Sphero labyrintissa

Kohderyhmä: yläkoululaiset 8. – 9. luokat (yksi luokka)
Esitiedot: -
Taustalla oleva matematiikka: funktiot ja muuttujat, kulma
Poikkitieteellisyys: JavaScript-ohjelmointi
Ajankäyttö: 3 h (2 h Fablabissa)
Opetustilat: luokkahuone ja Fablab

Tavoitteet

Opetella JavaScript-ohjelmointikieleen perustuvaa lohko-ohjelmointia, ymmärtää funktioiden ja muuttujien merkitys ohjelmoinnissa sekä kehittää ryhmätyöskentely- ja ongelmanratkaisutaitoja hauskalla tavalla Sphero-pallon avulla labyrintissa liikkuen.

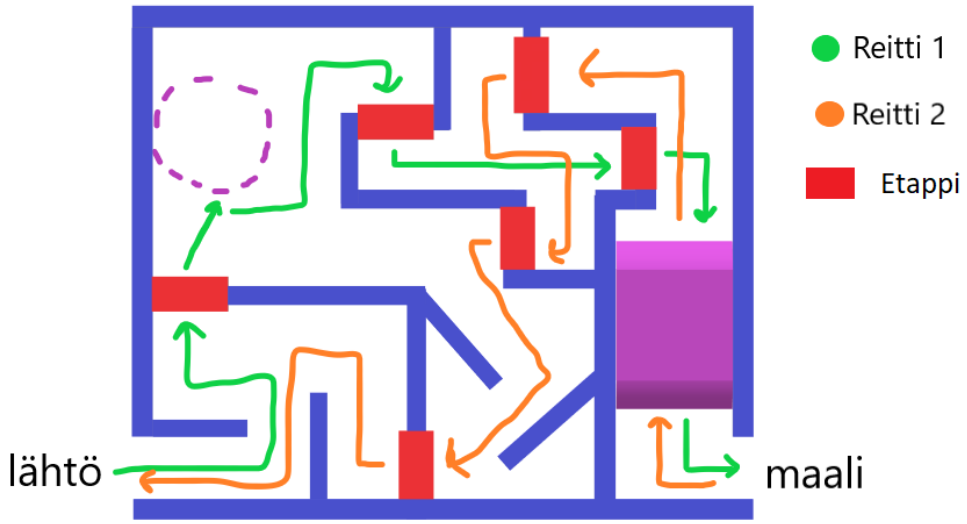
Sphero-robottipallo on puhelimeen tai tablettiin ladattavalla Sphero Edu -sovelluksella etänä ohjattava pieni pallo. Ohjelmointitapoina sovelluksessa on valittavina pallon reitin piirtäminen, suomenkielinen lohko-ohjelmointi tai oikean JavaScriptin kirjoittaminen. **Tässä projektissa hyödynnetään lohko-ohjelmointia.** Ohjelmoitaessa lohkoja apuna käyttäen ohjelman rungon hahmottaminen on helpompaa. Tarvittaessa oppilaat näkevät kirjoittamansa ohjelman myös valikon takaa oikeana koodina. Robottia voidaan ohjata myös sovelluksesta löytyvällä kauko-ohjaimella, mutta tämä tyyli ei opeta projektiin liittyviä tavoitteita.

Kuvaus projektista

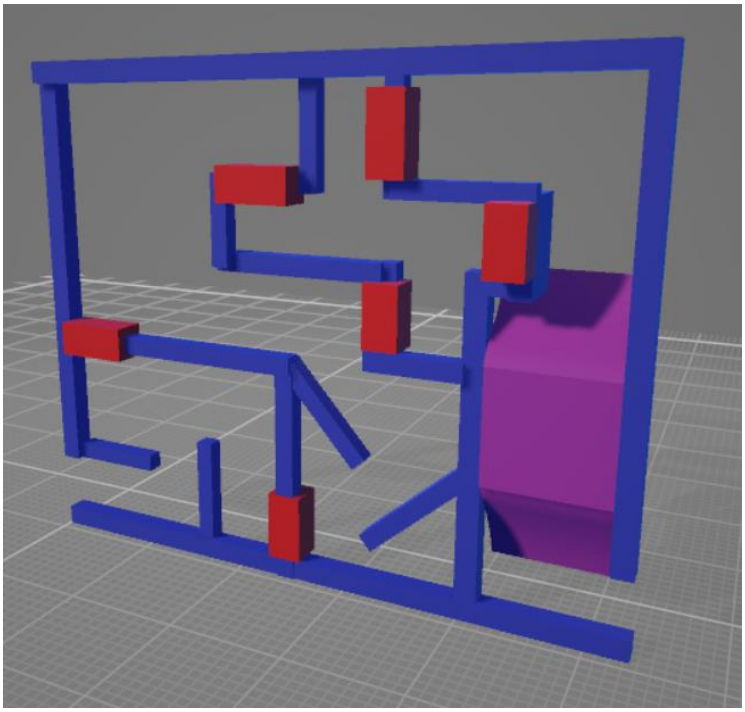
Projektiin kuuluu työskentelyä sekä koulun luokkahuoneessa että yliopiston Fablabissa palapelimallin mukaan:

1. Luokkahuoneessa: Oppilaat jaetaan ryhmiin (jako neljään). Ryhmissä tutustutaan ohjelmoinnin termistöön ja selvitetään, miten ne linkittyvät matematiikkaan (ks. oppilaan ohje).
2. Fablabissa: Oppilaat ryhmittyvät uudelleen ja siirtyvät labyrinttiin porrastetusti. Labyrintissa ratkotaan tehtäviä ohjelmoimalla Sphero-robottipalloa **lohko-ohjelmoinnilla** (2 h). Valmiita tehtäviä löytyy liitteistä.
3. Palautekeskustelua ja pohdintaa. Voidaan esimerkiksi käydä läpi onnistuneita ratkaisuja. Oppilaat tekevät myös itse- ja vertaisarvioinnin ryhmänsä työskentelystä.
4. Opettaja käy läpi oppilaiden tekemät ohjelmat ja kirjoitetut arvioinnit sekä antaa sanallisen palautteen oppilaille.

