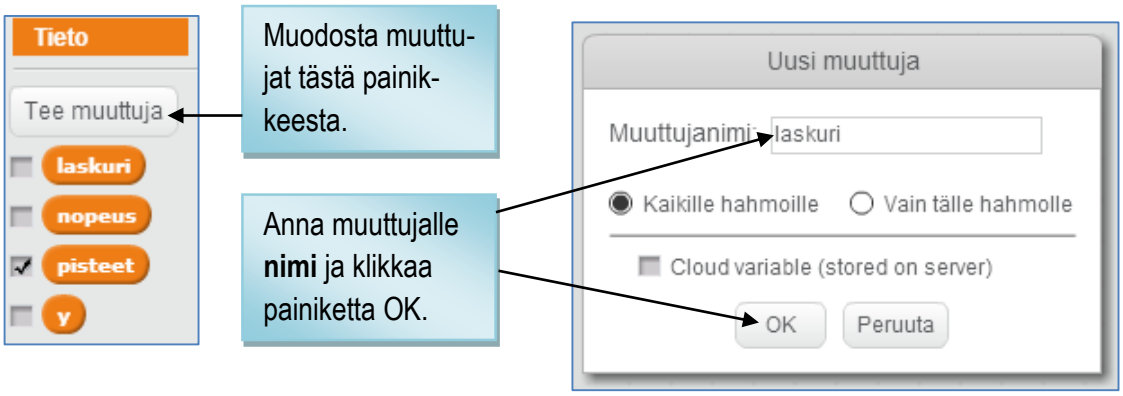



Lisää vielä jalkapallon hahmo (Soccer Ball) käyttäen **Hahmot-ikkunasta** löytyvää työkalua ”**Valitse hahmo kirjastosta**”.



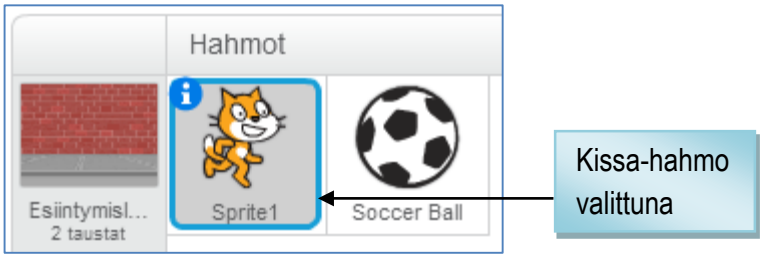

Siirrä hiirellä raahaten kissa-hahmo vaaka-suunnassa näyttämön vasempaan reunaan ja pystysuunnassa keskelle näyttämöä. Siirrä samoin jalkapallo vaaka-suunnassa ihan näyttämön oikeaan reunaan, **mutta niin ettei se kosketa näyttämön reunaa**. Pystysuunnassa jalkapallo voi olla myös näyttämön keskikohdalla. Katso mallia alla olevasta kuvasta.

Tarvitset ohjelmaan neljä muuttujaa. Mene **Tieto-osioon** ja muodosta muuttujat painikkeella ”**Tee muuttuja**”. Anna muuttujille nimiksi **laskuri**, **nopeus**, **pisteet** ja **y**. Näistä ainoastaan **pisteet**-muuttuja näytetään näyttämöllä (ruksattu).

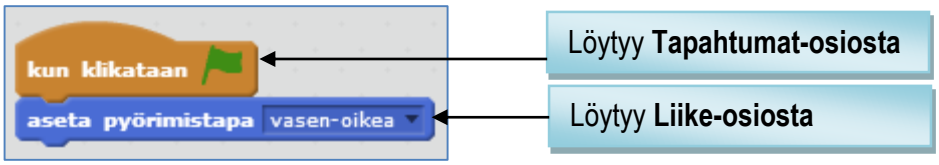



Kissa-hahmon koodipalikat:

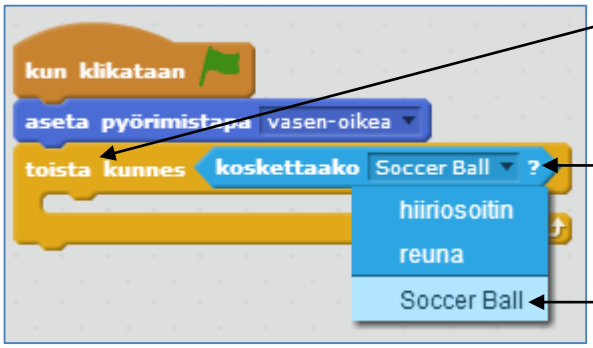
Klikkaa **Hahmot-ikkunassa kissa-hahmon kuvaketta**. Tällöin hahmo tulee valituksi ja pääset laatimaan ohjelmakoodia tälle hahmolle.




Haluamme koodin suorituksen käynnistyvän, kun pelaaja klikkaa vihreää lippua. Asetetaan hahmon kiertotyyliksi ”**vasen-oikea**”. Tällöin hahmo ei käänny ylösalaisin.



Seuraavaksi tarvitsemme toistorakenteen, jota toistetaan niin kauan, kunnes kissa koskettaa jalkapalloa. Tämä on samalla pelin päätösehto.




Tämä toistorakenne löytyy **Ohjaus-osiosta**

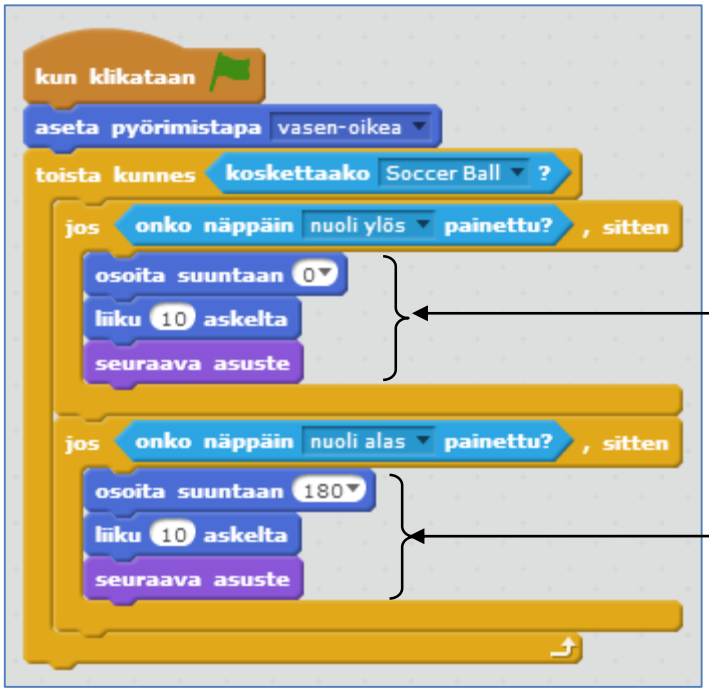
Tämä totuusarvopalikka löytyy **Tuntoaisti-osiosta** ja se palautta arvon **tosi**, kun kissa koskettaa jalkapalloa. Toistamista jatketaan siis niin kauan, kunnes kosketus tapahtuu (=peli päättyy). Valitse valintalistasta koskettuksen kohteeksi valinta **Soccer Ball**.

Lisää seuraavaksi toistorakenteen sisälle ehtorakenteet, joissa tutkitaan, onko näppäintä ”nuoli ylös” tai näppäintä ”nuoli alas” painettu.


Ehtorakenne löytyy **Ohjaus-osiosta.**

Totuusarvopalikka löytyy **Tuntoaisti-osiosta**. Valitse valintalistasta tutkittaviksi näppäimiksi näppäimet ”**nuoli ylös**” ja ”**nuoli alas**”.

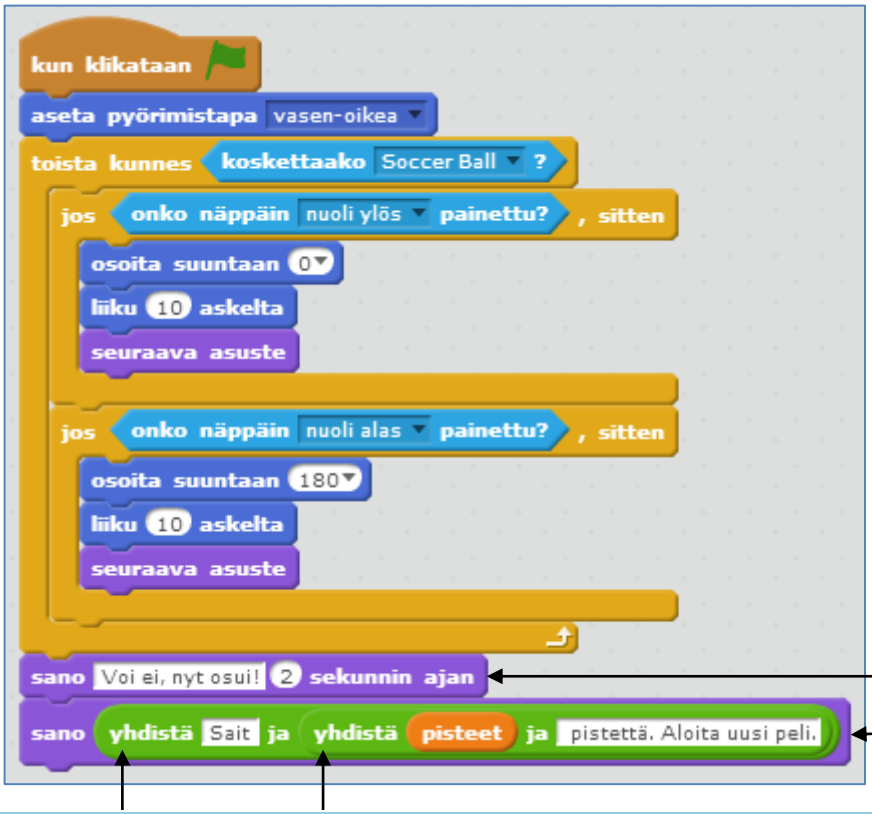
Jos ehto ”**onko näppäin <nuoli ylös> painettu?**” on **tosi**, niin silloin liikutetaan kissa-hahmoa ylöspäin. Vastaavasti jos ehto ”**onko näppäin <nuoli alas> painettu?**” on **tosi,** liikutetaan kissa-hahmoa alaspäin.




Asetetaan hahmon liikesuunta ylöspäin, liikutaan 10 askelta ja vaihdetaan asustetta. Kaksi sinistä koodipalikkaa löytyvät **Liike-osiosta** ja kolmas violetti **Ulkonäköosiosta**.

Asetetaan hahmon liikesuunta alaspäin, liikutaan 10 askelta ja vaihdetaan asustetta.

Toistorakenteen toiston päättyessä jalkapallo on osunut kissaan ja peli päättyy. Tällöin kissa kertoo osumasta puhekuplalla sanoen myös pelaajan pistemäärän. Alla olevassa kuvassa on esitetty kaikki kissa-hahmon koodipalikat.



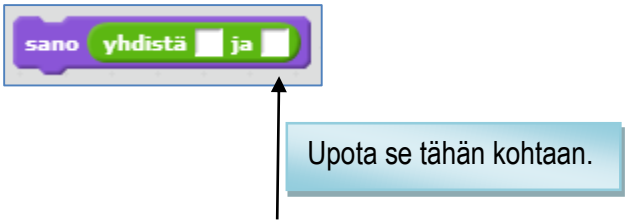

Löytyy **Ulkonäkö-osiosta**.

Komentopalikan ”**sano…**” parametrina on käytetty sisäkkäin kahta ”**yhdistä <hello> ja <world>**” funktiopalikkaa (löytyy **Toiminnot-osiosta**) sekä muuttujan **pisteet** arvoa (muuttujapalikka löytyy **Tieto-osiosta**).

Alimman koodipalikan muodostus tehdään neljässä vaiheessa:

1. Raahaa **sano**-komentopalikka koodialueelle ja pyyhi teksti ”Hello!” pois.


2. Raahaa **Toiminnot-osiosta** funktiopalikka ”**yhdistä <hello> ja <world>**” edellisen palikan parametriksi. Poista tekstit ”hello” ja ”world”.



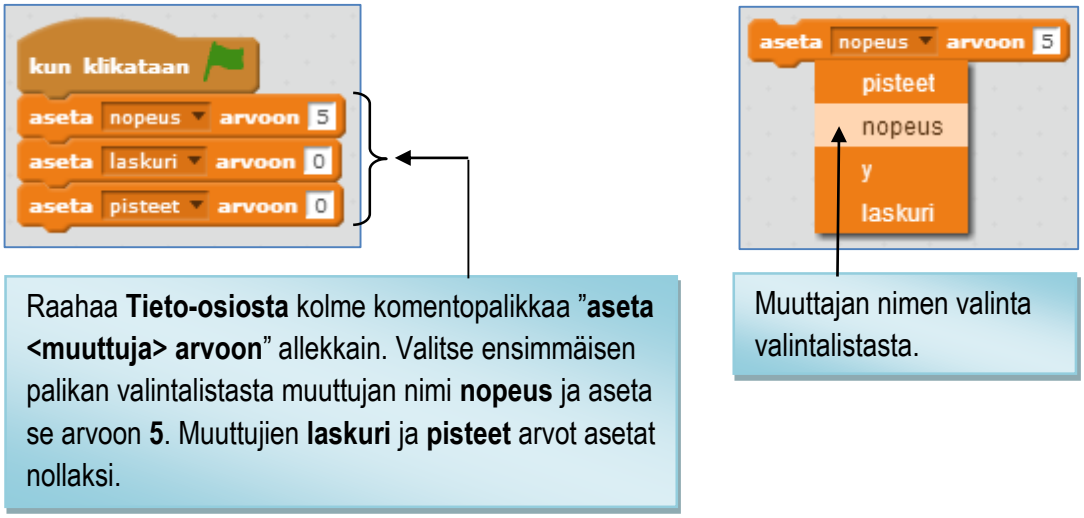

3. Raahaa toinen ”**yhdistä <hello> ja <world>**” funktiopalikka edellisen palikan toiseksi parametriksi ja poista siinä olevat tekstit.


4. Kirjoita ensimmäiseen valkoiseen neliöön teksti ”Sait” + välilyönti ja kolmenteen välilyönti + ”pistettä. Aloita uusi peli.”. Raahaa toiseen valkoiseen neliöön **Tieto-osiosta** muuttujapalikka **pisteet**.

