



# DOCENTENHANDLEIDING

# Inhoudsopgave

|                                                                                              |     |
|----------------------------------------------------------------------------------------------|-----|
| Docentenhandleiding .....                                                                    | 1   |
| Inleiding .....                                                                              | 4   |
| Het organiseren van uw werk met Scottie Go!.....                                             | 4   |
| Opmerkingen voor docenten .....                                                              | 6   |
| Module I .....                                                                               | 9   |
| Module introductie: commando's.....                                                          | 10  |
| Het spelkarakter en de eerste commando's .....                                               | 11  |
| Het gebruik van cijfers, grotere getallen maken, optellen en aftrekken .....                 | 14  |
| Een draai maken.....                                                                         | 16  |
| Achteruit lopen.....                                                                         | 23  |
| Module II .....                                                                              | 29  |
| Module introductie: objecten oprapen van het spelbord .....                                  | 30  |
| Springen .....                                                                               | 39  |
| Module III .....                                                                             | 55  |
| Module introductie: de REPEAT lus .....                                                      | 56  |
| De eerste lus.....                                                                           | 57  |
| Binnen en buiten een lus werken .....                                                        | 62  |
| Scottie GO! EDU als gereedschap om begrippen uit de wis- en natuurkunde te introduceren..... | 65  |
| Geneste lussen .....                                                                         | 72  |
| Module IV.....                                                                               | 77  |
| Bever als een brug.....                                                                      | 78  |
| Bever in beweging .....                                                                      | 84  |
| Gebruik de REPEAT lus en de SWITCH TO tegels samen .....                                     | 90  |
| Bever maakt een draai .....                                                                  | 95  |
| Module V.....                                                                                | 104 |
| MODULE INTRODUCTIE: interactie met Scottie zijn omgeving - ACTIVATE en PLACE tegels....      | 105 |
| SWITCH TO de bevers en verplaats ijsschotsen.....                                            | 122 |
| Module VI.....                                                                               | 132 |
| MODULE INTRODUCTIE: voorwaardelijk commando - REPEAT WHILE .....                             | 133 |
| De REPEAT WHILE tegel gebruiken.....                                                         | 135 |
| REPEAT WHILE met andere voorwaardes .....                                                    | 144 |
| REPEAT WHILE genest in een lus.....                                                          | 147 |
| Module VII.....                                                                              | 158 |
| MODULE INTRODUCTIE: VARIABLES (VARIABLEN) .....                                              | 159 |
| Asfaltwegen bouwen .....                                                                     | 162 |
| X en Y variabelen.....                                                                       | 170 |
| Variabelen en lussen.....                                                                    | 175 |
| Wiskundige bewerkingen op variabelen .....                                                   | 180 |

|                                                               |     |
|---------------------------------------------------------------|-----|
| Module VIII.....                                              | 187 |
| MODULE INTRODUCTIE: voorwaardelijke commando's IF, ELSE ..... | 188 |
| Actieveld .....                                               | 192 |
| Meerdere voorwaardelijke commando's gebruiken .....           | 204 |
| HEUVEL en HERE tegels .....                                   | 209 |
| Module IX.....                                                | 212 |
| MODULE INTRODUCTIE (A): ELSE IF .....                         | 213 |
| MODULE INTRODUCTIE (B): de ELSE tegel .....                   | 217 |
| MODULE INTRODUCTIE (C): [optioneel] - REPEAT (FOREVER).....   | 220 |
| ELSE .....                                                    | 222 |
| ELSE IF en MARKERINGEN .....                                  | 226 |
| Lopende banden .....                                          | 240 |
| Module X.....                                                 | 248 |
| MODULE INTRODUCTIE: functies .....                            | 249 |
| Functies [extra] .....                                        | 251 |
| Eerste functie, HEUVELS.....                                  | 253 |
| Tekenen.....                                                  | 261 |
| Een tegel meerdere keren gebruiken .....                      | 266 |

# Inleiding

## Het organiseren van uw werk met Scottie Go!

### Zorg dat de werkplek goed verlicht is

Voor het spelen van Scottie Go!, moeten uw leerlingen foto's nemen van de code die ze schrijven. Wanneer er niet genoeg verlichting is, kan dit problemen geven met het scannen van de code.