Esimerkkikuvat (2D ja 3D) labyrintista ja sen rakenteeseen perustuvat 6 erilaista matemaattista ohjelmointiongelmaa:



Kuva 1: Esimerkkikuva Sphero-pallolle rakennettavasta labyrintista



Kuva 2: Kuvan 1 labyrintin 3D-luonnos, koko n. 1 m^2

Labyrintissa ideana on liikkua pelkästään ohjelmoimalla Sphero-palloa lohkoilla eli kauko-ohjaus on kiellettyä. Ensimmäisenä on tarkoitus selvitä labyrintin kolmesta helpommasta ongelmasta, jotka läpäisemällä päästään maaliin. Halutessaan ja ajan riittäessä ryhmä voi kulkea labyrintin uudelleen vastakkaiseen suuntaan, jolloin eteen tulee hieman haastavampia

ohjelmointiongelmia. Kaikkiaan ryhmillä on yhteensä kaksi tuntia aikaa suorittaa labyrinttia ja sen tehtäviä. Ideana on kuitenkin, että kaikki pääsevät labyrintin ainakin kerran läpi päästessään maaliin, eikä minkään ryhmän ole tarkoitus niin sanotusti jäädä jumiin labyrinttiin. Liitteenä on **reitin 1 eli helpomman reitin** ja **reitin 2 eli haastavamman reitin** esteiden ohjelmointiongelmien ideat ja ratkaisut Sphero Edu -sovelluksella lohkoilla sekä JavaScript-ohjelmointikielellä esitettyinä.

HUOM! Yllä olevaa labyrinttia ei ole rakennettu tai testattu, joten se toimii vain mallina ja oma labyrintti kannattaa testata ennen projektin teettämistä oppilailla. Liitteistä löytyvät malli vastaukset on suhteellisen helppo muokata uuteen labyrinttiin, jos eri etappien tehtävät pidetään samoina, sillä vain reitit etapeille ja mäki muuttuvat.

Projektia tukeva ohjelmistotermistö

Ohjelmointi suoritetaan Sphero Edu -sovelluksella, jossa kaikki toiminnot on kirjoitettu suomen kielellä helposti sijoitettavissa lohkoissa. Tärkeintä ei siis ole ohjelmointikielen tarkan kirjoittamisen osaaminen vaan palasten toiminnan ymmärtäminen. Projektissa käytetään muutamia perusohjelmointitoimintoja, joiden ymmärtämiseen opettaja voi ohjata oppilaita alkuteorialla.

Teoria on jaettu neljään osaan, jokaiselle ryhmälle oma osansa. Osien ohjelmatoiminnoista on laadittu johdattelevia kysymyksiä, joihin vastaaminen saa oppilaat selvittämään, mitä ne tarkoittavat ja miten niitä käytetään. Vastauksia etsiessä oppilaat voivat käyttää internettiä, käytettävää ohjelmaa ja mahdollisesti opettajan kokoamaa teoriapakkausta. Kun jokainen ryhmä on tutustunut omaan osa-alueeseensa, tehdään uusi ryhmäjako, jossa on oppilaita jokaisesta osa-alueesta. Tämä kannustaa tiedon jakamiseen oppilaiden välillä.

Aiheet, joista oppilailla tulee olla tietoa ennen ohjelmoinnin aloittamista:

- silmukka ja sen eri rakenteet
- muuttujat
- funktiot
- ohjelmoinnin laskutoimitukset

Johdattelevat kysymykset ja suurpiirteiset vastaukset, joihin tulisi päätyä:

Silmukat

- Mikä on silmukan pääidea?
    - Toistaa silmukassa olevia lohkoja määrätyn verran kertoja tai kunnes, jokin ehto toteutuu tai ei enää toteudu.
- Miten silmukka toimii?

    - Silmukka käy sen sisällä määritellyt ohjelmointikoodit järjestyksessä ylhäältä alas ja päädyttyään viimeiseen aloittaa joko alusta tai lopettaa silmukan sen toistomäärittelyn perusteella.
- Mistä välilehdestä ja millä värillä silmukat löytyvät lohko-ohjelmoinnissa?
    - Ohjauskomennot, violetti.
- Milloin kannattaa käyttää ”x kertaa”/ ”silmukka kunnes”/ ”jos __ niin”/ ”jos __ niin, muuten” -silmukkaa? Selitä esimerkiksi silmukan idean kautta.
    - “X kertaa” silmukkaa käytetään, kun tiedetään tarkasti montako kertaa jokin pitää toistaa.
    - “Silmukka kunnes” silmukassa silmukkaa toteutetaan, kunnes jokin totuusarvo käy toteen. Tämä voi olla hyödyllistä, esimerkiksi jos jonkin muuttujan arvoa kasvatetaan silmukan sisällä ja silmukka tulee lopettaa, kun muuttuja on saavuttanut tietyn arvon.
    - “Jos __ niin” silmukka toteutetaan vain, jos sen totuusmääritelmä käy toteen, muuten se ohitetaan koodissa.
    - “Jos __ niin, muuten” silmukka toimii hyvin samaan tapaan kuin “Jos __ niin” silmukka, mutta tässä tilanteessa on määritelty myös “muuten” tapahtuma, jos ensimmäinen arvo ei käy toteen niin silmukassa siirrytään ”muuten” kohtaan ja toteutetaan sen koodipätkät.
- Mitä hyötyä silmukoista on koodia kirjoittaessa?
    - Nopea, selkeä, lyhentää koodia huomattavasti.
- Mitä voit sijoittaa Sphero Edu ohjelmassa ”Tosi/epätosi” kohdan tilalle?
    - Totuusarvon.