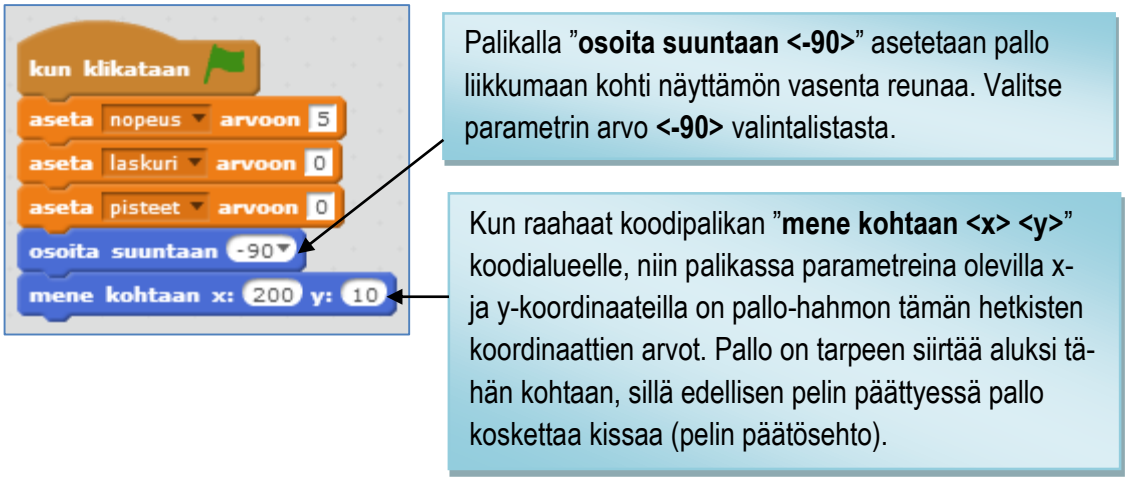

Jalkapallo-hahmon koodipalikat:

Klikkaa **Hahmot-ikkunasta jalkapallo-hahmon kuvaketta**, jolloin se tulee valituksi ja pääset sijoittamaan koodipalikoita tälle hahmolle.

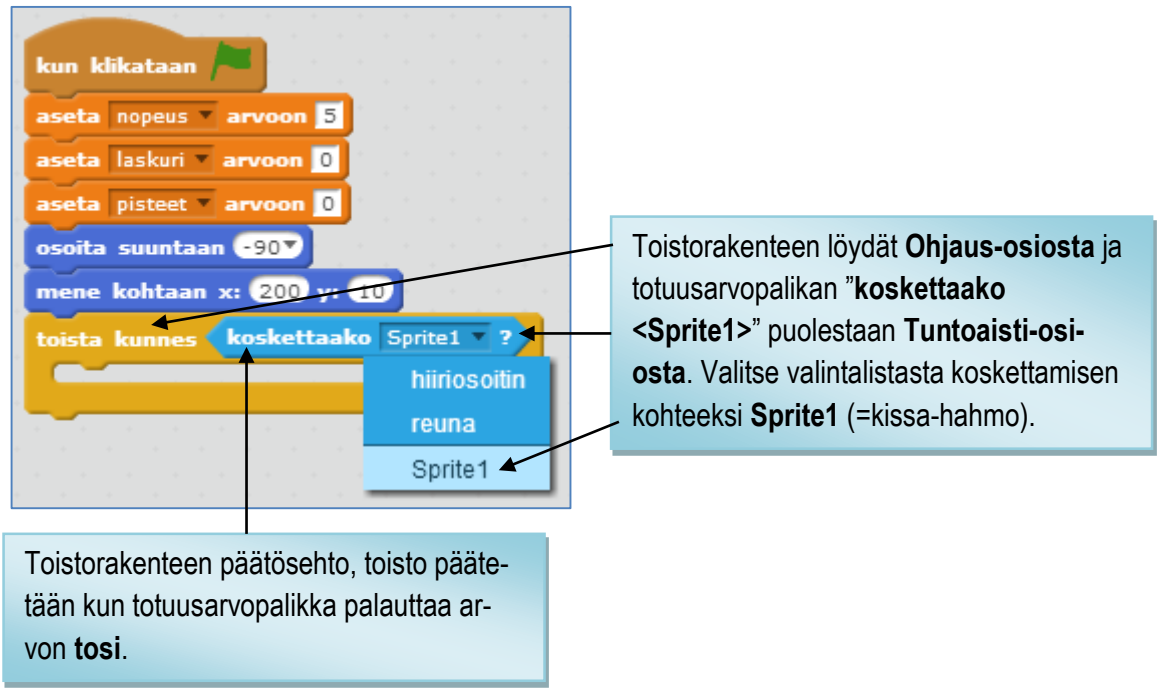
Jalkapallon ohjelmakoodin suoritus aloitetaan tapahtumalla ”kun klikataan vihreä lippu”. Tämän jälkeen asetetaan muuttujille halutut alkuarvot.




Pallo on nyt näyttämön oikeassa reunassa, eli kohdassa johon sen aiemmin hiirellä raahasit. Asetetaan seuraavaksi pallon liikesuunta ja lähtöpaikka. Tarvitset tähän kaksi **Liike-osiosta** löytävää komentopalikkaa.

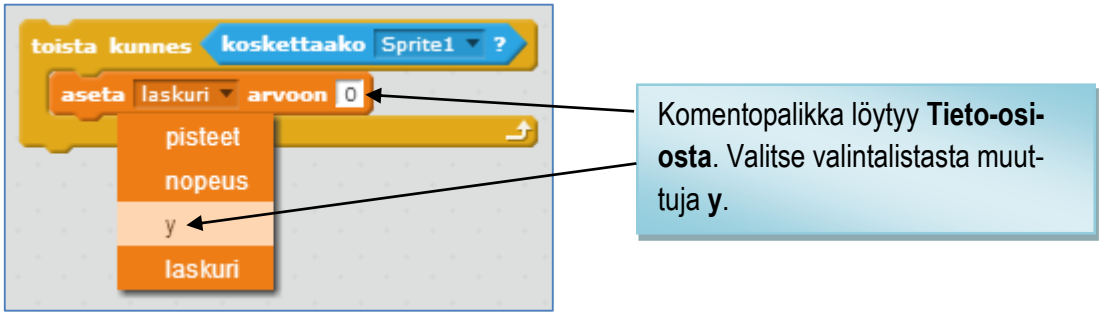



Mitä pallon tulee tehdä? Sen tulee liikkua näyttämön oikeasta reunasta vasempaan reunaan pikku hiljaa nopeutuvalla vauhdilla aina siihen asti, kunnes kissa ja pallo koskettavat toisiaan. Tarvitsemme tähän toistorakenteen, jonka lopetusehtona on hahmojen koskettaminen.

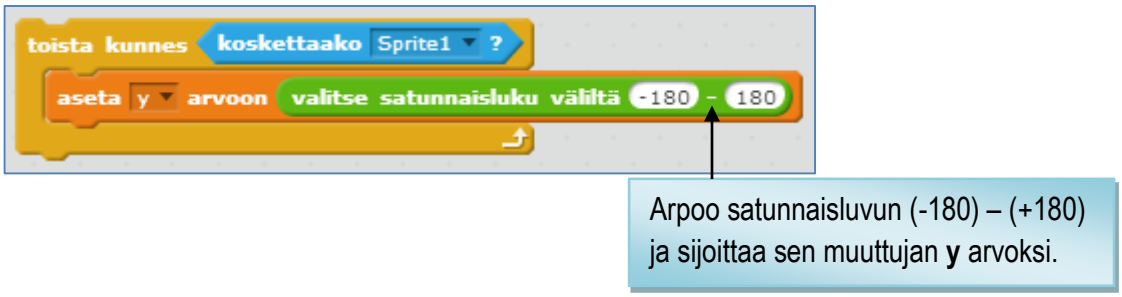



Kaikki seuraavat palikat upotetaan viimeksi lisätyn toistorakenteen "**toista kunnes <ehto>**" sisälle. Haluamme, että pallo lähtee liikkeelle aina eri korkeudelta (y-koordinaatista). Toteutamme tämän arpomalla y-koordinaatin ja siirtämällä pallon pystysuuntaisen sijainnin tähän kohtaan. Jatkossa esitämme vain toistorakenteen sisään tulevat palikat ja vasta lopuksi koko hahmon koodipalikat yhdessä kuvassa.

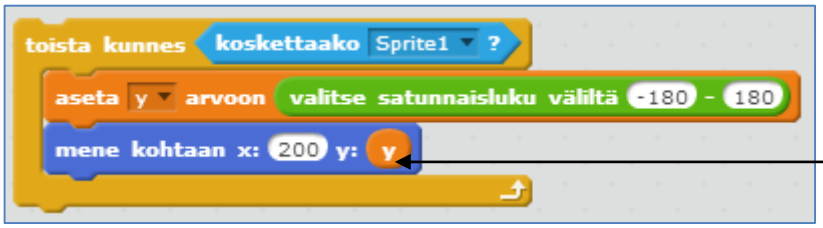
Raahaa toistorakenteen sisään ensimmäiseksi komentopalikka "**aseta <muuttuja> arvoon <x>**". Valitse valintalistasta muuttujaksi **y**.




Muuttujalle **y** annetaan arvoksi satunnaisluku väliltä (-180) – (+180). Raahaa **Toiminnot-osiosta** funktiopalikka "**valitse satunnaisluku väliltä <alaraja> - <yläraja>**" edellisen palikan parametriksi.



Pallo siirretään kohtaan x=200 ja y=arvottu luku käyttäen **Liike-osiosta** löytyvää komentopalikkaa ”**mene kohtaan < x > < y >**”.




Toisena parametrina on muuttujan **y** arvo, raahaa se tähän **Tieto-osiosta**.

Pallon tulee liikkua kohti näyttämön vasenta reunaa. Liike loppuu, jos pallo koskettaa kissaa tai reunaa. Tarvitsemme tähänkin toistorakennetta, jossa on kaksi ehtoa. Ehdot on liitetty toisiinsa loogisella **tai-operaattorilla**.

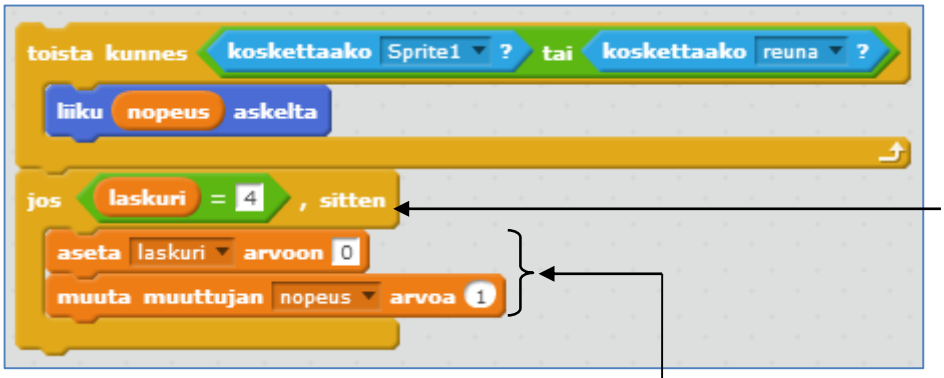

Lisää ensin toistorakenne ”**toista kunnes <ehto>**”. Löydät sen **Ohjaus-osiosta**. Raahaa sitten ehtoon **Toiminnot-osiosta** looginen **tai-operaattori.**

Lisää kaksi päätösehtoa: ”**koskettaako <sprite1>**” sekä ”**koskettaako <reuna>**” Nämä totuusarvopalikat löydät **Tuntoaisti-osiosta**. Valitse valintalistasta ensimmäiseksi kosketuksen kohteeksi **Sprite1** (kissa-hahmo) ja toiseksi **reuna**.


Toistorakennetta toistetaan, kunnes pallo koskettaa kissaa tai reunaa.

Muuttujan **nopeus** arvo määrää, kuinka monta askelta palloa liikutetaan yhdellä toistokerralla. Mitä suurempi muuttujan arvo on, sitä nopeammin pallo liikkuu lisäten pelin vaikeusastetta. Komentopalikka ”**liiku**” löytyy **Liike-osiosta** ja muuttujapalikan **nopeus** löydät **Tieto-osiosta**.

Muuttujan **laskuri** arvo kasvaa aina yhdellä pallon päästessä vasempaan reunaan. Jos laskurin arvo on neljä, asetetaan muuttujan **laskuri** arvoksi jälleen nolla ja kasvatetaan muuttujan **nopeus** arvoa yhdellä. Näin pallon liike nopeutuu pikku hiljaa pelin edetessä.




Ehtorakenteella tutkitaan, onko muuttujan **laskuri** arvo yhtä suuri kuin 4. Ehtorakenne löytyy **Ohjaus-osiosta**, vertailuoperaattori **Toiminnot-osiosta** ja muuttuja **Tieto-osiosta**.

Jos ehto "**laskuri = 4**" on **tosi**, suoritetaan nämä komentopalikat. Ensimmäisessä asetetaan muuttujan **laskuri** arvoksi **nolla** ja toisessa kasvatetaan muuttujan **nopeus** arvoa **yhdellä**. Koodipalikat löytyvät **Tieto-osiosta**.

Pelaaja on onnistunut väistämään pallon, jos pallo koskettaa reunaa ja se **ei samanaikaisesti kosketa kissaa**. Voimme selvittää tämän käyttämällä ehtorakennetta, loogisia **ja**- sekä **ei-opetaattoreita** sekä kahta kosketuksen havaitsevaa totuusarvopalikkaa. Ehtorakenne muodostetaan viidessä vaiheessa:

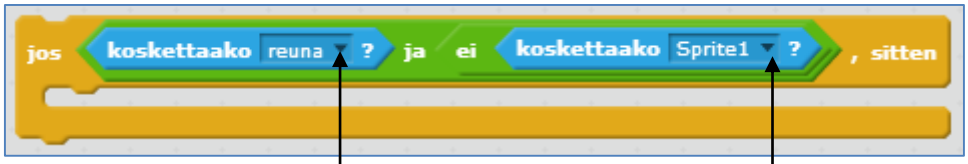
1. Raahaa koodialueelle ensin alla olevan kuvan mukainen ehtorakenne **Ohjaus-osiosta**.


2. Raahaa ehtoon looginen **ja-operaattori Toiminnot-osiosta**.


3. Raahaa ja-operaattorin toiseen ehtoon vielä looginen **ei-operaattori** myös **Toiminnot-osiosta**.


Looginen **ei-operaattori** upotettuna **ja-operaattorin** toiseen ehtoon.