### Combineer nooit tegels uit verschillende sets

Elke tegel in een set heeft zijn unieke ID -nummer en grafische afbeelding die herkend wordt door uw mobiele apparaat. Als u tegels uit verschillende sets combineert in een code, dan kunnen ze niet goed herkend worden. Wij adviseren u om de achterkant van de tegels van elke set, met een andere kleur te markeren, zodat u kunt zien welke tegels bij elkaar horen.

**COMBINEER NOOIT TEGELS VAN DE LABYRINTH VERSIE MET TEGELS VAN DE EDUCATIEVE VERSIE.**

### Voor een optimaal resultaat adviseren wij om één set te gebruiken voor 2 of 3 kinderen

Zorg, voordat u begint met het gebruik van Scottie Go!, dat uw leerlingen de spelregels kennen. U kunt bijvoorbeeld een "contract" maken met de regels. Een van de regels kan dan zijn, dat de leerlingen geen tegels van andere sets lenen. Wanneer de les is afgelopen, zorg er dan voor dat de leerlingen de tegels terugdoen in de doos waar ze ze uit gehaald hebben, zodat ze niet door elkaar raken. Indien u met jongere kinderen werkt, adviseren wij u om de voor die les benodigde tegels van tevoren klaar te leggen.

### Creëer een speciale docenten account, waarmee u toegang heeft tot alle opdrachten

U kunt een docenten account aanmaken, zodat u toegang heeft tot alle opdrachten in alle modules. Zo heeft u toegang tot elke gewenste opdracht, ook al heeft u niet alle voorgaande opdrachten uitgevoerd. Met de opties die u via het docenten account beschikbaar heeft, kunt u kiezen welke programmeerconcepten van Scottie Go! u wilt gebruiken, voor welke groep leerlingen. Zo kunt u rekening houden met de leeftijd en capaciteiten van uw leerlingen.

Voor het creëren van het docenten account, voert u de volgende stappen uit in de Scottie Go! Edu app: **Kies speler -> nieuwe speler** en voer de volgende code in: **Teacher\_13\_cps#**

**GEEF DEZE CODE NIET AAN UW LEERLINGEN!**

Via het docenten account kunt u uw leerlingen in groepen laten werken. U heeft dan maar één mobiel apparaat nodig voor het laten zien van de opdrachten en voor het scannen van de code die uw leerlingen gecreëerd hebben.

Verder kunt u met het docenten account optimaal gebruik maken van Scottie Go! Voor het downloaden van extra materialen, kunt u inloggen op uw account op:

[www.scottiego.nl/downloads/](http://www.scottiego.nl/downloads/)

# Opmerkingen voor docenten

## Er is meer dan één goede oplossing

In deze handleiding vindt u oplossingen voor elke opdracht. Onthoud echter, dat er talrijke manieren zijn om met verschillende codes hetzelfde probleem op te lossen. Sommigen zullen misschien zelfs korter zijn dan de voorgestelde oplossing.

## Het spel bevat 179 tegels die in elke module en opdracht gebruikt kunnen worden

Elke opdracht kunt u steeds opnieuw oplossen, gebruik makend van de nieuwe programmeerconcepten en de nieuwe tegels die geïntroduceerd worden in de volgende modules en opdrachten.

## De Scottie Go! Edu app gebruikt gamificatie-technieken, maar is een volwaardige educatieve tool

Wanneer u de Scottie GO! Edu app gebruikt, zult u uw curriculum doelen halen op het gebied van programmeervaardigheden. U kunt gebruik maken van de nieuwste ontwikkelingen met betrekking tot het logisch benaderen van problemen in andere contexten en die oplossen met computertechnologie:

- het begrijpen, analyseren en oplossen van problemen gebaseerd op logisch en abstract denken en het visueel presenteren van gegevens;
- programmeerconcepten gebruiken en problemen oplossen met gebruik van computers en andere digitale apparaten:  
het samenstellen en programmeren van algoritmen, organiseren, zoeken naar en delen van informatie, het gebruiken van computerprogramma's;
- sociale competenties ontwikkelen zoals:  
communiceren en samenwerken in groepen, gebruikmaken van virtuele omgevingen, deelnemen in groepsprojecten en projectmanagement.