Muuttujat

- Mikä on muuttuja ohjelmoinnissa ja matematiikassa?
    - Jonkin tyyppinen arvo, joka voidaan määrätä ja jota voidaan käsitellä. Matematiikassa muuttujaa merkataan yleensä esim. x, y, z symboleilla.
- Miten muuttuja kannattaa nimetä?
    - Esimerkiksi tyyppiarvoonsa viitaten.
- Mitkä ovat Sphero Edu -ohjelman mahdolliset muuttujan tyypit, ja mitä näillä muuttujan tyypeillä tarkoitetaan?
    - Merkkijono: Tallennettu merkkijonoarvo. Arvot voivat olla sanoja tai lauseita, kuten ”Hei maailma!”
    - Numero: Tallennettu numeroarvo. Arvot voivat olla kokonaislukuja tai liukulukuarvoja (numeroita, joissa on desimaaleja) kuten 1, 2, 3 tai 2,5; 10,8
    - Looginen: Tallennettu looginen arvo. Arvo voi olla tosi tai epätosi.
    - Väri: Tallennettu väriarvo. Arvot koostuvat eri väreistä.
- Miten muuttuja voidaan sijoittaa koodiin Sphero Edu ohjelmassa?

    - Jokaisella muuttujatyypillä on oma muotonsa. Ohjelmassa on eri toimintoja, joihin voi asettaa eri tyyppiarvoja. Sopivan tyyppiarvon tunnistaa oikeasta muodosta. Ilman toimintoon sijoitusta muuttujalla ei ole merkitystä koodissa.
- Miten muuttujan arvoa voidaan muuttaa myöhemmin?
    - Aseta ”muuttuja” arvoon ”x” toiminnolla, joka löytyy muuttujan luonnin jälkeen Muuttujat valikosta.

Funktiot

- Mikä on funktio ohjelmoinnissa ja matematiikassa?
    - Ohjelmoinnissa: Funktio on käytännössä koodinpätkä, jota voi kutsua muualta ohjelmakoodista. Sillä on tietyt määrätyt toiminnot, jotka se toteuttaa joka kerta.
    - Matematiikassa: Funktio eli kuvaus, joka selittää muuttujan riippuvuutta toisesta muuttujasta.
- Miten funktio kannattaa nimetä?
    - Esimerkiksi toiminnon perusteella. Itselle ja muille selkeällä tavalla. Esim. “Pluslasku” -funktio, joka saa parametreina pari numeromuuttujaa ja jotka se plussaa yhteen.
- Miten funktio helpottaa ohjelmointia?
    - Kokonaisohjelman selkeyttäminen.
    - Jos funktiota tarvitaan useamman kerran, niin se on helppo kutsua yhdellä komennolla.
- Mitä ovat parametrit?
    - Parametrit ovat muuttujia, jotka määritellään funktion ylimmällä rivillä funktion nimen jälkeen olevien sulkujen sisällä. Kun funktiota kutsutaan, sen parametreille annetaan arvot kutsuvaiheessa. Funktio käyttää parametrejä tehdessään toimintonsa.
- Onko tiettyä määrää, jota parametreja pitäisi asettaa?
    - Ei, parametreja ei tarvitse välttämättä olla laisinkaan. Jos niitä tarvitaan esimerkiksi laskua varten, tarvitaan vain laskuun käytettävät parametrit ja kaikki muu on turhaa ja tulee jättää pois.
- Mitä funktioon tulisi lisätä viimeisenä?
    - Lopetus toiminto: lopeta ohjelma/lähetä saatu arvo pääohjelmaan käytettäväksi.

Laskutoiminnot

- Millä merkeillä saat suoritettua nämä laskutoimitukset:
    - Pluslasku: +
    - Miinuslasku: -
    - Kertolasku: *

    - Jakolasku: /
    - Potenssi: ^
    - Prosentti: %
- Mitä tarvitset, jotta voit suorittaa laskutoimintoja Sphero Edu –lohko-ohjelmoinnissa?
    - Operaattoreita, mahdollisesti tarvittaessa muuttujia.
- Mihin voit sijoittaa operaattoreita Sphero Edu -lohko-ohjelmoinnissa?
    - Totuusarvoihin
    - Aseta ”muuttuja” arvoon ”x” toimintoon
    - Laskutoimitus-operaattorit voidaan sijoittaa niihin kohtiin mihin numeronkin voi sijoittaa, mutta siitä ei välttämättä ole hyötyä.
- Mitä nämä merkinnät tarkoittavat ohjelmoinnissa?
    - === Vasemman arvon täytyy olla yhtä suuri kuin oikean
    - !== Vasemman arvon täytyy olla erisuuri kuin oikean
    - < Vasemman arvon täytyy olla pienempi kuin oikean
    - <= Vasemman arvon täytyy olla pienempi tai yhtä suuri kuin oikean
    - > Vasemman arvon täytyy olla suurempi kuin oikean
    - >= Vasemman arvon täytyy olla suurempi tai yhtä suuri kuin oikean

Arviointi

Projekti arvioidaan asteikolla hyväksytty/hylätty. Kullekin ryhmälle annetaan lisäksi sanallinen arviointi, jossa otetaan huomioon oppilaiden antamat itse- ja vertaisarviot (ks. oppilaan pohja). Tässä on esimerkkikysymyksiä, joihin sanallinen arviointi voisi vastata:

Kuinka oppilaat toimivat ryhmänä? Osallistuivatko kaikki ryhmän jäsenet? Miten oppilaat keskustelivat keskenään? Kuinka ohjelmointi sujui? Onnistuivatko oppilaat ohjelmoimaan lyhyitä ohjelmia?

Eriyttäminen

Projektin ensimmäinen osa eli ohjelmoinnin termistöön tutustuminen ei vaadi eriyttämistä.

Labyrinttia varten ryhmät sekoitetaan uudelleen siten, että jokaisessa ryhmässä on oppilaita kaikista aiemmista ryhmistä. Jos joukossa on oppilaita, joille JavaScript-ohjelmointikieli on entuudestaan tuttu, esimerkiksi harrastuneisuuden takia, kyseiset oppilaat voivat halutessaan muodostaa keskenään ryhmän ja kokeilla labyrintin ratkaisemista ilman lohkoja eli suoraan JavaScript-kielellä.

Jos oppilaat pääsevät reitin 1 läpi varatussa ajassa, he voivat halutessaan ratkaista labyrintin toiseen suuntaan reittiä 2. Tehtävät on jaoteltu vaikeustason mukaan siten, että tehtävät 1 – 3 ovat helpompia ja tehtävät 4 – 6 haastavampia.