4. Raahaa ensimmäiseksi ehdoksi totuusarvopalikka ”**koskettaako <reuna>**” ja toiseksi ehdoksi ”**koskettaako <Sprite1>**”. Molemmat löydät **Tuntoaisti-osiosta**.



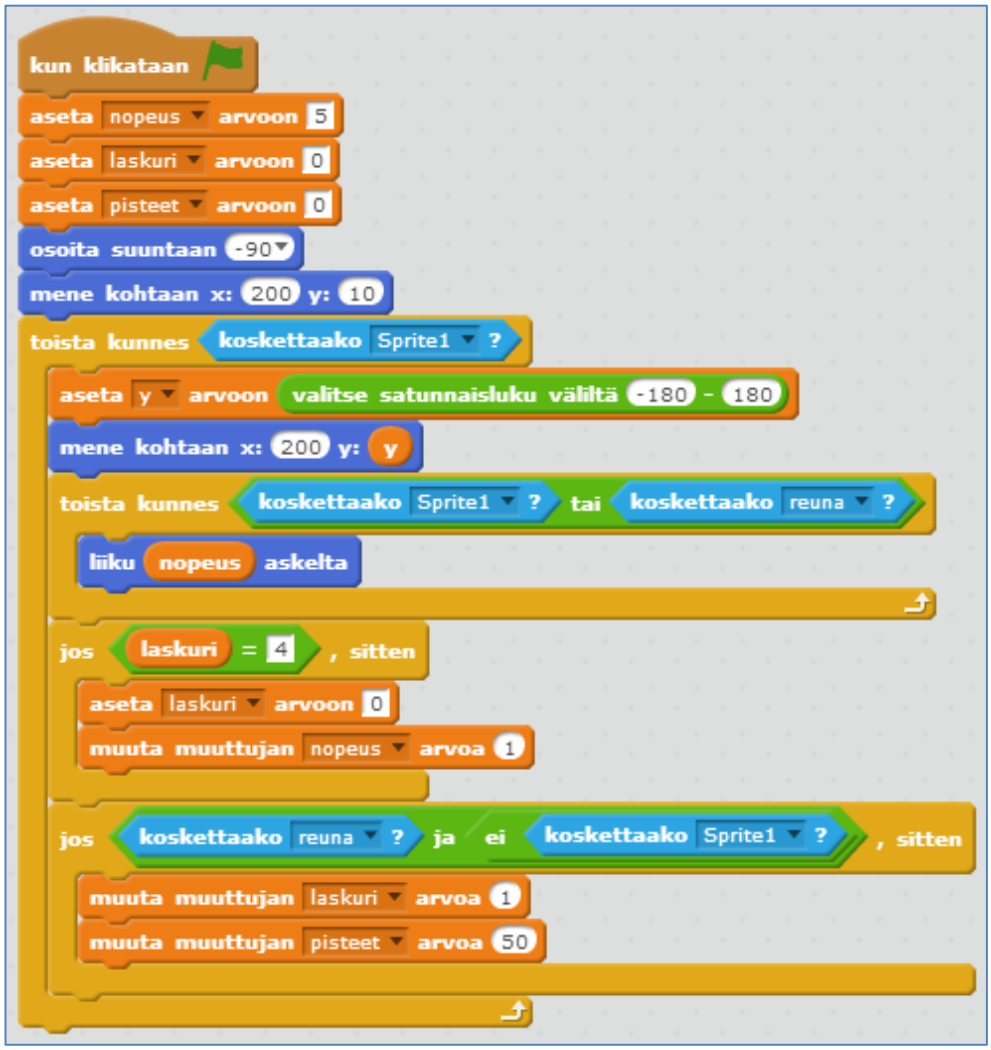

Valitse valintalistasta ensimmäiseksi kosketuksen kohteeksi **reuna** ja toiseksi **Sprite1**.

5. Ehtorakenteen ehto on **tosi**, kun pallo koskettaa reunaa ja **ei kosketa** kissaa. Tällöin kasvatetaan muuttujan **laskuri** arvoa yhdellä ja muuttujan **pisteet** arvoa 50:llä. Raahaa ja upota ehtorakenteen sisään kaksi ”**muuta muuttujan <nimi> arvoa <määrä>**” komentopalikkaa.


Löydät molemmat komentopalikat **Tieto-osiosta**. Ensimmäisen palikan valinta-listasta valitse muuttujan nimeksi **laskuri** ja syötä muutettavaksi arvoksi luku **1**. Valitse toiseen palikkaan muuttujan nimeksi **pisteet** ja syötä arvoksi luku **50**.

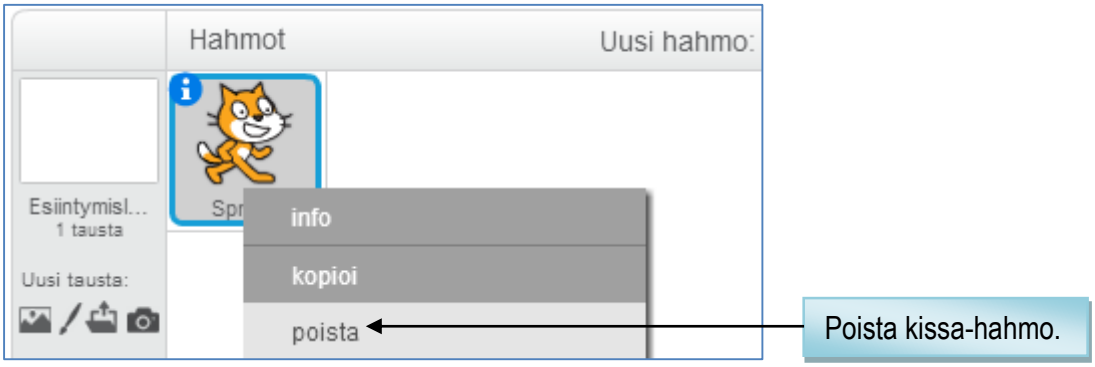
Väistelypeli on nyt valmis ja voit aloittaa pelaamisen käynnistämällä ohjelman vihreä lippu-kuvaketta klikkaamalla. Seuraavalla sivulla on esitetty vielä pallo-hahmon kaikki koodipalikat. Alla oleva kuva esittää pelin näyttämöä.

Pallo-hahmon kaikki koodipalikat:



## Ristiin rastiin näyttämöä lentävä papukaija

Tähän ohjelmaan tarvitset uuden hahmon. Poista ensin kissa-hahmo klikkaamalla hiiren oikeaa painiketta **Hahmot-ikkunassa** kissan kuvakkeen päällä ja valitse valikosta valinta **Poista**.




Lisää sitten uusi hahmo kuvakkeesta ”**valitse hahmo kirjastosta**”.

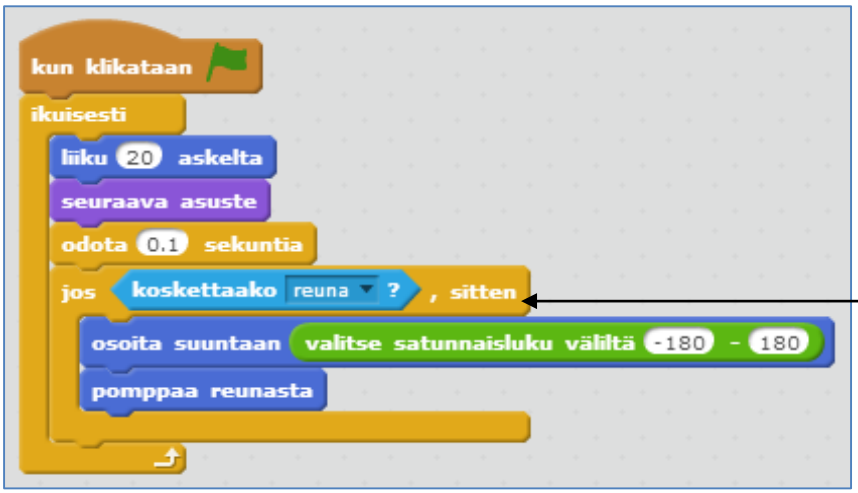

Selaa kirjastoa ja valitse sieltä **papukaija** (**Parrot**).


Pienennä papukaijan hahmoa **pienennystyökalulla**.


Käytetään tässä ohjelmassa myös taustakuvaa. Klikkaa **Hahmot-ikkunasta** kuvaketta ”**Valitse tausta kirjastosta**”.

Valitse taustaksi **desert**.


Klikkaa **papukaijan kuvaketta hahmot-ikkunassa**. Näin pääset laatimaan koodia nimenomaan papukaijan hahmolle. Alla olevassa kuvassa on esitetty papukaija-hahmon kaikki koodipalikat. Ohjelman suoritus käynnistyy vihreää lippua klikkaamalla. Sen jälkeen ohjelma siirtyy ikuiseen toistorakenteeseen. Toistorakenteessa liikutetaan papukaijaa 20 askelta, vaihdetaan seuraava asuste ja odotetaan 0.1 sekuntia. Ehtorakenteessa tutkitaan, koskettaako papukaija näyttämön reunaa, jos koskettaa, ehto on **tosi** ja silloin suoritetaan ehtorakenteen **sitten-osa** (nykyisessä Scratchin versiossa osan nimi on ”**niin”**). Tässä osassa olevat koodipalikat arpovat papukaijalle satunnaisen suunnan ja pompauttavat sen irti reunasta.




Jos papukaija-hahmo koskettaa reunaa tehdään kolme asiaa. 1. Arvotaan satunnaisluku väliltä (-180) – (+180). 2. Osoitetaan papukaija arvottuun suuntaan. 3. Pompautetaan papukaija irti reunasta.

Tämä koodi lennättää papukaijaa satunnaisesti ristiin rastiin näyttämöä kunnes ohjelman suoritus päätetään punaisesta stop-merkistä.

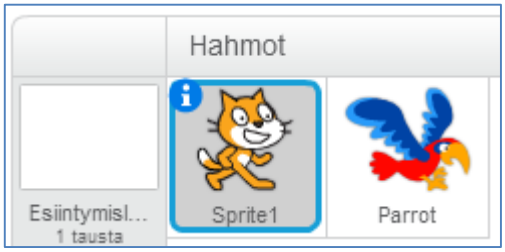
## Papukaijaa jahtaava kissa

Ohjelman suunnittelu.

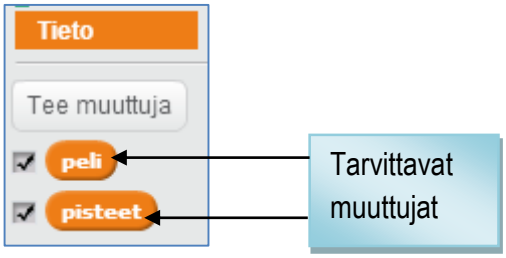
Ohjelmassa on kaksi hahmoa, papukaija ja kissa. Papukaija lentää ristiin rastiin pitkin näyttämöä ja osuessaan reunaan se päästää äänen ”bird”. Pelaaja ohjaa kissa-hahmoa nuolinäppäimillä neljään eri suuntaan. Näyttämöllä on taustakuva, jonka päällä hahmot liikkuvat. Ohjelman käynnistämisen jälkeen papukaija aloittaa lentämisen ja ruudussa näkyvät pisteet lähtevät vähenemään 10 000 alaspäin. Papukaija lähtee liikkeelle näyttämön oikeasta yläkulmasta ja kissa vasemmasta alakulmasta. Peli päättyy, kun kissa saa papukaijan kiinni. Tällöin kissa sanoo puhekuplalla tekstit ”<pistemäärä> pistettä” ja ”Peli päättyi”. Tämän jälkeen hahmot eivät enää liiku, vaan ohjelma tulee käynnistää uudestaan.

Ohjelman toteutus:

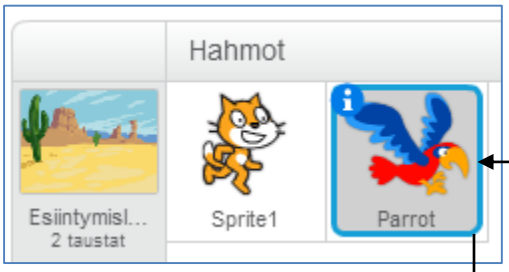
Ohjelmaan tarvitaan kaksi hahmoa, kissa ja papukaija. Näyttämölle tulevat hahmot ovat aluksi liian suuria, joten ne tulee pienentää sopivan kokoisiksi pienennystyökalulla.




Ohjelmassa on kaksi muuttujaa **peli** ja **pisteet**. Molemmat muuttujat näkyvät kaikille hahmoille. Muuttuja **peli** kertoo hahmoille, milloin peli on käynnissä (arvo = 1) ja milloin se päättyy (arvo = 0). Muuttujan **pisteet** arvona on pelaajan pistemäärä.

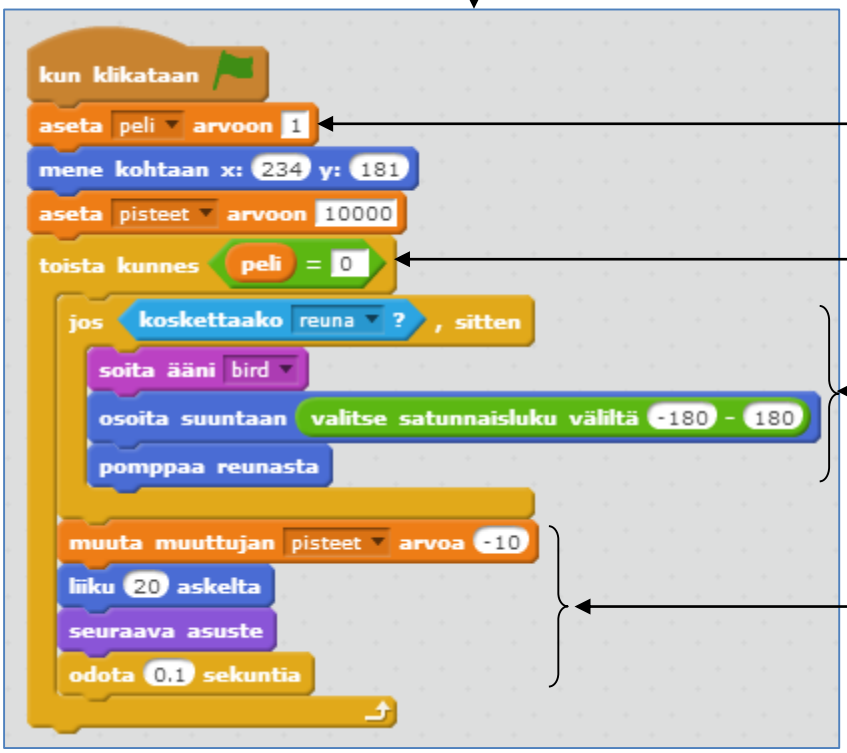


Kummallekin hahmolle laaditaan oma koodi. Hahmo, jolle koodi laaditaan, valitaan **Hahmot-ikkunasta**.