**Onthoud – ons spel is slechts een tool die uw studenten helpt bij het leren, het is geen doel op zich!**

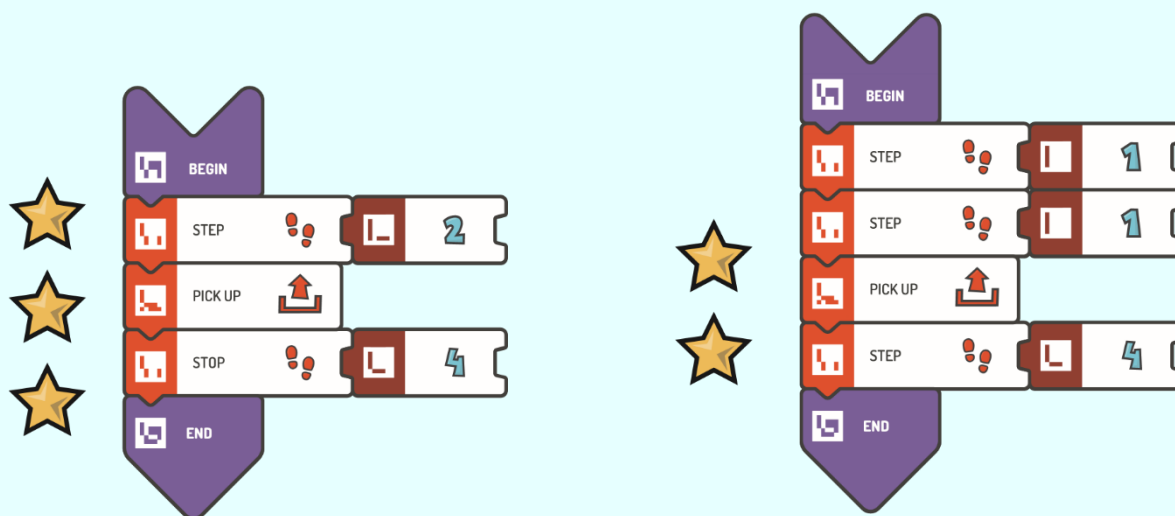
## Wanneer u Scottie Go! EDU in de klas gebruikt adviseren wij u om lesdoelen, lesmanagement, de regels voor actieve methodes en evaluaties te gebruiken.

Scottie Go! EDU volgt de spiraalbenadering met betrekking tot de nieuw geïntroduceerde ideeën. In een opdracht wordt bijvoorbeeld een commando geïntroduceerd dat ook, in een andere context, gebruikt wordt in de daarop volgende opdrachten. De moeilijkheidsgraad van de opdrachten, stijgt met elke opdracht, naarmate het spel vordert. Hierdoor kunnen de docent en zijn/haar leerlingen hun rekenkundige vaardigheden ontwikkelen. Elke keer als er een nieuw commando geïntroduceerd wordt, wordt dit aangegeven met een uitroepteken bij het spelkarakter in de rechter bovenhoek van het scherm.

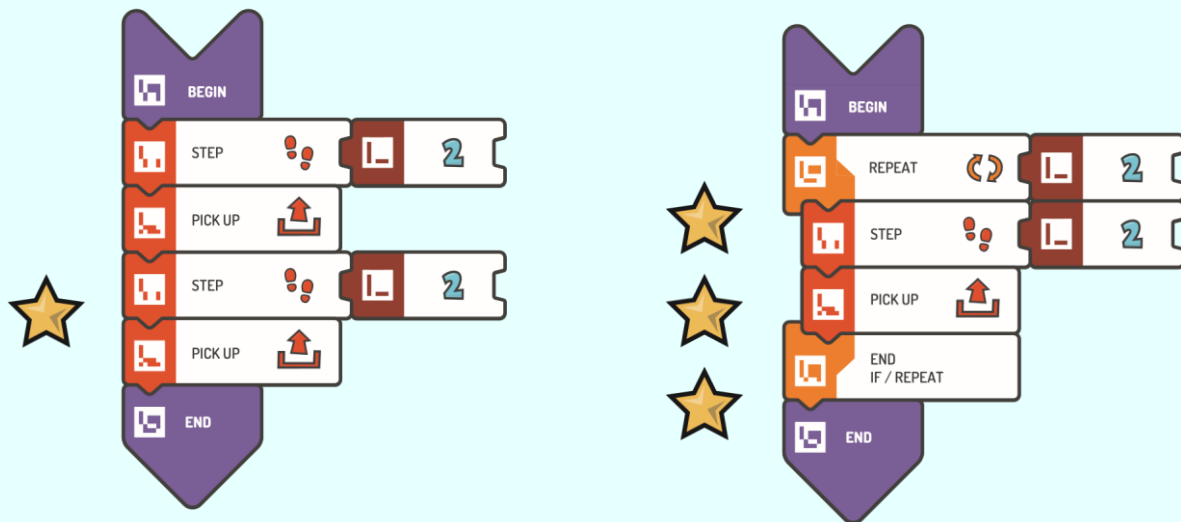
## Een goede codeur en een ambitieuze leerling zouden altijd naar de optimale oplossing voor een probleem moeten zoeken. De code is "optimaal" wanneer er zo min mogelijk regels code zijn gebruikt en het doel toch wordt bereikt.