Liitteet

Linkki Sphero-pallon valmistajan englanninkielisille sivuille:
https://www.sphero.com/education/

Sphero Edu sovelluksen verkkoversio

https://edu.sphero.com/

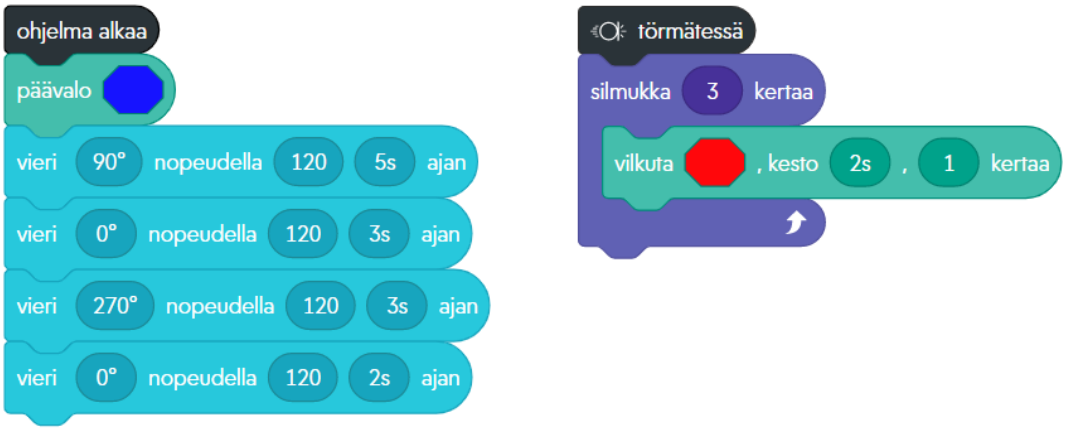
Liite 1: Esimerkki labyrintin tehtävistä ratkaisuineen

Liite 1:

Tehtävä 1:

Pallolle asetetaan tietty väri (muu kuin punainen). Ohjelmoidaan pallo liikkumaan ensimmäiselle esteelle. Törmätessään esteeseen pallon pitää vilkuttaa 3 kertaa punaista valoa. Punainen lanka: miten pallo saadaan väistämään esteitä (kulmat, nopeus ja etenemisaika vaikuttavat -> koordinaatisto), silmukka

Ratkaisu tehtävään 1:

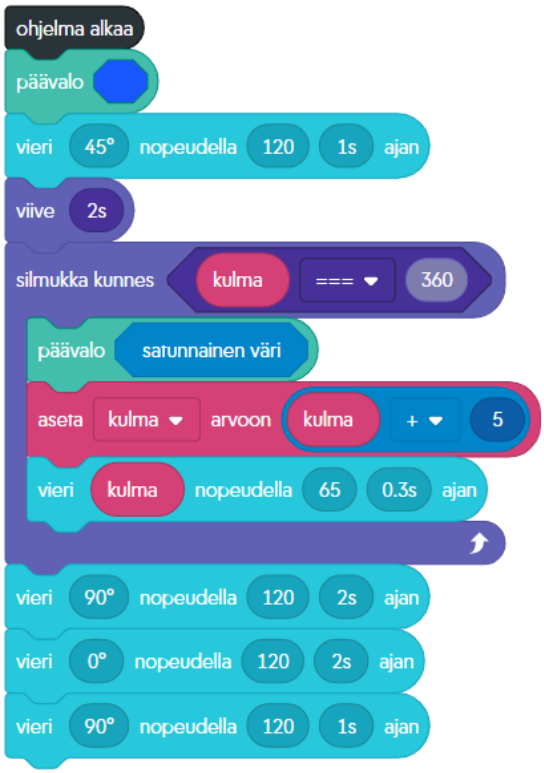



```
async function startProgram() {
    setMainLed({ r: 22, g: 19, b: 255 });
    await roll(90, 120, 5);
    await roll(0, 120, 3);
    await roll(270, 120, 3);
    await roll(0, 120, 2);
}

async function onCollision() {
    for (var _i0 = 0; _i0 < 3; ++_i0) {
        await strobe({ r: 255, g: 7, b: 11 }, 2, 1);
        await delay(0.025);
    }
}
registerEvent(EventType.onCollision, onCollision);
```

Tehtävä 2:

Pallo pitää jälleen ohjelmoida etenemään seuraavalle esteelle. Matkan varrella pallo pitää kuitenkin ohjelmoida tekemään täysi ympyrä, jonka aikana sen väri vaihtelee satunnaisesti. Värin vaihtelun avulla erotetaan, että tällöin pallo suorittaa ympyrää. Punainen lanka: hyödynnetään edellisen kohdan silmukkaa, mutta silmukka pitää nyt rakentaa kulman vaihtelun perusteella, jotta pallo liikkuu täydet 360 astetta. Valmiista vaihtoehdoista ei löydy kulmaa, joten se pitää luoda uutena muuttujana. -> muuttuja käsite ja silmukka, jossa arvoa kasvatetaan.

Ratkaisu tehtävään 2:




```
var kulma = 0;


async function startProgram() {
    setMainLed({ r: 24, g: 86, b: 255 });
    await roll(45, 120, 1);
    await delay(2);
    while (!(kulma === 360)) {
        setMainLed(getRandomColor());
        kulma = kulma + 5;
        await roll(kulma, 65, 0.3);
        await delay(0.025);
    }
    await roll(90, 120, 2);
    await roll(0, 120, 2);
    await roll(90, 120, 1);
}
```

Tehtävä 3:

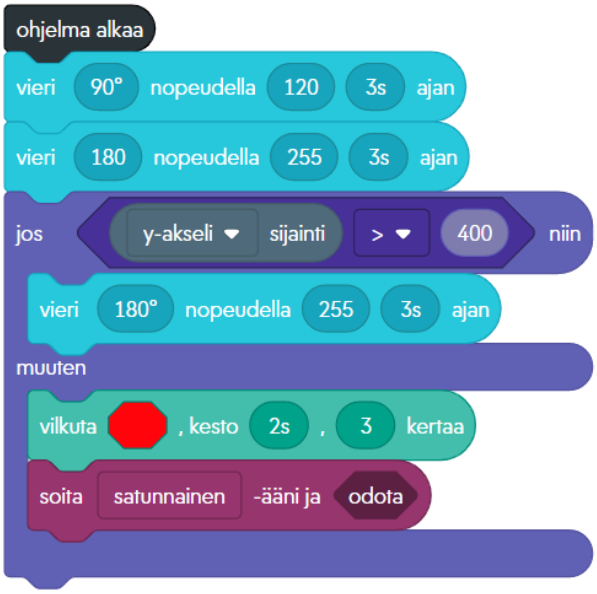
Tässä kohdassa palloa ei ole tarkoitus ohjelmoida liikkumaan vieri -komennon avulla, vaan ohjelman suoritus perustuu funktioon. Funktio itsessään pitää sisällään pallon liikuttamiseen tarvittavan komennon. Pallon on tarkoitus soittaa jokin satunnainen ääni saapuessaan seuraavalle esteelle. Myös ääni -toiminto sisällytetään funktioon. Punainen lanka: Ymmärretään perusidea funktiosta ohjelmoinnissa, ja siitä miten ne pitää rakentaa koodiin.

Ratkaisu tehtävään 3:


```
async function startProgram() {
    await eteenpain();
}

async function eteenpain() {
    await roll(90, 120, 5);
    await Sound.play(true);
    exitProgram();
}
```

Näiden jälkeen on ratkaistu kaikki kolme helpomman radan estettä. **Pallo ei kuitenkaan vielä tässä vaiheessa ole maalissa, vaan sen pitää vielä ylittää mäki.** Alla on esitetty tähän ratkaisu. Pallon on tarkoitus sensoreillaan tunnistaa y-akselin muutos, jolloin se ymmärtää myös laskeutua alas mäeltä. Mikäli pallo tarvitsee suuremman nopeuden ylittääkseen mäen, se ilmoittaa tämän soittamalla satunnaisen äänen ja vilkuttamalla punaista valoa. Punainen lanka: jos/kun -silmukka tutuksi, pallon sisäisiin sensoreihin tutustuminen ja niiden hyödyntäminen koodissa.

Ratkaisu mäen ylittämiseen:




```
async function startProgram() {
    await roll(90, 120, 3);
    await roll(180, 255, 3);
    if (getLocation().y > 400) {
        await roll(180, 255, 3);
    } else {
        await strobe({ r: 255, g: 4, b: 10 }, 2, 3);
        await Sound.play(true);
    }
}
```

➔ PALLO PÄÄSI MAALIIN 😊

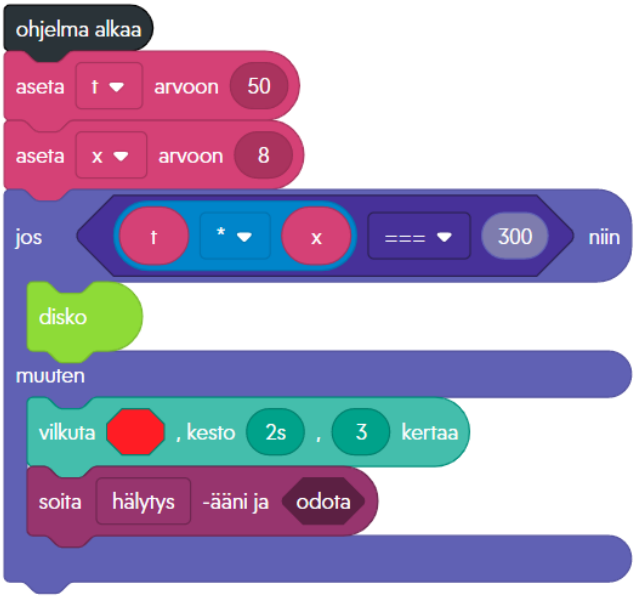
Mikäli opiskelijoilla on aikaa jäljellä, he voivat lähteä ratkaisemaan astetta hankalampia tehtäviä (eriyttävä tehtävä). Nämä tehtävät 4–6 on esitetty alla ratkaisuineen. Ideana on hyödyntää samoja teemoja kuin aiemmissa tehtävissä, mutta koodien rakenteet ovat nyt monimutkaisempia. HUOM! ratkaisu mäen ylitykseen on yllä, pallo pitää kuitenkin kalibroida ”vastakkaiseen suuntaan” ennen aloitusta ja suunnat pitää muuttaa oikeiksi.

Lisätehtäviä:

Tehtävä 4:

Saavuttaessa neljännelle esteelle oppilaiden tulee muodostaa yhtälö, joka on tosi, jotta pallo pääsee jatkamaan matkaa. Mikäli yhtälö on muodostettu oikein, pallo suorittaa ”disko”-funktion, jossa se pyörii viidesti ympäri vaihtaen samalla valoaan. Mikäli muuttujille on annettu virheelliset arvot, pallo vilkuttaa punaista valoa ja hälyttää. Punainen lanka: muuttujien ja funktioiden yhdistäminen, jos/muuten –rakenne.

Ratkaisu tehtävään 4:

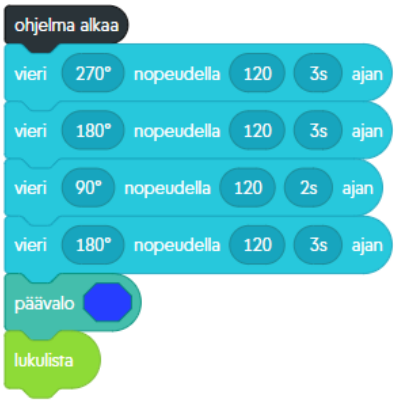


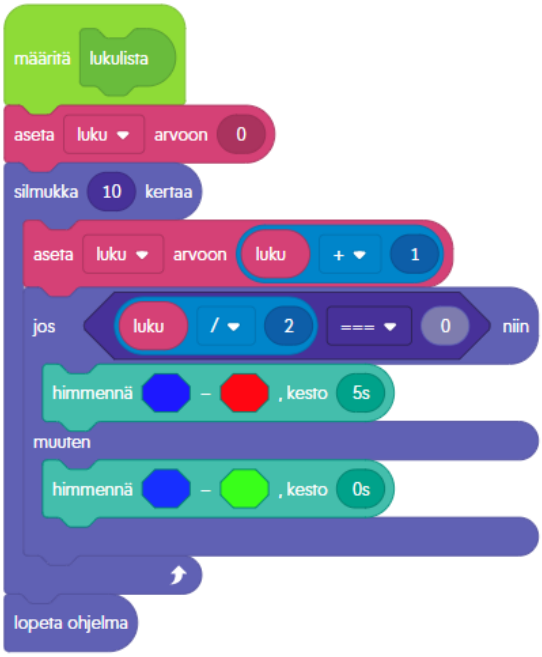

```
var t = 0;
var x = 0;


async function startProgram() {
    t = 50;
    x = 8;
    if (t * x === 300) {
        await disko();
    } else {
        await strobe({ r: 255, g: 28, b: 36 }, 2, 3);
        await Sound.Mechanical.Alarm.play(true);
    }
}

async function disko() {
    for (var _i0 = 0; _i0 < 5; ++_i0) {
        await spin(360, 5);
        setMainLed(getRandomColor());
        await delay(0.025);
    }
}
```

Tehtävä 5:

Seuraavaksi pallo pitää ohjelmoida seuraavan esteen luo, jonka edessä pallo suorittaa jälleen kokonaislukuihin perustuvan laskutoimituksen. Tässä tehtävässä ei kuitenkaan anneta valmiiksi mitään lukuja. Ideana on miettiä jokin kokonaisluvuista muodostuva väli, jonka kaikki alkiot käydään läpi. Jotta varmistetaan kaikkien alkioiden läpikäynti, on pallon vaihdettava väriään parillisten lukujen kohdalla. Lukulistan läpikäyminen toteutetaan funktiolla. Punainen lanka: monimutkaisemman funktion toteuttaminen, jossa kaikki alkiot käydään läpi tarkastellen vielä jos/kun -rakenteella kunkin luvun parillisuus.

Ratkaisu tehtävään 5:



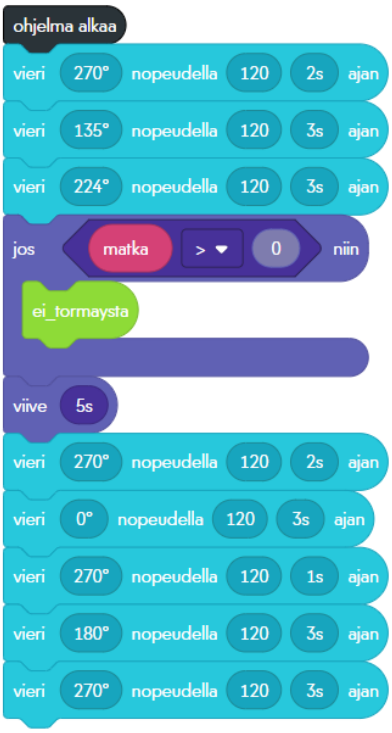


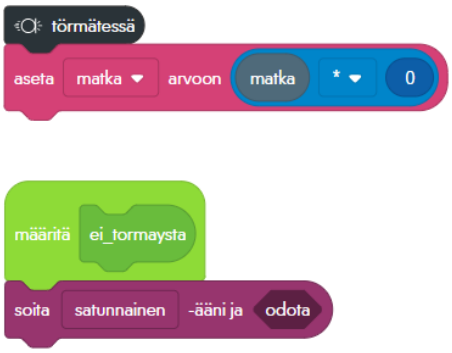

```
var luku = 0;

async function startProgram() {
    await roll(270, 120, 3);
    await roll(180, 120, 3);
    await roll(90, 120, 2);
    await roll(180, 120, 3);
    setMainLed({ r: 38, g: 61, b: 255 });
    await lukulista();
}

async function lukulista() {
    luku = 0;
    for (var _i0 = 0; _i0 < 10; ++_i0) {
        luku = luku + 1;
        if (luku / 2 === 0) {
            await fade({ r: 29, g: 24, b: 255 }, { r: 255, g: 6, b: 18 }, 5);
        } else {
            await fade({ r: 23, g: 47, b: 255 }, { r: 57, g: 255, b: 26 }, 0);
        }
        await delay(0.025);
    }
    exitProgram();
}
```

Tehtävä 6:

Viimeisenä kohtana pallo pitää ohjelmoida kulkemaan mutkaisen radan osion läpi. Kohdatessaan viimeisen esteen pitää oppilaiden toteuttaa jokin itse suunnittelemansa funktio. Funktiossa pallon pitää jollakin tavalla hyödyntää sen omien sensoreidensa muodostamaa dataa. Alla on esitetty yhdenlainen ”malliratkaisu”. Malliratkaisussa funktion toteutuksessa on hyödynnetty pallon sensorien mittaamaa edetyn matkan pituutta. Mikäli pallo törmää seinään eli pallon pujottelu esteissä epäonnistuu, pallon laskema matka nollataan. Lopuksi pallo kuljetetaan ulos labyrintistä.

Ratkaisu tehtävään 6:





```
var matka = 0;


async function startProgram() {
    await roll(270, 120, 2);
    await roll(135, 120, 3);
    await roll(224, 120, 3);
    if (matka > 0) {
        await eitormaysta();
    }
    await delay(5);
    await roll(270, 120, 2);
    await roll(0, 120, 3);
    await roll(270, 120, 1);
    await roll(180, 120, 3);
    await roll(270, 120, 3);
}

async function onCollision() {
    matka = getDistance() * 0;
}
registerEvent(EventType.onCollision, onCollision);

async function eitormaysta() {
    await Sound.play(true);
}
```

➔ PALLO PÄÄSSYT TAKAISIN LÄHTÖPISTEESEEN ELI RADAN VAIKEAMPIKIN OSIO ON SUORITETTU ONNISTUNEESTI 😊

Sphero labyrintissa

Tutustumme ohjelmoinnin termistöön ja opimme, kuinka robotteja voidaan ohjata labyrintissa JavaScript-ohjelmointikieleen perustuvalla lohko-ohjelmoinnilla.

Ohjelmoinnin käsitteistä (luokkahuone):

Tutustutaan ensimmäiseksi hieman ohjelmoinnin eri osa-alueisiin. Vastatkaa ryhmissä oman aiheenne kysymyksiin. Tarvitsette näitä tietoja myöhemmin Sphero-pallon kanssa ratkaistavassa labyrintissa.