Valitse **Hahmot-ikkunasta** ensin **papukaija-hahmo** klikkaamalla sen kuvaketta hiirellä.

Papukaija-hahmon koodipalikat:




Kun muuttujan **peli** arvo on **1**, peli on käynnissä.

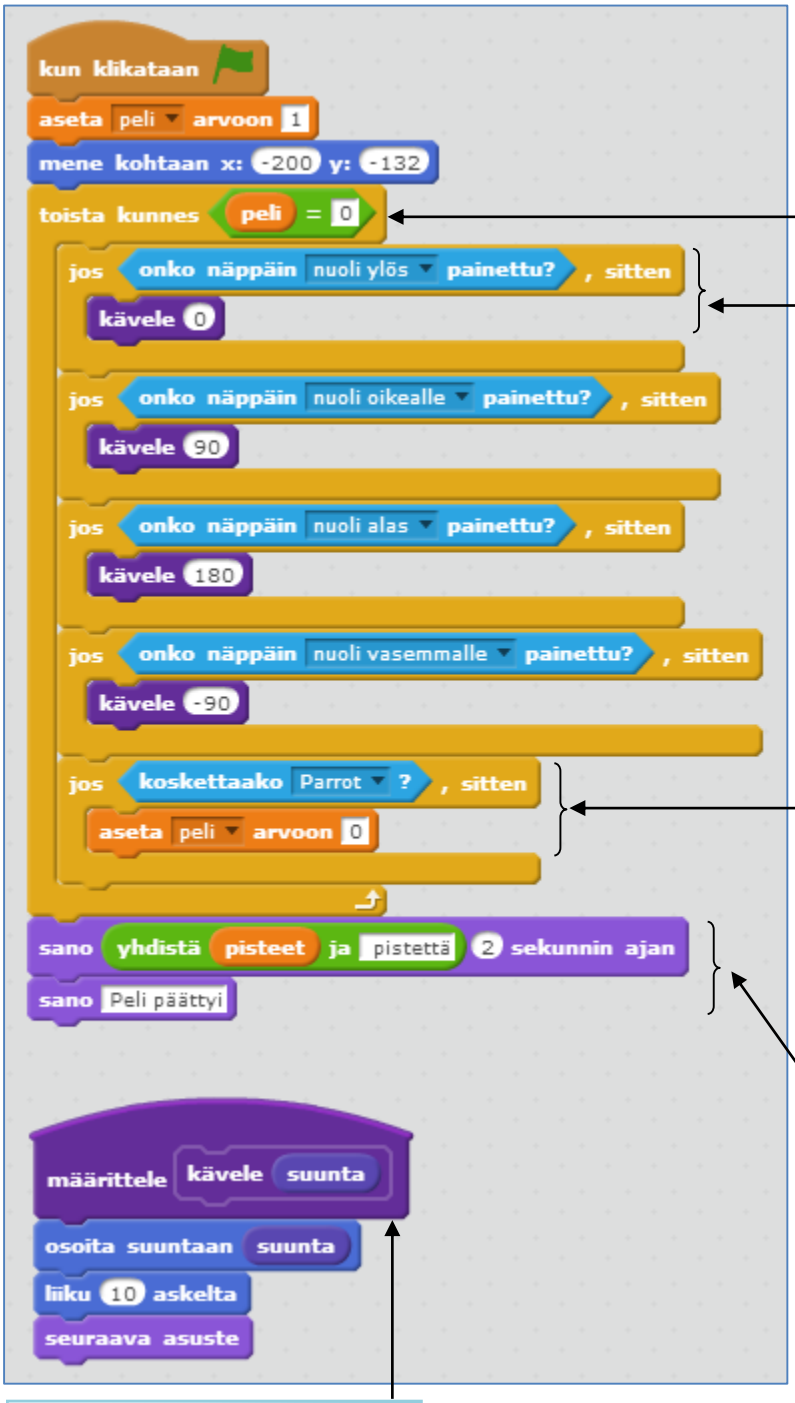
Toistetaan niin kauan, kunnes muuttujan **peli** arvo on **0**.

Jos papukaija koskettaa reunaa, soitetaan ääni ”bird”, arvotaan sille uusi suunta ja pompautetaan se pois reunasta.

Vähennetään pisteitä ja liikutetaan papukaijaa vaihtaen samalla asustetta.

Valitse seuraavaksi **Hahmot-ikkunasta kissa-hahmo** klikkaamalla hahmon kuvaketta hiirellä ja muodosta hahmolle alla olevan kuvan mukaiset koodipalikat.

Kissa-hahmon koodipalikat:




Kun muuttujan **peli** arvo on **1**, peli on käynnissä.

Jos totuusarvopalikka ”**onko näppäin <nuoli ylös> painettu?**” palauttaa arvon **tosi**, suoritetaan ehtorakenteeseen upotettu koodipalikka. Tällöin siirrytään palikkaan **kävele** ja annetaan parametrille **suunta** arvoksi **0** (ylöspäin).

Jos totuusarvopalikka ”**koskettaako <Parrot>?**” palauttaa arvon **tosi**, niin silloin kissa ja papukaija koskettavat toisiaan. Tällöin suoritetaan ehtorakenteeseen upotettu komentopalikka, jossa asetetaan muuttujan **peli** arvoksi **0**. Tämän seurauksena molemmissa hahmoissa olevat toistorakenteet

Peli päättyy samalla, kun toistorakenne päättyy. Tällöin hahmo sanoo puhekuplalla pisteet ja tekstin ”Peli päättyi”

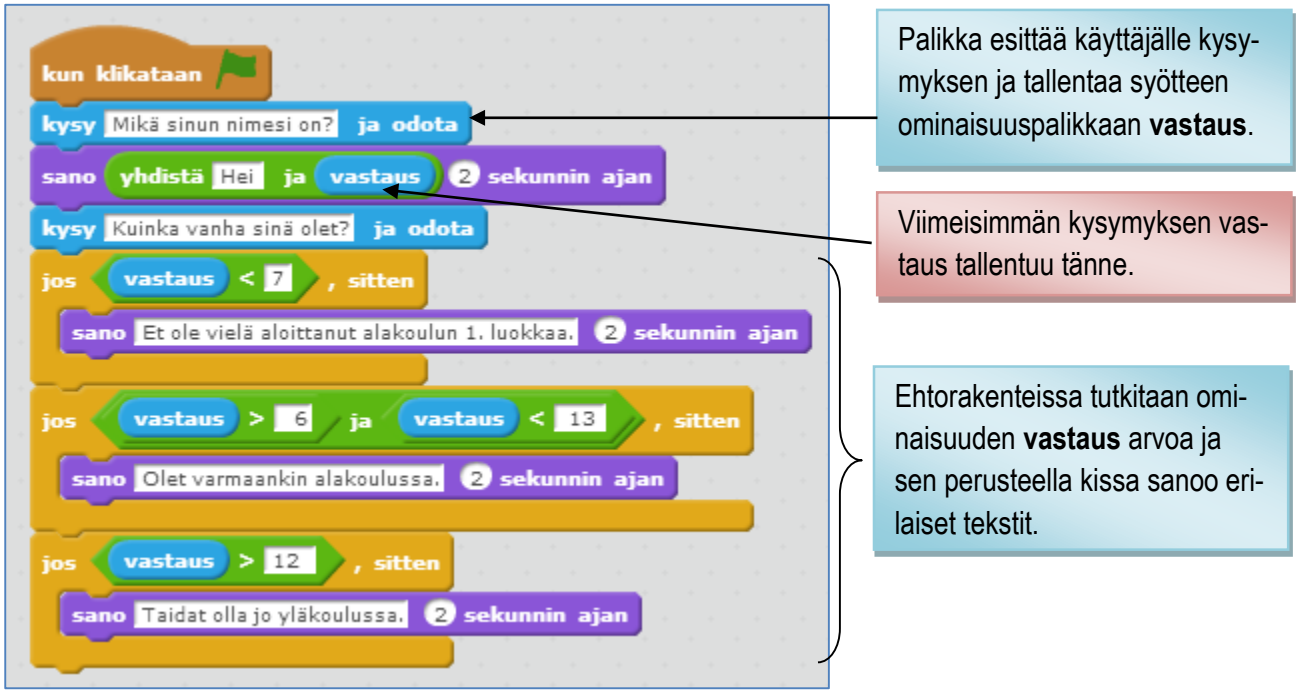
**Kävele**-palikkaa käytetään kissa-hahmon liikuttamiseen ja asusteen vaihtamiseen. Kissan suunta osoitetaan parametrin **suunta** arvolla.

## Syötteen kysyminen käyttäjältä ja sen käsittely

Ohjelman suunnittelu:

Kissa-hahmo kysyy käyttäjältä kaksi kysymystä, joihin käyttäjä vastaa kirjoittamalla vastauksensa tekstikenttään. Ensin kissa kysyy käyttäjän nimeä ja vastauksen saatuaan sanoo ”Hei <nimi>”. Sitten kissa kysyy käyttäjän ikää ja iän perusteella kissa sanoo ”Et ole vielä aloittanut alakoulun 1. luokkaa” (vastaus < 7), ”Olet varmaankin alakoulussa” (6 < vastaus < 13), tai ”Taidat olla jo yläkoulussa” (vastaus > 12).

Ohjelman toteutus:



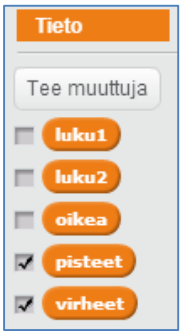
## Kertolaskupeli

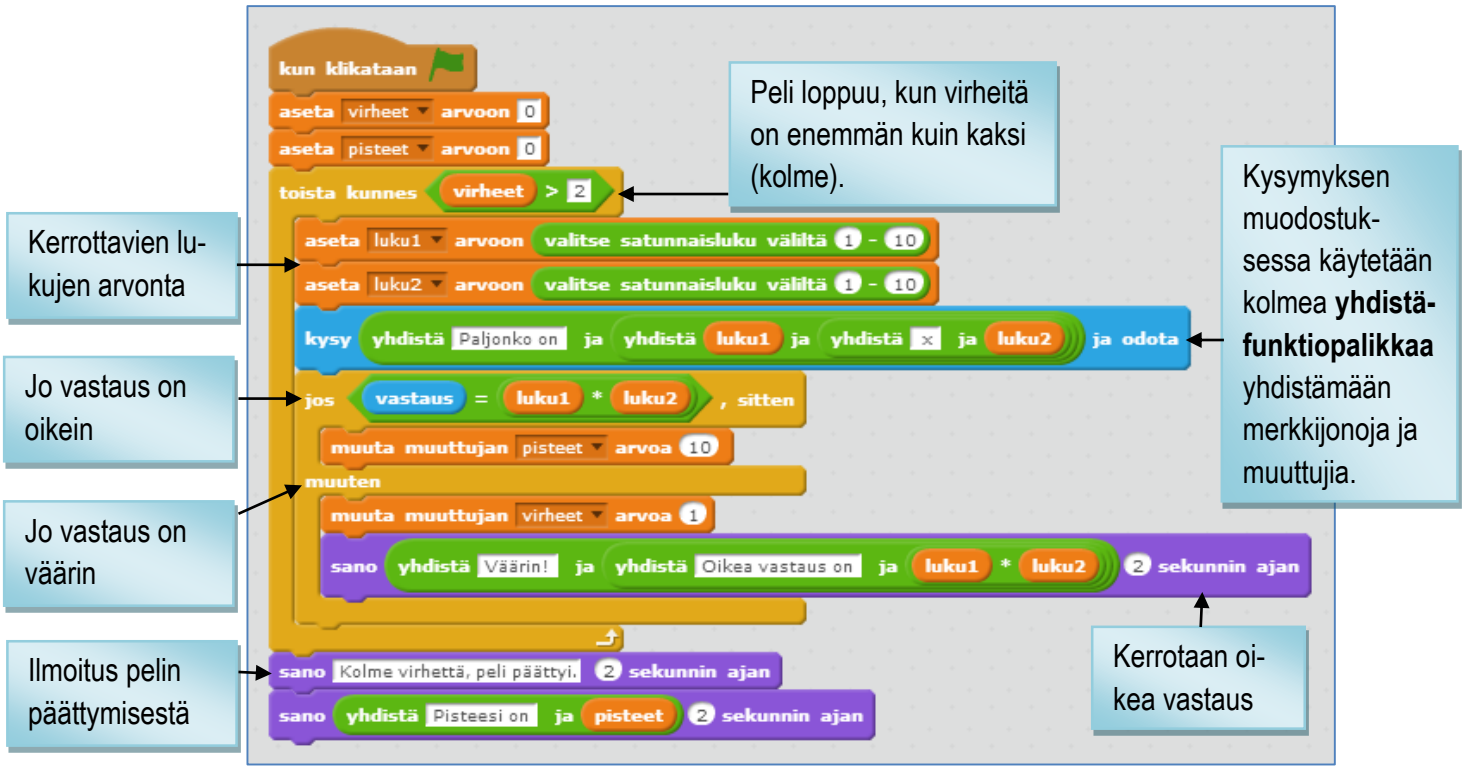
Ohjelman suunnittelu:

Kissa esittää pelaajalle kertolaskukysymyksiä (1-10 kertotaulu), joihin pelaaja vastaa syöttämällä mielestään oikean vastauksen. Jos vastaus on oikein, pisteet lisääntyvät kymmenellä ja kissa esittää uuden kysymyksen. Jos taas vastaus on väärin, kissa ilmoittaa oikean tuloksen ja virheet kasvavat yhdellä. Peli jatkuu niin kauan, kunnes pelaajan virheiden määrä kasvaa kolmeen. Kolmannen virheen jälkeen kissa ilmoittaa pelin päätyneen ja kertoo pelaajan pisteet.

Ohjelman toteutus:

Pelissä tarvitaan kaikkiaan viisi muuttujaa, joista muuttujat **pisteet** ja **virheet** ovat näkyvissä näyttämöllä, muut kolme muuttujaa ovat **luku1**, **luku2** ja **oikea**.