Docenten dienen tijdens hun lessen een atmosfeer te creëren die het leren stimuleert. De ScottieGo! EDU app bevat een aantal feedback elementen in een aantrekkelijke, op een game gebaseerde, omgeving. Na het voltooien van elke opdracht, krijgen de leerlingen 1 tot 3 sterren als beloning. Hoe beter (korter) de code, hoe meer sterren ze krijgen. Dit zorgt voor veel speelplezier, zonder ze tegen elkaar te laten concurreren.

Ze krijgen drie sterren als de code het meest optimaal is, twee sterren als de code goed is, maar minder efficiënt. Eén ster betekent dat de opdracht is voltooid, maar dat de code te lang was.



Voor sommige opdrachten, zullen de leerlingen specifieke tegels moeten gebruiken als ze meer dan één ster willen verdienen. In onderstaand voorbeeld moeten ze bijvoorbeeld de REPEAT tegel gebruiken in hun oplossing.



Vanaf dit punt schakelen wij over van de u-vorm naar de je-vorm. Hier is voor gekozen, omdat de uitleg van de modules dan eenvoudig gebruikt kan worden in de les.

# Module I

## Europa 1



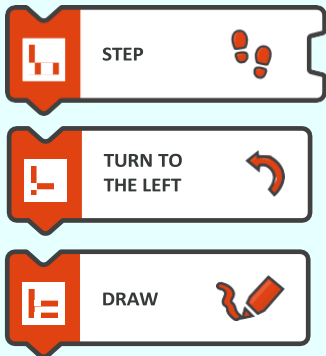
Ik houd van een speurtocht en ben benieuwd hoe goed jij het zult doen. Je moet langs alle velden met een "X".

# Module introductie: commando's

In Scottie GO! maken we een onderscheid tussen twee soorten commando's: simpel en complex.

## Simpele commando's

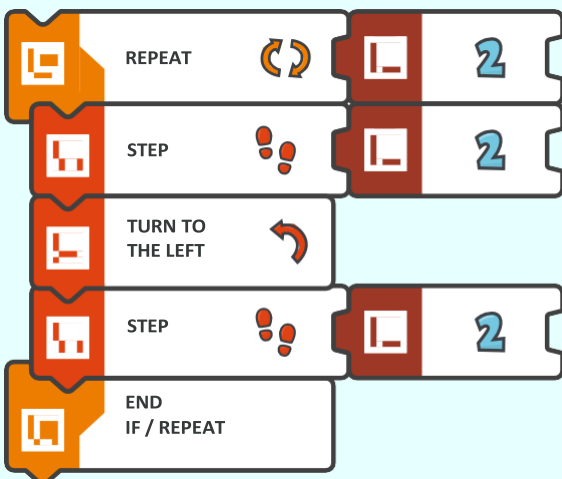
Simpele commando's vormen de basis van elk programma. Elk simpel commando bevat een enkele activiteit of opdracht die uitgevoerd moet worden. Voorbeelden van simpele commando's zijn bijvoorbeeld de opdrachten voor het gedrag en bewegen van Scottie:



Doorgaans bevat één regel code één commando en net als in Scottie Go! worden de commando's in regels onder elkaar geplaatst. Deze regels worden exact in deze volgorde uitgevoerd.

## Complexe commando's

Complexe commando's bepalen hoe de verschillende delen, van het volledige programma dat we ontwikkelen, worden geïnterpreteerd. Ze vormen meestal een reeks van een aantal eenvoudige opdrachten of andere complexe opdrachten. Normaal gesproken worden complexe commando's tussen haakjes geplaatst. In het geval van Scottie Go! worden ze tussen twee bij elkaar horende oranje blokken geplaatst. Voorbeelden van complexe commando's zijn de **REPEAT** lus, **REPEAT WHILE** lus of het voorwaardelijke commando **IF**.