Vinkki: Sphero Edu -sovelluksessa voit saada lisätietoja lohkoista ja niiden toiminnoista "lohko-ohje"-toiminnon avulla. Tietokoneella ohje löytyy painamalla lohkosta hiiren oikealla näppäimellä ja sovelluksella pitkään painamalla. Sovelluksen lohko-ohjelmointi perustuu JavaScript-kieleen. Kun teette projektin ohjelmointitehtäviä, voitte kurkata, miltä JavaScript kieli näyttää, painamalla sovelluksessa kolmen pisteen kuvakkeesta ja valitsemalla JavaScript Code.

Kysymyksissä saa käyttää apuna internettiä, puhelimeen ladattavaa Sphero Edu -sovellusta ja tarvittaessa voitte pyytää apua myös opettajalta.

Ryhmä 1: Silmukat

- Mikä on silmukan pääidea?
- Miten silmukka toimii?
- Mistä välilehdestä ja millä värillä silmukat löytyvät lohko-ohjelmoinnissa?
- Milloin kannattaa käyttää "x kertaa"/ "silmukka kunnes"/ "jos __ niin"/ "jos __ niin, muuten" -silmukkaa? Selitä esimerkiksi silmukan idean kautta.
- Mitä hyötyä silmukoista on koodia kirjoittaessa?
- Mitä voit sijoittaa Sphero Edu ohjelmassa "Tosi/epätosi" kohdan tilalle?

Ryhmä 2: Muuttujat

- Mikä on muuttuja ohjelmoinnissa ja matematiikassa?
- Miten muuttuja kannattaa nimetä?
- Mitkä ovat Sphero Edu -ohjelman mahdolliset muuttujan tyypit, ja mitä näillä muuttujan tyypeillä tarkoitetaan?
- Miten muuttuja voidaan sijoittaa koodiin Sphero Edu -ohjelmassa?
- Miten muuttujan arvoa voidaan muuttaa myöhemmin?

Ryhmä 3: Funktiot

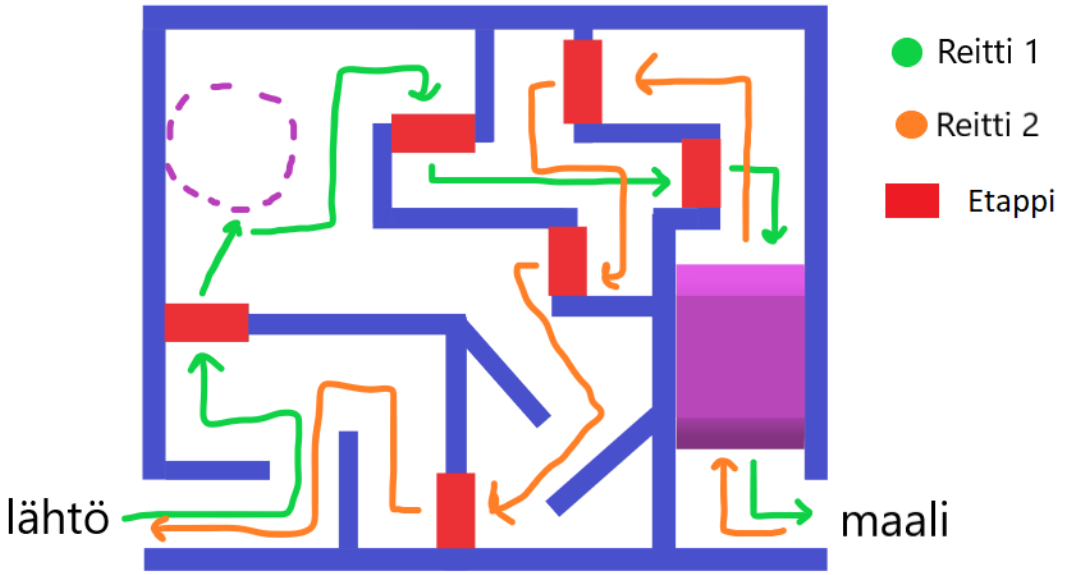
- Mikä on funktio ohjelmoinnissa ja matematiikassa?

- Miten funktio kannattaa nimetä?
- Miten funktio helpottaa ohjelmointia? (Vinkki: saman asian tekeminen monta kertaa)
- Mitä ovat parametrit?
- Onko tiettyä määrää, jota parametreja pitäisi asettaa?
- Mitä funktioon tulisi lisätä viimeisenä?

Ryhmä 4: Laskutoiminnot

- Millä merkeillä saat suoritettua nämä laskutoimitukset?
    - Pluslasku
    - Miinuslasku
    - Kertolasku
    - Jakolasku
    - Potenssi
    - Prosentti
- Mitä tarvitset, jotta voit suorittaa laskutoimintoja Sphero Edu –lohko-ohjelmoinnissa?
- Mihin voit sijoittaa operaattoreita Sphero Edu -lohko-ohjelmoinnissa?
- Mitä nämä merkinnät tarkoittavat ohjelmoinnissa?
    - ===
    - !==
    - <
    - <=
    - >
    - >=

Labyrintti (Fablab):



Jakaudutaan uusiin ryhmiin ja siirrytään labyrinttiin porrastetusti. Labyrintissa ratkotaan seuraavia ohjelmointiongelmia. Labyrintissä on yksi tehtävä per etappi (este). Kunkin ratkaistun tehtävän jälkeen olette yhden etapin lähempänä maalia ja voitte nostaa pallon esteen toiselle puolelle. Tehkää jokaiseen tehtävään oma ohjelmapohja, jonka säästätte lopullista palautekeskustelua ja koontia varten.

Tehtävä 1:

Asettakaa pallon ledille haluamanne väri (älkää kuitenkaan valitko punaista). Ohjelmoikaa pallo ensimmäisen esteen (etapin) luo niin, että se törmää siihen. Törmätessään pallon pitää vilkuttaa punaista led-valoa 3 kertaa 2 sekunnin ajan.