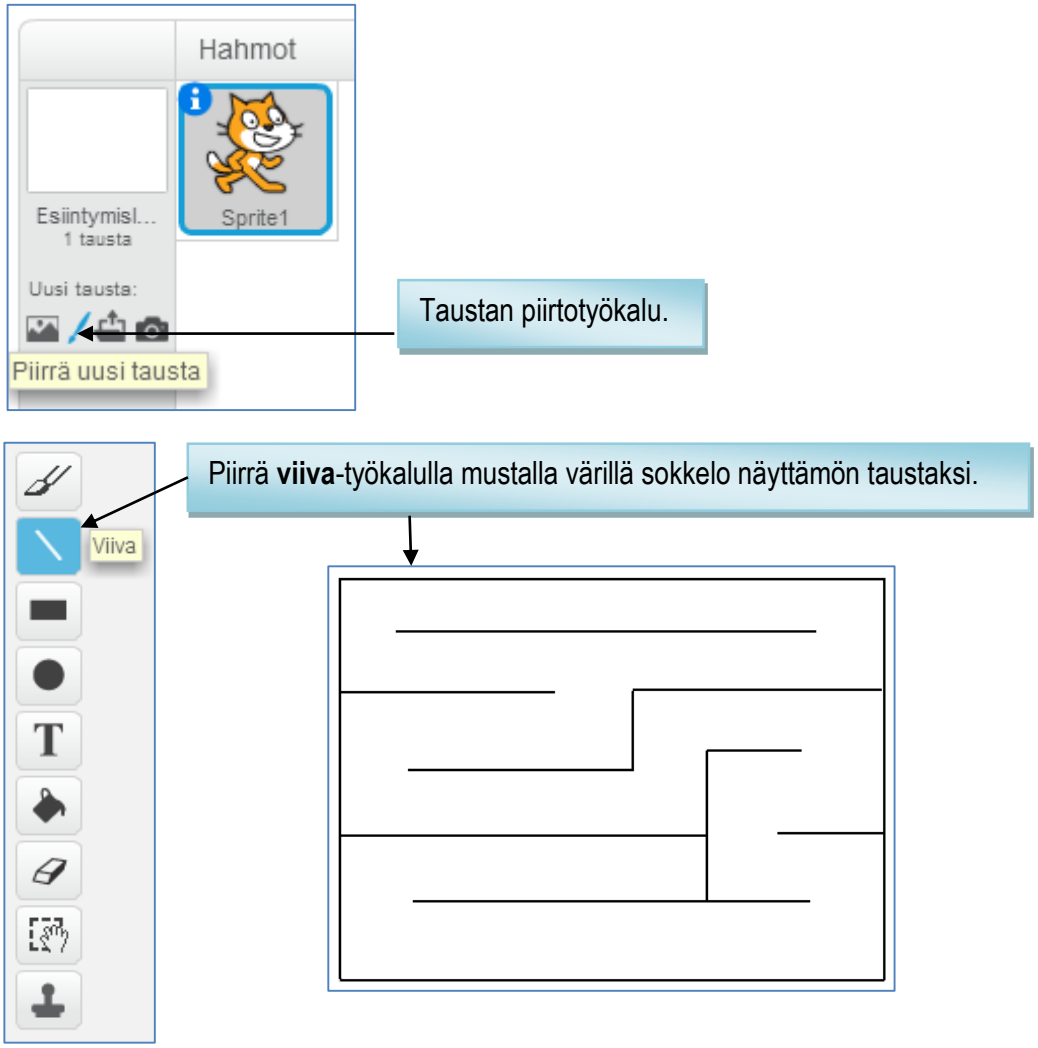
# Sokkelopeli

Ohjelman määrittely:

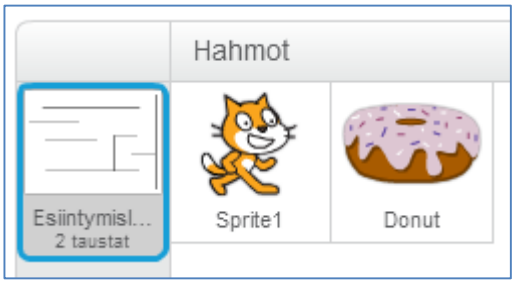
Näyttämön taustalla on mustilla viivoilla piirretty sokkelo. Pelaaja ohjaa kissaa sokkelossa nuolinäppäimillä ja hänen tarkoituksena on viedä kissa sokkelon alaosassa olevan donitsin luo. Peli loppuu, jos kissa osuu matkalla mustaan viivaan. Pelaajan pisteet lähtevät vähenemään 10 000:sta alkaen. Mitä kauemmin matka donitsin luo kestää, sitä vähemmän pelaajalle tulee pisteitä. Peli päättyy onnistuneesti silloin, kun kissa pääsee donitsin luo koskettamatta matkalla kertaakaan mustia viivoja. Kissan koskettaessa donitsia se häviää näkyvistä = kissa söi sen.

Ohjelman totetus:

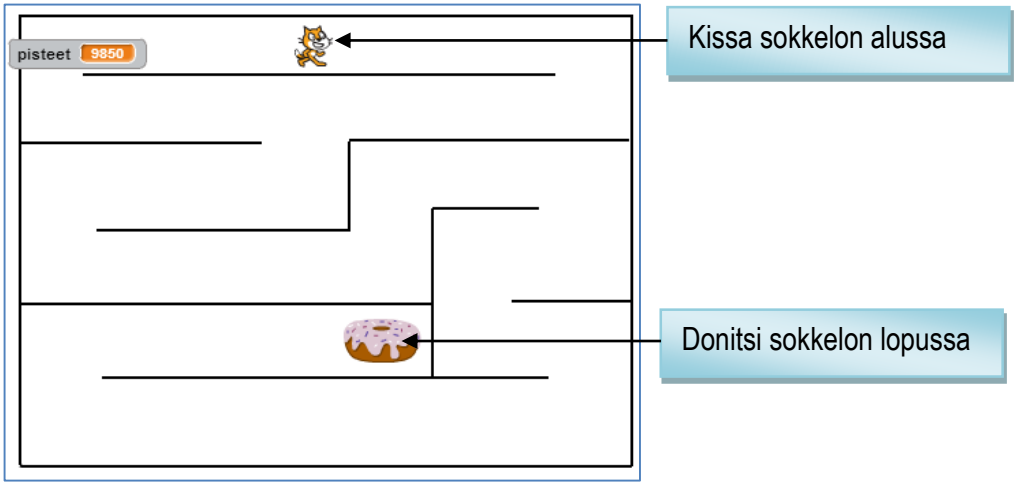
Tähän peliin ohjelmoijan tulee piirtää itse sokkelona toimiva tausta.



Lisää projektiin toiseksi hahmoksi **donitsi** (Donut).

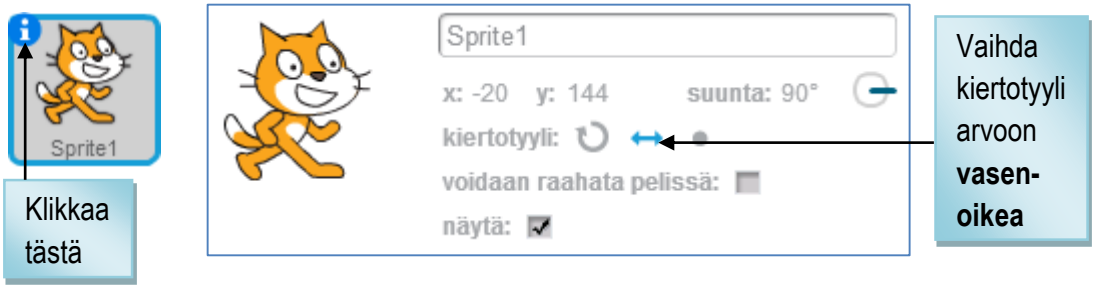



Pienennä molemmat hahmot niin, että ne sopivat hyvin piirtämääsi sokkeloon. Etenkin kissa-hahmolla tulee olla juuri sopivasti tilaa, jotta pelin vaikeusaste on sopiva. Sijoita kissa sokkelon alkuun ja donitsi loppuun.

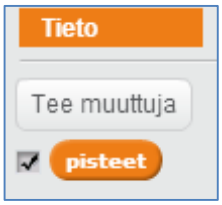



Sijoita kissa haluamaasi aloituskohtaan, etsi **Liike-osiosta** komentopalikka ”**mene kohtaan <x>, <y>**”. Siirtäessäsi kissaa näyttämöllä tämän palikan parametrit muuttuvat automaattisesti kissan koordinaattien mukaan. Raahaa tämä komentopalikka valmiiksi **koodialueelle** myöhempää käyttöä varten.


Klikkaa **hahmot-ikkunassa** kissa-hahmon vasemmassa ylänurkassa olevaa **i-kirjaimen kuvaketta**. Näin pääset asettamaan hahmon ominaisuuksia.



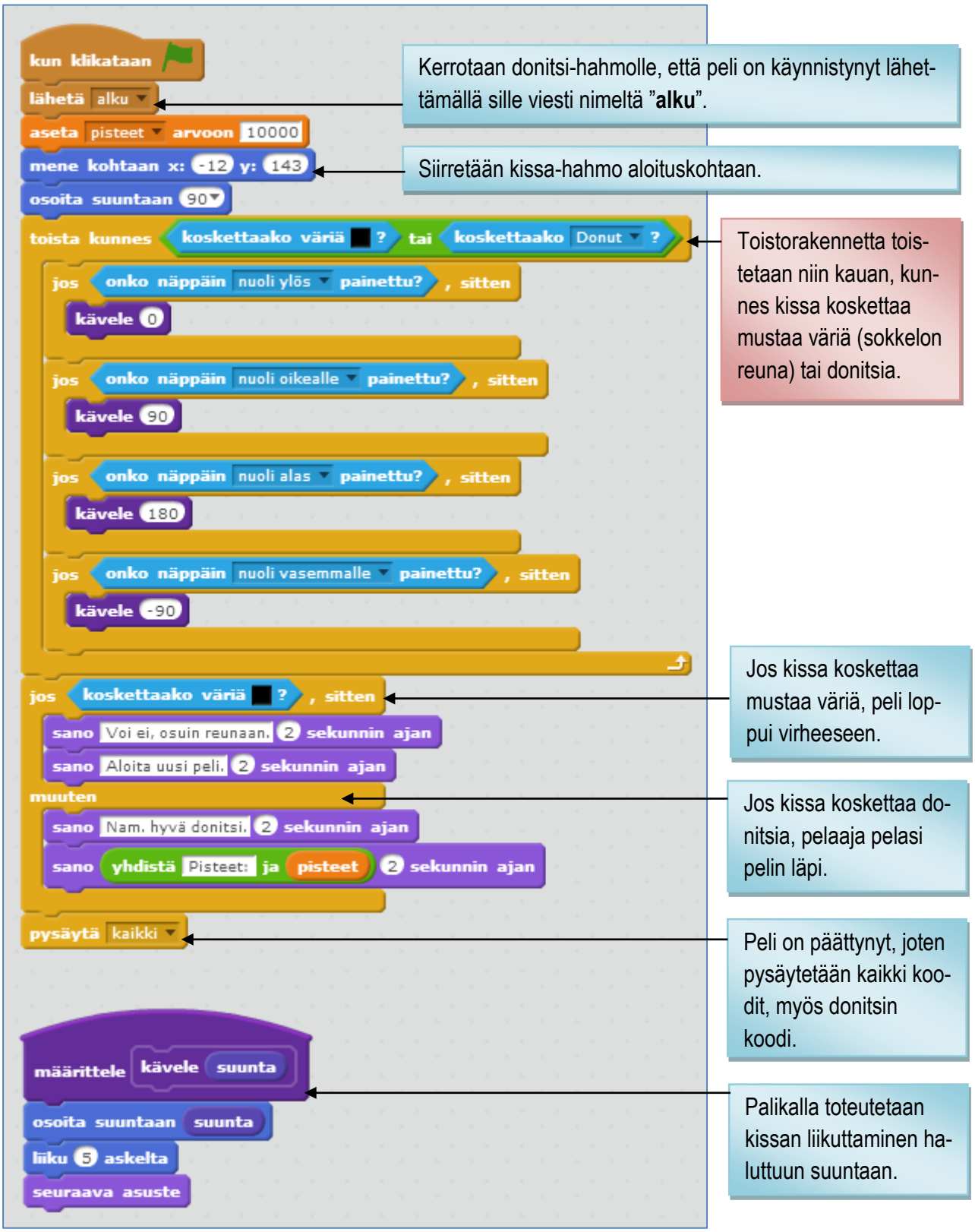
Takaisin hahmoihin pääset vasemman yläkulman sinisellä nuolella. Tarvitset peliin vain yhden muuttujan pisteitä varten ja muuttujan arvo näytetään näyttämöllä. Anna muuttujalle nimeksi **pisteet** ja tuo se näkyviin ruksaamalla valinta.




Molemmat hahmot tarvitsevat omat koodipalikkansa. Hahmo, jolle koodipalikat lisätään, valitaan **hahmot-ikkunasta**.

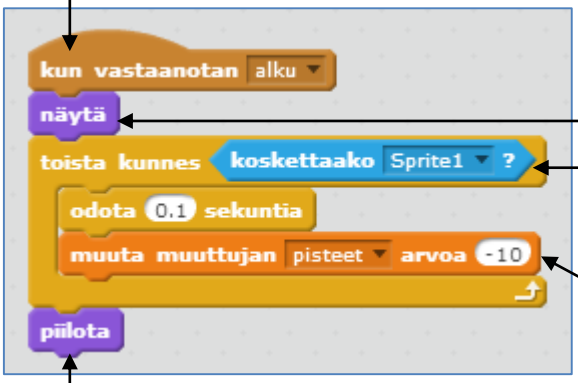
Kissan koodipalikat:

Ensimmäiseksi heti tapahtumapalikan jälkeen kissa lähettää donitsille viestin nimeltä ”**alku**”.



Donitsin koodipalikat:

Jokaisella hahmolla on itsenäinen koodi, mutta hahmot voivat viestiä toisilleen yhteisillä muuttujilla tai lähettämällä tapahtumia. Kun peli käynnistyy, kissa-hahmo lähettää donitsi-hahmolle viestin **alku** (viestin nimi on ”**alku**”). Donitsin koodi vastaanottaa tämän tapahtuman ja siitä käynnistyy donitsin toiminta.




Tuodaan donitsi näkyviin, koska jos kissa aiemmassa pelissä on päässyt donitsin luo, se on piilotettuna.

Toistorakennetta toistetaan, kunnes donitsi koskettaa kissaa.

Vähennetään pisteitä.

Donitsi on koskettanut kissaa ja toistorakenne on päättynyt. Piilotetaan donitsi pois näkyvistä = kissa söi sen.