# LET OP!

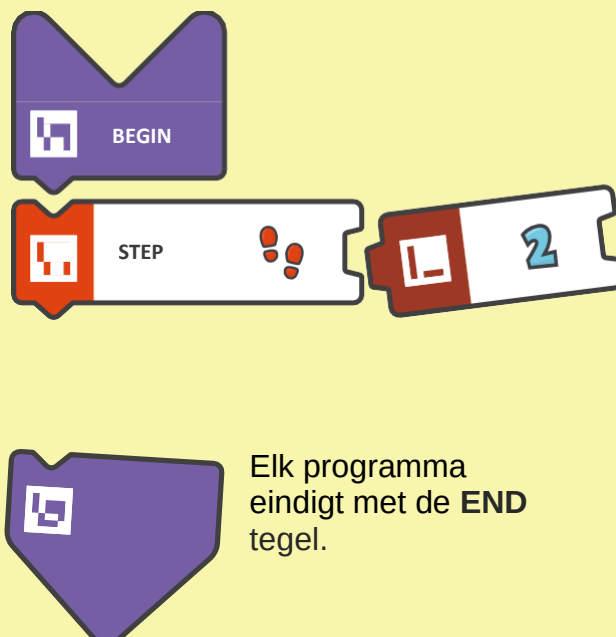
## Het spelkarakter en de eerste commando's

Het belangrijkste van de eerste opdracht is om de leerlingen om te leren gaan met de uitleg van de **Scottie Go! EDU** app. Leerlingen moeten leren hoe ze de programma's moeten scannen, zoals beschreven op pagina 14 van de Scottie Go! gebruikershandleiding.



Alhoewel de leerlingen meestal geen problemen hebben met het gebruik van de uitleg, zijn ze soms geneigd om de uitleg gewoon te volgen, zonder echt te letten op de betekenis van de tips en wat ze daar nog meer mee kunnen dan alleen wat ze zien in het voorbeeld. Het is belangrijk om op te merken, dat opdrachten **1-3 in Module I** het begrip van een **parameter** behandelen, wat een sleutelbegrip is door het hele spel. Een parameter is de waarde die bepaalt, hoe een commando uitgevoerd moet worden. Bij het eerste commando dat we leren in het spel - **STEP** - moeten de leerlingen de parameter aangeven voor het aantal stappen dat Scottie moet zetten.

Elk programma begint met de **BEGIN** tegel. Tussen de **BEGIN** en **END** tegels kunnen we elk gewenst aantal commando's leggen, die op volgorde uitgevoerd zullen worden.



Sommige commando's, zoals **STEP**, vereisen een parameter.

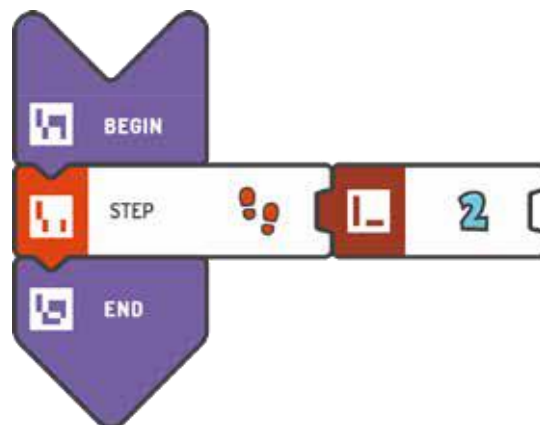
**STEP (2)** betekent dat Scottie 2 velden vooruit gaat;

**STEP (3)** betekent dat Scottie 3 velden vooruit gaat etc.

Opdracht:



Voorbeeldoplossing:



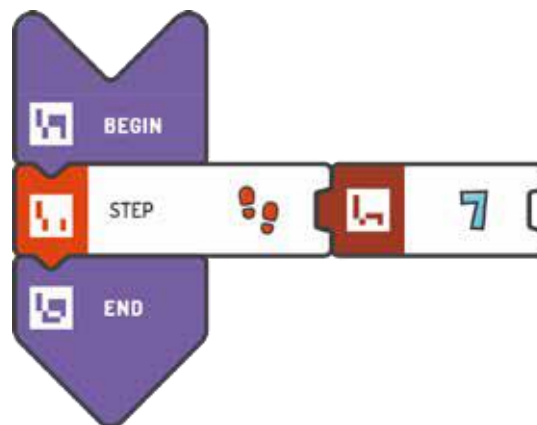
Voorbeeldoplossing (in tekstformaat):