Tehtävä 2:

Ohjelmoikaa pallo seuraavan esteen luo. Matkan varrella haluamassanne kohtaa pallon pitää kuitenkin tehdä täysi ympyrä. Jotta ympyrä erottuu pallon muusta etenemisestä, ohjelmoikaa pallo vaihtamaan väriään satunnaisesti ympyrää kierrettäessä.

Tehtävä 3:

Edellisissä tehtävissä saitte ohjelmoida pallon esteiden luo “vieri”-komennolla. Nyt eteneminen pitää toteuttaa kuitenkin erillisen funktion avulla. Lisätkää funktioon myös toiminto, jolla pallosta saadaan kuulumaan haluamanne ääni sen saapuessa uudelle esteelle.

Mäen ylitys:

Mäen ylityksen voitte jälleen toteuttaa “vieri”-komennoilla. Sphero pallo sisältää useita sensoreita, joiden avulla se mittaa mm. sijaintia ja kaltevuutta. Jokaisen ohjelman suorituksen jälkeen näiden sensorien keräämää dataa voi tarkastella painamalla oikeasta yläkulmasta löytyvää ”Sensor Data”-painiketta. Tässä tehtävässä teidän pitää hyödyntää pallon sensorien mittaamaa y-akselin muutosta eli koordinaatistosta ajatellen pallon reitin varrella tapahtuvia korkeuden muutoksia.

Toteuttakaa mäen ylitystä varten ohjelma, jossa pallo ensin vierii kohti mäkeä. Mikäli pallo pääsee mäen huipulle eli sen y-akselin sijainti kasvaa, pallo jatkaa mäkeä alas. Jos taas pallo ei saavuta mäen huippua, se pysähtyy, vilkuttaa punaista led-valoa kolmesti 2 sekunnin ajan ja antaa äänimerkin. Muuttakaa pallon ”vieri”-komennon nopeuksia, kunnes saavutatte mäen huipun.

Vinkki: Jos/muuten-silmukka

➔ PALLO PÄÄSI MAALIIN 😊 HUIPPUA TYÖTÄ!

Halutessanne voitte jatkaa nyt haastavampiin tehtäviin. Ylittäkää mäki uudelleen, mutta vastakkaiseen suuntaan. Huomioikaa pallon kalibrointi ennen aloitusta.

Tehtävä 4:

Saavuttaessa neljännelle esteelle teidän tulee muodostaa yhtälö, joka on tosi, jotta pallo pääsee jatkamaan matkaa. Mikäli yhtälö on muodostettu oikein, pallo suorittaa ”disko”-funktion, jossa se pyörii viidesti ympäri vaihtaen samalla valoaan. Mikäli muuttujille on annettu virheelliset arvot, pallo vilkuttaa punaista valoa ja hälyttää.

Vinkki: Aloita luomalla kaksi muuttujaa (numero), joille ohjelman alussa asetat tietyt arvot (ja joita voit vaihtaa). Tulon tarkistaminen kannattaa toteuttaa jos/muuten-silmukalla, johon ”disko”-funktio voidaan sijoittaa. Saat pallon pyörimään paikallaan kierto -komennolla. Mieti tarkkaan, miten saat ”disko”-funktion luotua mahdollisimman yksinkertaisesti.

Tehtävä 5:

Seuraavaksi pallo pitää ohjelmoida seuraavan esteen luo, jonka edessä pallo tarkastelee paikallaan ollessaan jälleen kokonaislukuja. Ideana on miettiä jokin kokonaisluvuista muodostuva väli, jonka kaikki alkiot eli luvut käydään läpi. Jotta varmistetaan kaikkien lukujen läpikäynti, on pallon vaihdettava väriään parillisten lukujen kohdalla. Lukulistan läpikäyminen toteutetaan funktiolla.

Vinkki: Valitse pieni väli, esim. 1-10, sillä muuten ohjelman suoritus kestää kauan. Funktiossa kannattaa hyödyntää seuraavia jo aiemmissa tehtävissä esiin tulleita aiheita luomalla esimerkiksi muuttuja, jonka alkuarvoksi asetetaan 0 ja jonka arvoa kasvatetaan jokaisessa silmukan suorituksessa yhdellä. Parillisten lukujen tarkastelu kannattaa toteuttaa jos/muuten -rakenteella suoraan funktion sisään.

Tehtävä 6:

Viimeisenä kohtana pallo pitää ohjelmoida kulkemaan mutkaisen radan osion läpi. Kohdatessaan viimeisen esteen saatte toteuttaa itse suunnittelemanne funktion. Funktiossa pallon pitää jollakin tavalla hyödyntää sen omien sensoreidensa muodostamaa dataa. Tutkikaa eri vaihtoehtoja ja tehkää mieleisenne. Ohjelmoikaa pallo lopuksi ulos labyrintistä.

Vinkki: Funktion toteutuksessa voi hyödyntää esim. pallon sensorien mittaamaa edetyn matkan pituutta. Mikäli pallo törmää seinään eli pallon pujottelu esteissä epäonnistuu, pallon laskema matka nollataan.

➔ PALLO PÄÄSI TAKAISIN LÄHTÖPISTEESEEN 😊 ONNEKSI OLKOON, OLETTE SUPERHYVÄ TIIMI JA SELVISITTE LABYRINTISTA KAHDESTI!

Arviointi:

Kirjoittakaa lyhyt itsearvio omasta työskentelystä sekä vertaisarvio ryhmätyöskentelystä ja projektin onnistumisesta.

Vastatkaa esimerkiksi näihin kysymyksiin:

Itsearviointi (tehdään yksin):

- Opitko jotain uutta ohjelmoinnista?
- Mitä tykkäsit projektista?
- Osallistuitko tasapuolisesti projektin tekoon?
- Kuunneltiinko sinua?
- Mikä oli sinulle helppoa? Entä haastavaa?

Vertaisarviointi (ryhmässä):

- Kuinka toimitte ryhmänä?
- Osallistuivatko kaikki ryhmän jäsenet tasapuolisesti?
- Miten keskustelitte ja delegoitte tehtävänjakoa?
- Kuinka ohjelmointi sujui?
- Onnistuitteko ohjelmoimaan lyhyitä ohjelmia?
- Oliko jokin tehtävä liian vaikea?