## Nuottien soittaminen eri soittimilla

Ohjelman määritys:

> Käyttäjän painaessa numeronäppäintä 1-5, ohjelma soittaa alimmat nuotit 1-5 (48, 50, 52, 53, 55). Nuotin kestoa voi kasvattaa ”nuoli ylös”-näppäimellä ja vähentää ”nuoli alas”-näppäimellä. ”Nuoli oikealle”-näppäin siirtyy seuraavaan soittimeen ja ”nuoli vasemmalle”-näppäin edelliseen soittimeen. Nuotin kesto ja soittimen numero näkyvät näyttämöllä.

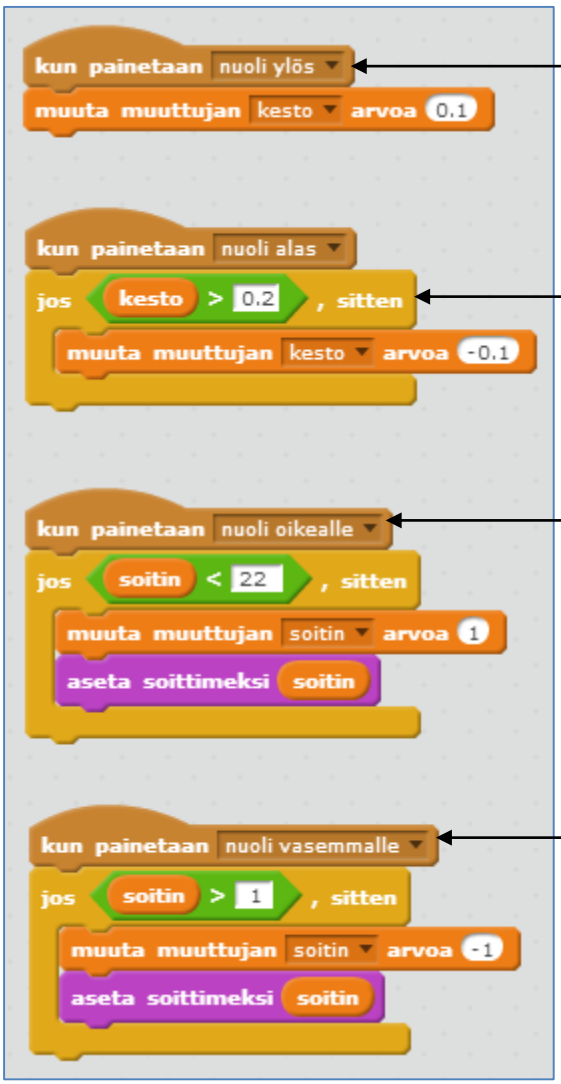
Ohjelman toteutus:

Ohjelmassa on kaksi muuttujaa, **soitin** ja **kesto**, muodosta ne **Tieto-osiossa**.


Asetetaan muuttujalle **soitin** arvo 1 ja muuttujalle **kesto** arvo 0.5.

Asetetaan **tempo** arvoon 60 bmp ja soittimeksi muuttujan **soitin** arvo, eli 1.

Jos numeronäppäintä ”1” on painettu, soitetaan nuottia ”48” muuttujan **kesto** määrittämän ajan.




Tapahtuman ”**kun painetaan <nuoli ylös>**” aktivoituessa kasvatetaan muuttujan **kesto** arvoa määrällä **0.1**. Tämä kasvattaa nuotin soittamisen kestoa.

Jos muuttujan **kesto** arvo on suurempi kuin **0.2**, vähennetään muuttujan arvoa määrällä **0.1**. Tämä pienentää nuotin soittamisen kestoa.

Vaihdetaan seuraavaan soittimeen.

Vaihdetaan edelliseen soittimeen.

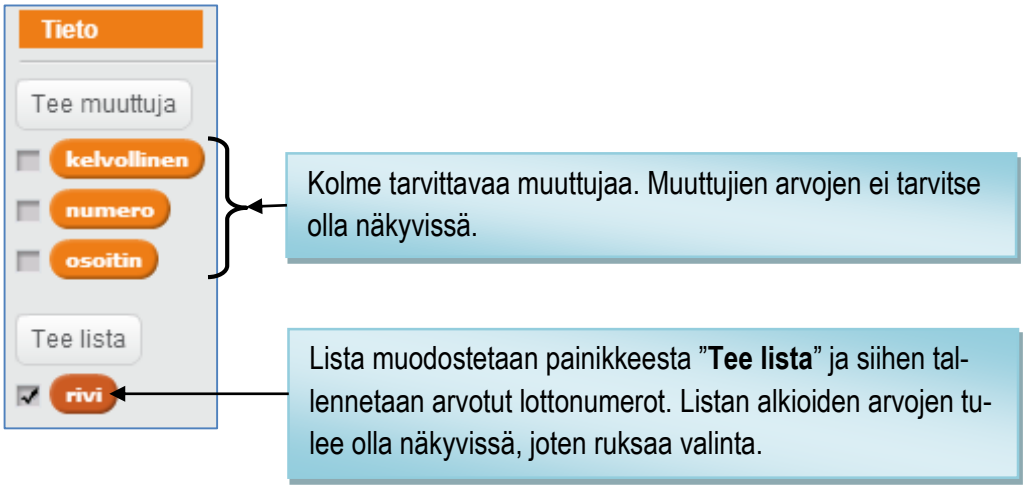
## LOTTORIVIN ARVONTA

Ohjelman määrittely:

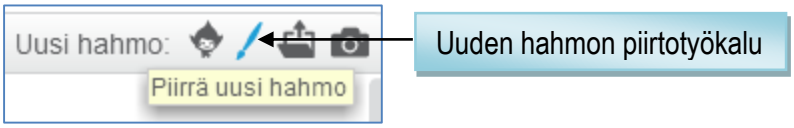
Ohjelmassa on painike ”Arvo lottorivi”, jota klikkaamalla rivin arvonta käynnistyy. Siirrettäessä hiiriosoitin painikkeen päälle sen tausta muuttuu keltaiseksi osoittaen mahdollisuutta klikata sitä. Ohjelma arpoo seitsemän numeroa väliltä 1-39 ja jokainen arvottu numero voi esiintyä ainoastaan yhden kerran rivissä. Ohjelma näyttää arvotun rivin näyttämöllä listan arvoina. Yhden arvonnan jälkeen ohjelmalla voidaan arpoa lisää rivejä, mutta vain yksi rivi näkyy kerrallaan arvonnan tuloksena.

Ohjelman toteutus:

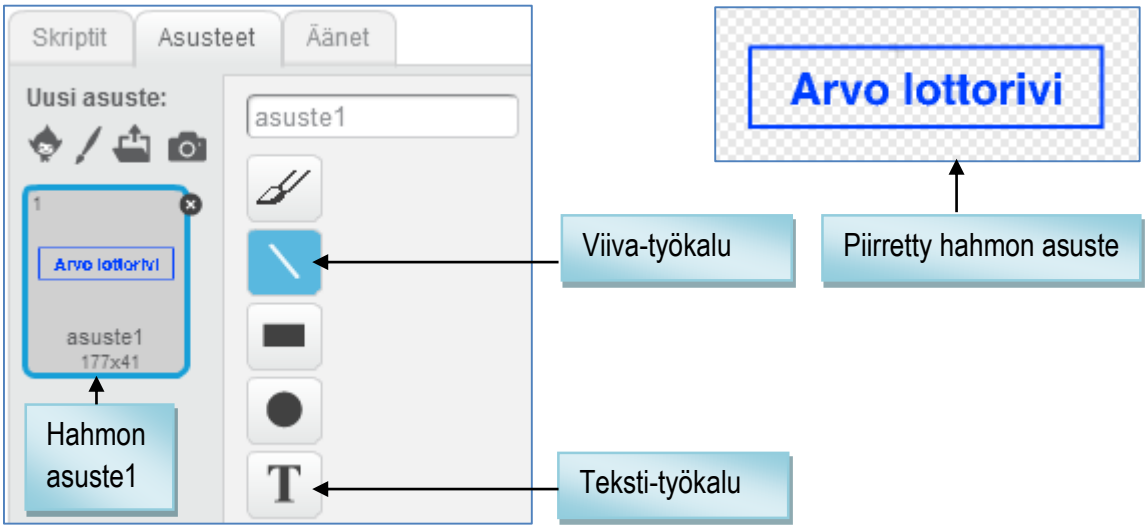
Tässä esimerkissä keskeistä on listan käyttäminen tietovarastona. Listaan tallennetaan arvotut numerot. Lisäksi tarvitaan kolme muuttujaa. Anna listalle nimeksi **rivi** ja muuttujille nimet **kelvollinen**, **numero** ja **osoitin**.



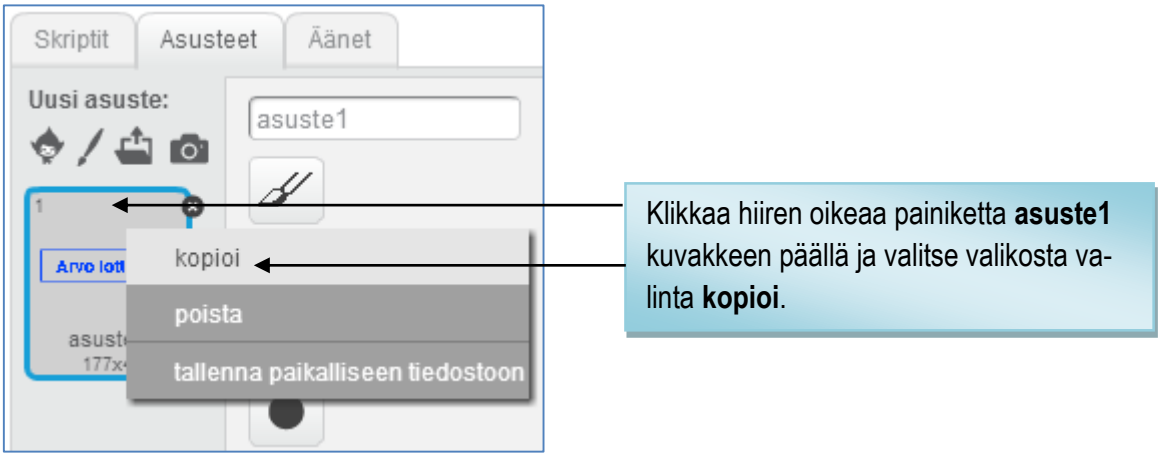

Kissa-hahmoa ei tarvita, joten poista se projektista klikkaamalla hiiren oikeaa painiketta **Hahmot-ikkunassa kissa-hahmon** päällä ja valitse valikosta valinta **poista**. Piirrämme itse painikkeen hahmon, aloita klikkaamalla **Hahmot-ikkunasta** kuvaketta piirrä uusi hahmo.



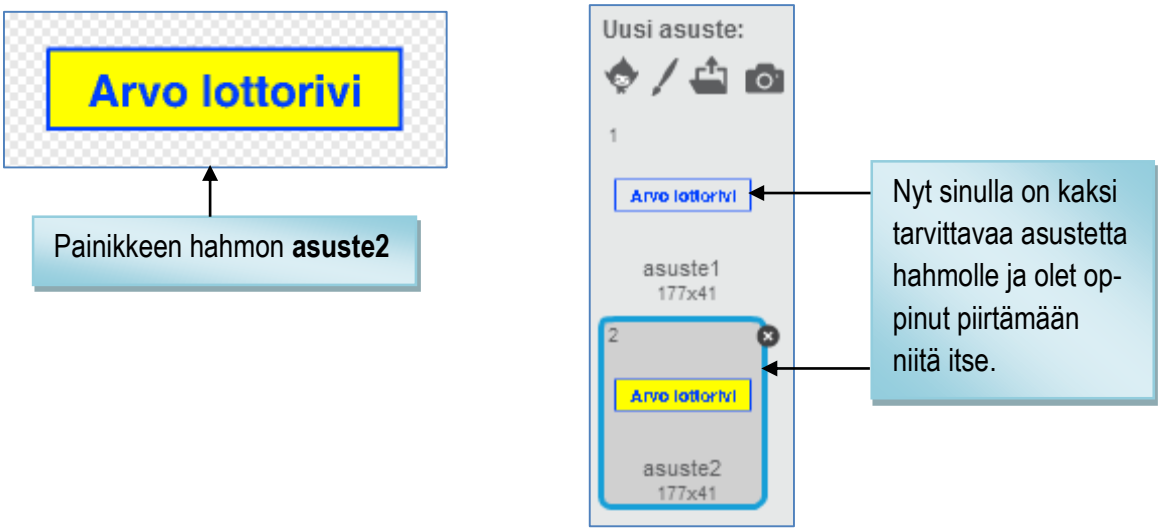
Koodialueelle aukeaa **asusteet-välilehti**, johon voit piirtää hahmon. Valitse haluamasi väri ja käytä **viiva-** ja **tekstityökaluja** painikkeen piirtämiseksi.




Kopioi piirtämäsi asuste toisen asusteen pohjaksi.

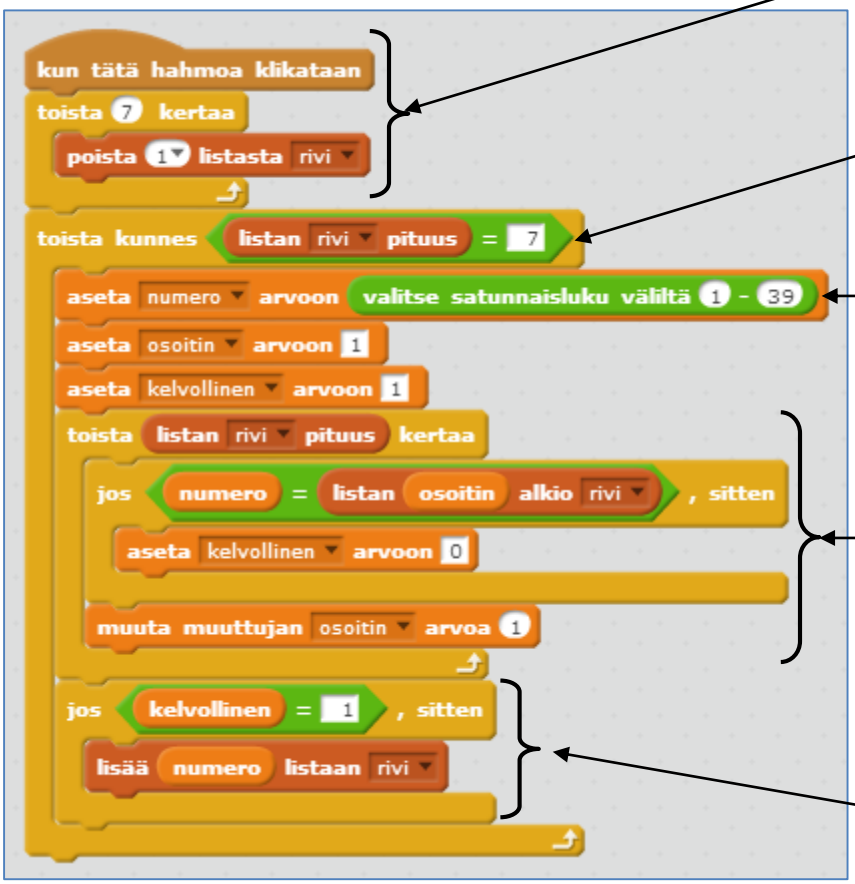



Käytä työkalua ”**Täytä värillä**” ja väritä sillä painikkeen tausta haluamasi väriseksi. Teemme myöhemmin koodin, joka vaihtaa painikkeelle asusteen2, kun hiiriosoitin tulee sen päälle.



Ohjelman koodipalikat:


Kun hiiriosoitin koskettaa hahmoa, vaihdetaan asusteeksi **asuste2**, muuten asusteena on **asuste1**.




Rivin arvonta käynnistyy painiketta (hahmoa) klikkaamalla. Aluksi poistetaan listasta kaikki siellä mahdollisesti olevat luvut.

Toistorakennetta toistetaan, kunnes listassa **rivi** on 7 numeroa (=listan pituus).

Arvotaan muuttujan **numero** arvoksi satunnaisluku väliltä 1-39.

Käydään kaikki listassa olevat alkiot läpi ja tutkitaan, onko jonkin **alkion arvo** yhtä suuri kuin muuttujan **numero** arvo (=arvottu luku). Jos on, asetetaan muuttujan **kelvollinen** arvoksi nolla. Tätä lukua emme voi hyväksyä, koska yksittäinen numero voi esiintyä rivissä vain yhden kerran (ei samoja numeroita useampaan kertaan).

Jos muuttujan **kelvollinen** arvo on **1**, arvottu luku ei ole sama kuin muut luvut listassa **rivi** ja se voidaan lisätä listaan. Lisäyksen johdosta listan **rivi** pituus kasvaa yhdellä.


Kuva esittää näyttämön vasemman yläreunan näkymää. Listassa **rivi** on 7 alkiota ja jokaisen alkion arvona on luku väliltä 1-39. Sama luku ei esiinny useampaan kertaan.

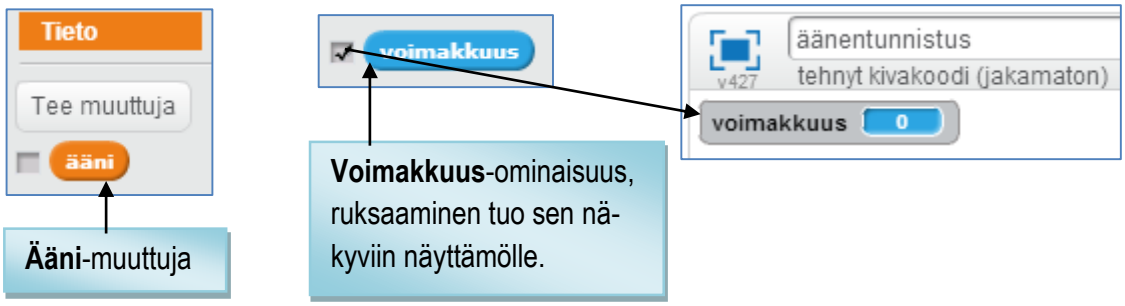
## ÄÄNEEN REAGOIVA OHJELMA

Ohjelman määritys:

Ohjelma tunnistaa mikrofonista tulevan äänen ja reagoi siihen. Jos ääni on heikko, kissa sanoo ”Kuulen heikkoa ääntä”. Jos taas ääni on voimakkaampi, kissa sanoo ”Kuulen selvää ääntä”. Ohjelma muuttaa kissan kokoa havaitun äänen voimakkuuden perusteella. Voimakkaampi ääni tekee kissasta suuremman. Puhekuplan jälkeen kissan koko palautuu entiselleen.

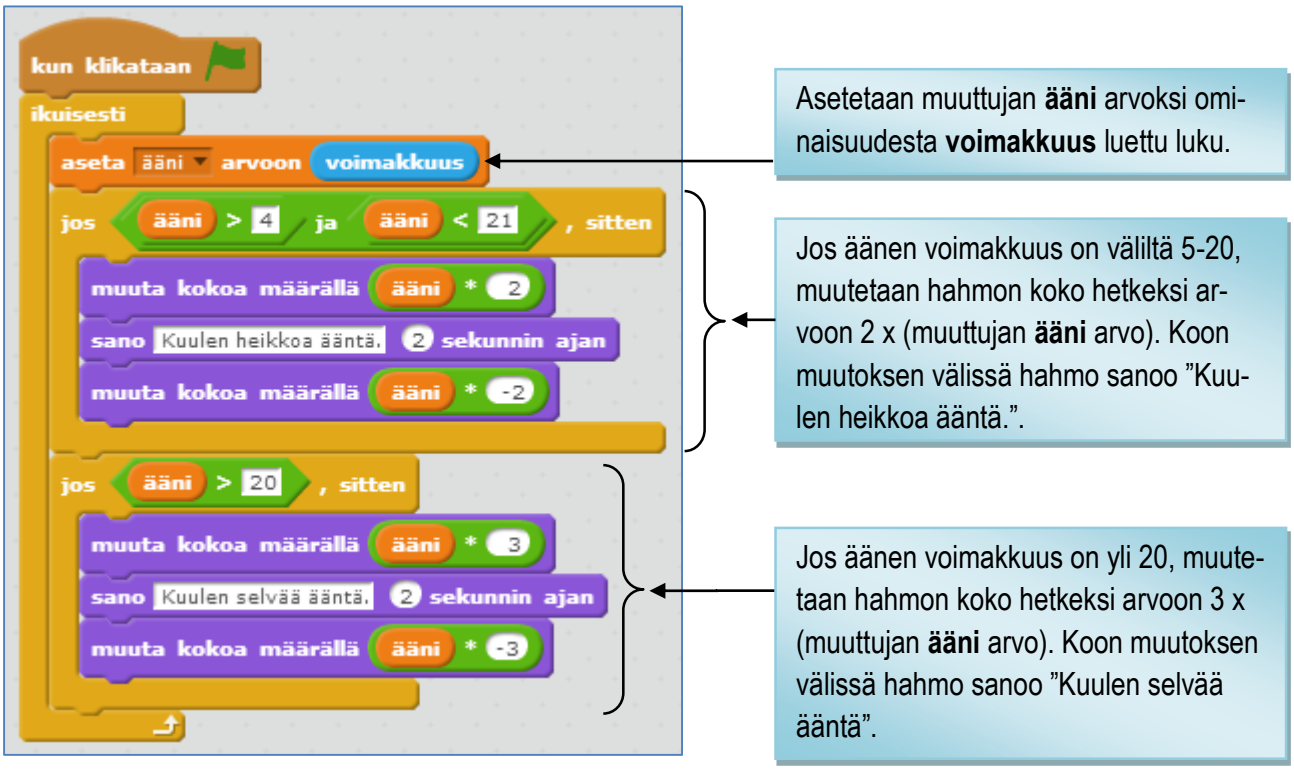
Ohjelman toteutus:

Ohjelmassa tarvitaan mikrofonia tulkitsemaan ympäristön ääniä. Tarvitse yhden muuttujan nimeltä **ääni**. Muuttujan arvon ei tarvitse olla näkyvissä. **Tuntoaisti-osiosta** löytyy ominaisuuspalikka **voimakkuus**. Tästä palikasta voimme lukea mikrofonin havaitseman äänen voimakkuuden lukualueelta 1-100. Esimerkin keskeinen opetusaihe on ääneen reagoivan ohjelman toteutus. Voimakkuus-ominaisuuden arvo tuodaan näkyviin näyttämölle laittamalla ruksi valintaruutuun.




Aina käytettäessä tietokoneen mikrofonia tai kameraa ohjelma kysyy tähän lupaa. Anna lupa klikkaamalla painiketta Allow (salli).

Koodipalikat:



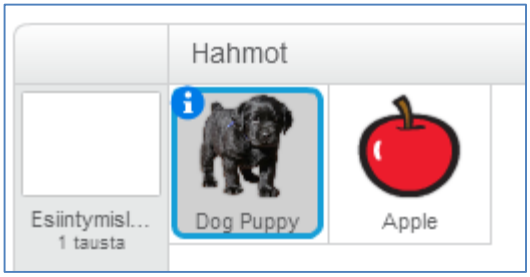

## Videoliikkeeseen reagoiva ohjelma

Ohjelman määrittely:

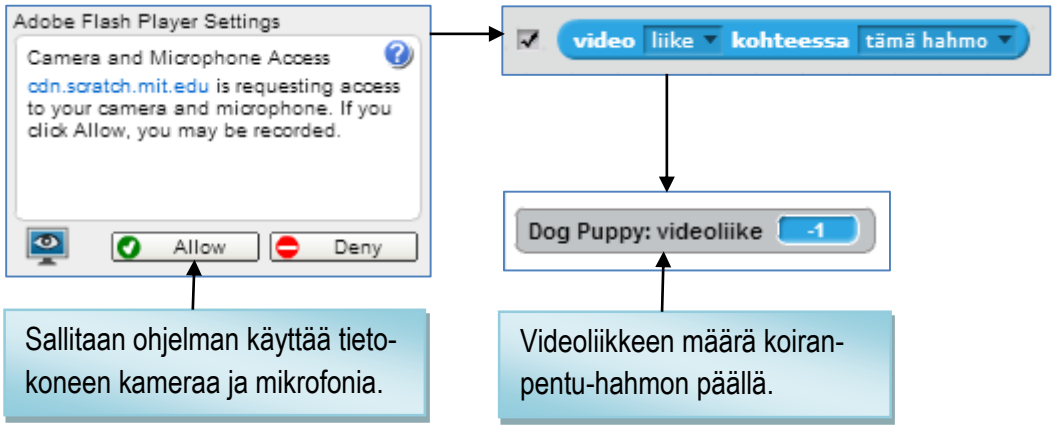
> Näyttämöllä on kaksi hahmoa, koira ja omena. Näyttämön taustalla näkyy tietokoneen kameran kuvaaman videoliikkeen määrä. Kun videoliikkeen määrä koira-hahmon päällä kasvaa yli raja-arvon, koira päästää haukkumisäänen. Vastaavassa tilanteessa omenan kohdalla kuuluu puhe ”Tämä on omena”. Videoliikkeen määrä molempien hahmojen kohdalla on näkyvissä näyttämöllä. Tässä esimerkissä keskeisiä oppimisaiheita ovat videoliikkeen tunnistus ja omien äänien nauhoittaminen hahmojen ääniksi.

Ohjelman toteutus:

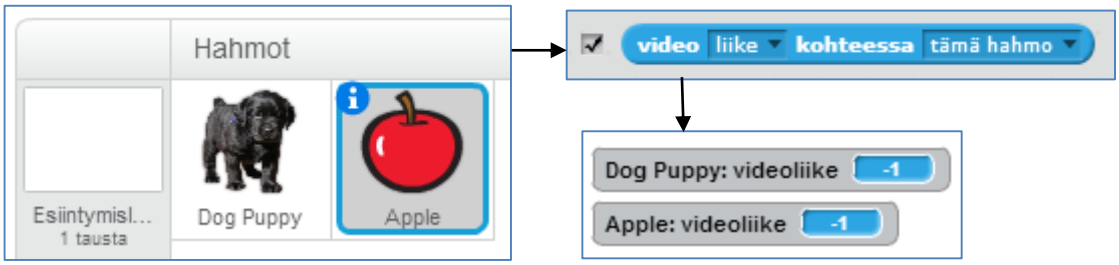
> Tässä projektissa tarvitaan tietokoneeseen liitettyä kameraa, mikrofonia ja kaiuttimia. Poista aluksi kissa-hahmo ja lisää projektiin hahmokirjastosta **koiranpentu** (**Dog Puppy**) ja **omena** (Apple).



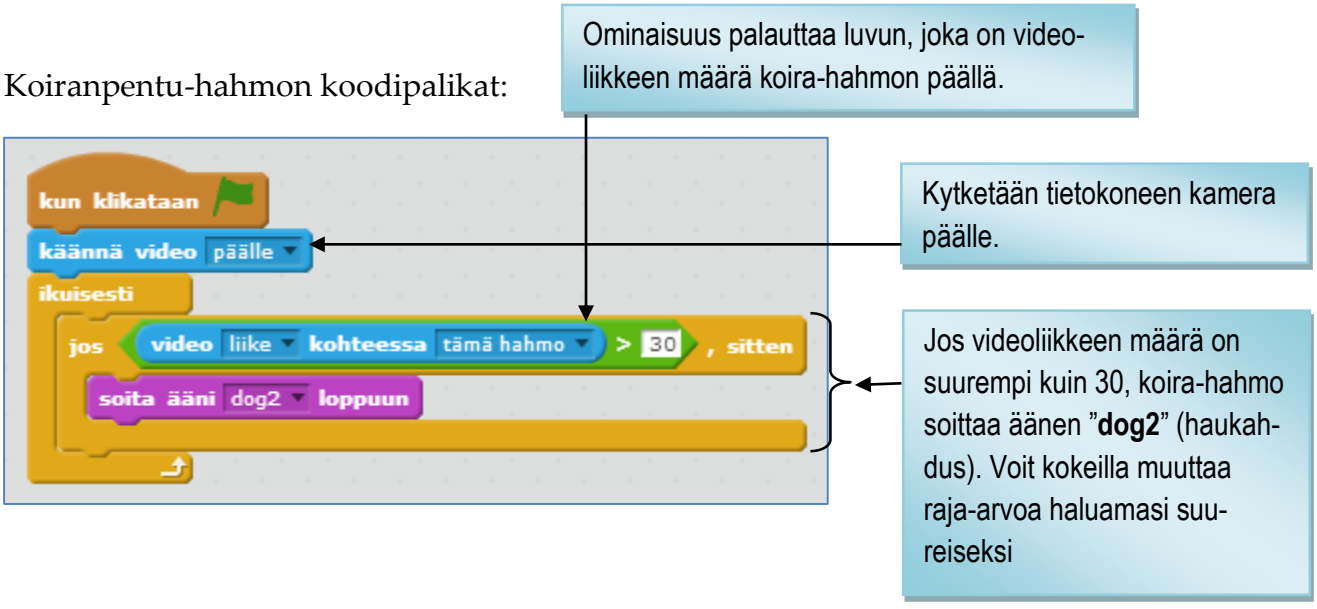
**Tuntoaisti-osiosta** löydät ominaisuuspalikan ”**video <liike> kohteessa <tämä hahmo>**”. Aseta ruksi palikan eteen, niin videoliikkeen määrä tulee näkyviin näyttämölle. Joudut ensin antamaan ohjelmalle luvan käyttää tietokoneesi kameraa ja mikrofonia, valitse **Allow** (salli). Tämän jälkeen ominaisuuden ”**video <liike> kohteessa <tämä hahmo>**” arvo tulee näkyviin näyttämölle.



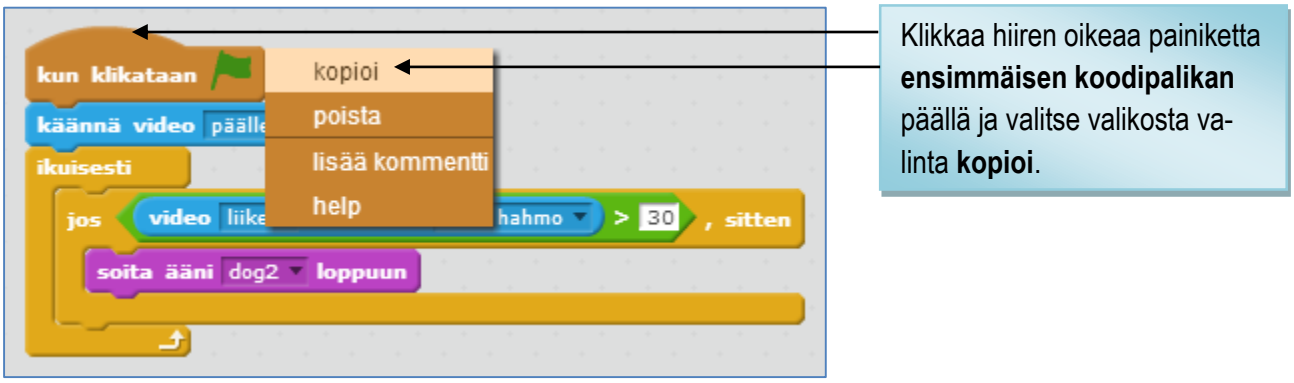

Valitse **Hahmot-ikkunasta** seuraavaksi **omena-hahmo**, ja tuo myös sen videoliikkeen määrä näkyviin näyttämölle.




Koiranpentu-hahmon koodipalikat:



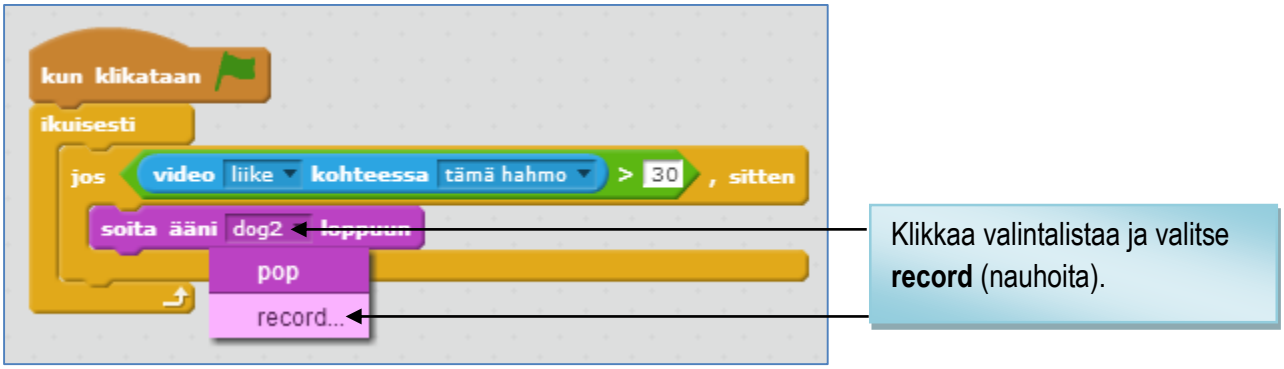
Omena-hahmon koodipalikat ovat hyvin samanlaiset kuin koirankin. Voit muodostaa ne kopioimalla ja raahaamalla.



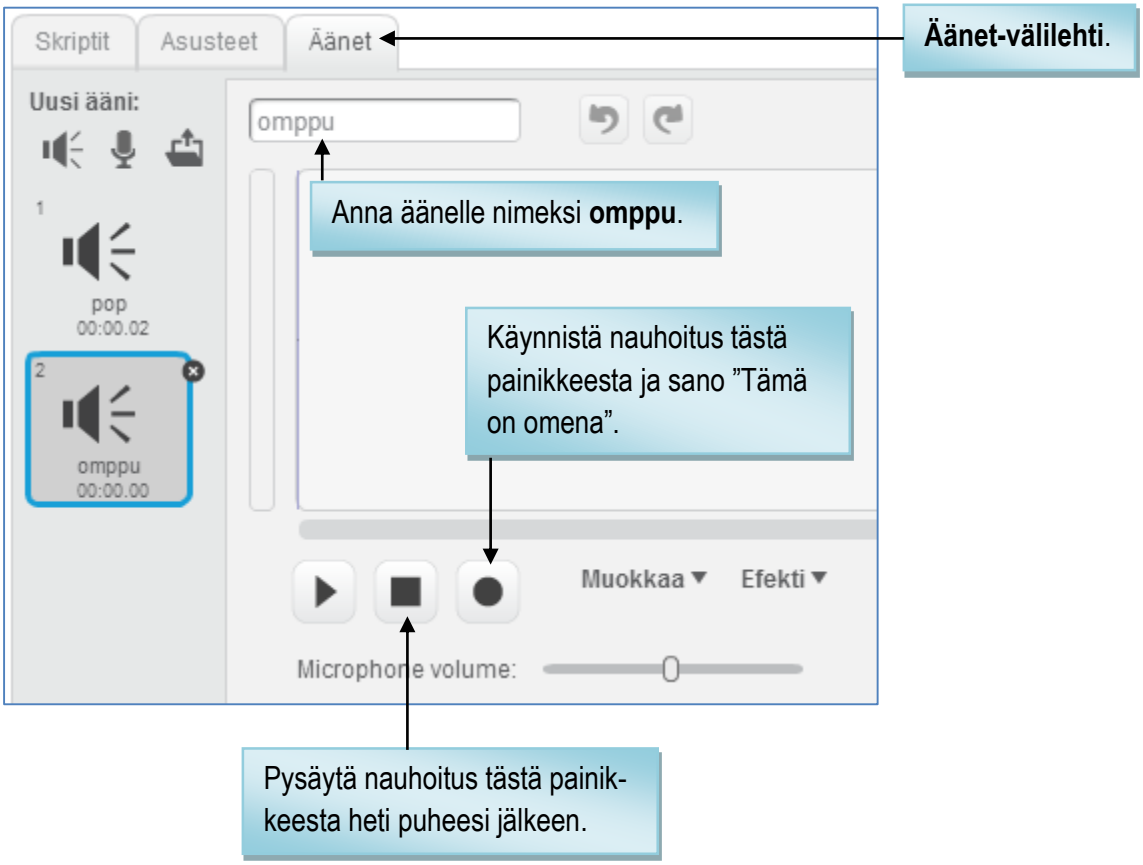

Kaikki koodipalikat kopioituvat, raahaa ne tämän jälkeen **Hahmot-ikkunassa** olevan **omena-hahmon** päälle ja klikkaa hiiren vasenta painiketta. Kaikki nämä koodipalikat siirtyvät nyt **omena-hahmon** koodiksi.


Omenan koodissa ei ole enää tarpeen kääntää videota päälle, joten voit poistaa komentopalikan ”**käännä video päälle**”.

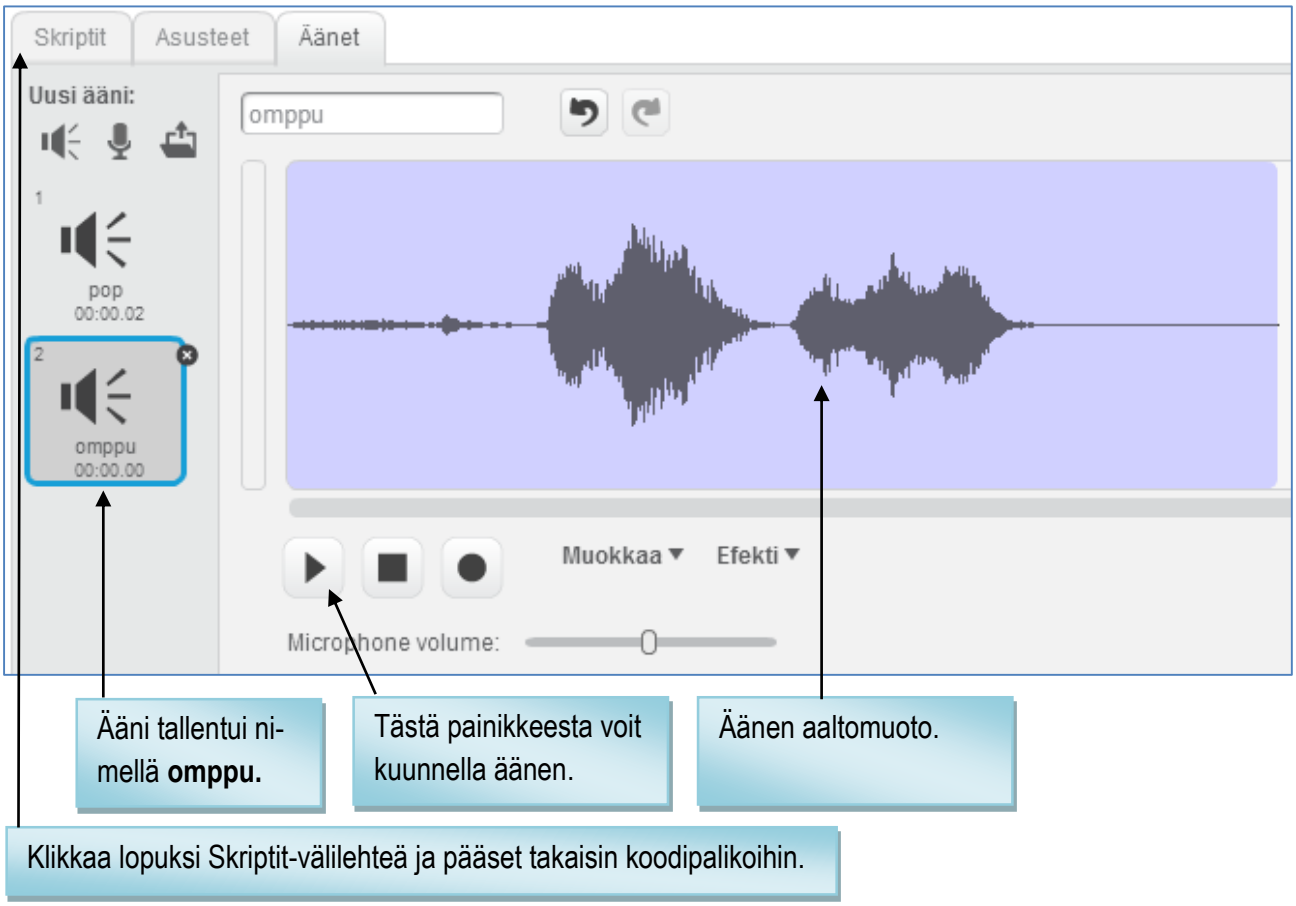
Haluamme, että omena kun omena-hahmo havaitsee hahmon päällä videoliikettä, se soittaa äänitiedoston, jossa sanotaan ”Tämä on omena”. Tällaista tekstiä ei löydy valmiina, joten nauhoitamme sen itse. Klikkaa komentopalikassa ”**soita ääni <ääni> loppuun**” olevaa valintalistaa ja valitse valinta **record** (nauhoita).



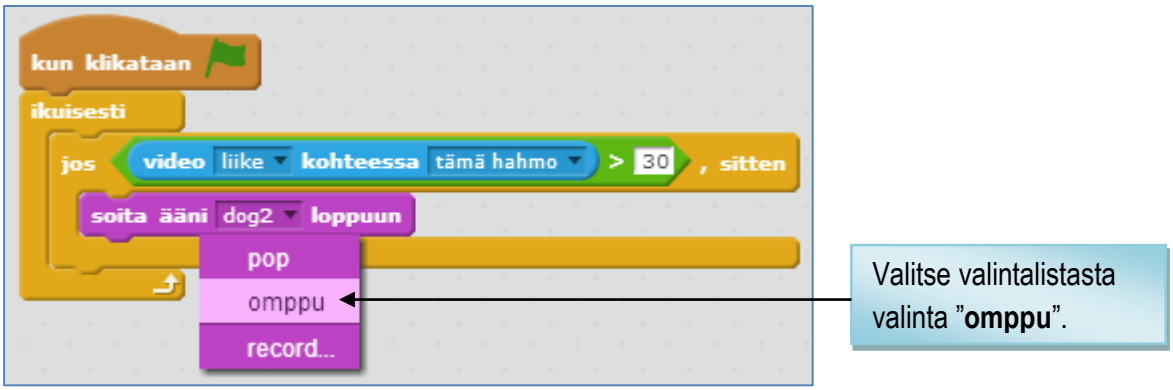

Heti tämän jälkeen **koodialueelle** aukeaa **äänet-välilehti**, jossa voimme nauhoittaa uuden äänen.



Nauhoituksen jälkeen äänen äänimuoto tulee näkyviin ikkunaan.




Vaihda soitettavaksi ääneksi äsken nauhoittamasi ääni ”**omppu**”.




Nyt voit käynnistää ohjelman. Näet tietokoneen kameran ottaman videokuvan näyttämön taustalla. Liikuta kättäsi videokuvassa hahmojen päällä. Koiran pitäisi päästää haukahdus ja omenan puhua nauhoittamasi puhe. Jos jostain syystä et kuule ääntä, katso millaisia arvoja ohjelma esittää havaitulle videoliikkeelle ja pienennä tarvittaessa ehdossa olevaa rajaarvona toimivaa lukua **30**.