**BEGIN**  
**STEP (2)**  
**END**

Opdracht:



Voorbeeldoplossing:

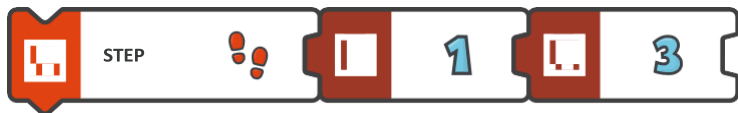


Voorbeeldoplossing (in tekstformaat):

**BEGIN**  
    **STEP (7)**  
**END**

# Het gebruik van cijfers, grotere getallen maken, optellen en aftrekken

Je kunt getallen creëren door tegels met cijfers te combineren.



Je kunt getallen ook creëren door ze op te tellen met de **PLUS** tegel,

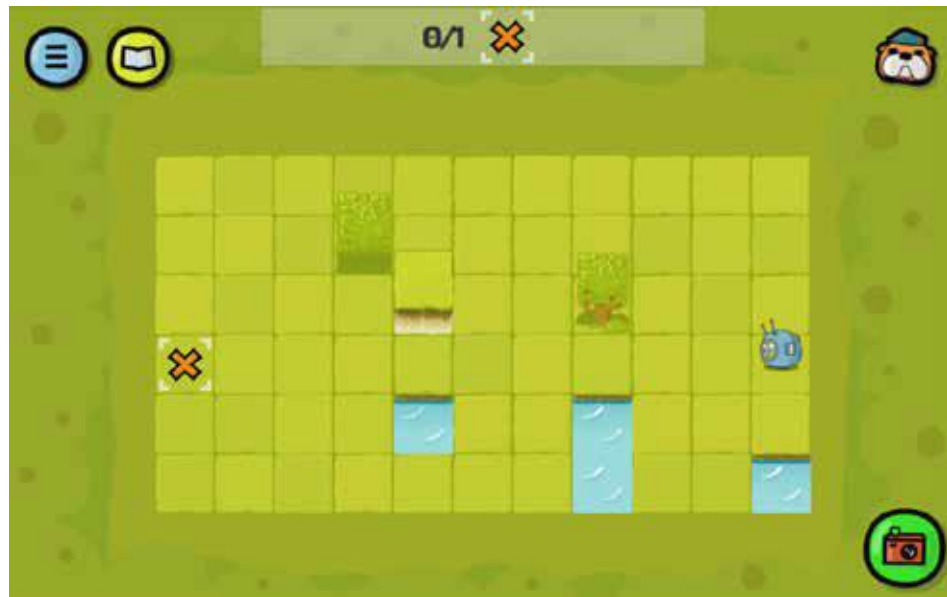


of door ze af te trekken met de **MIN** tegel.

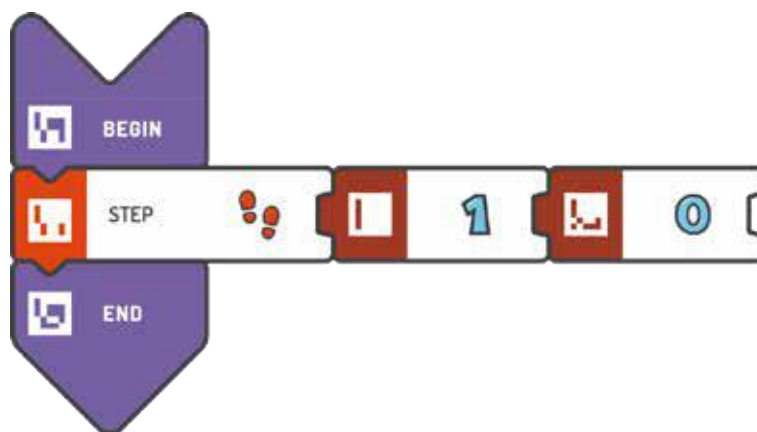


Wiskundige berekeningen mogen meerdere tegels met getallen/cijfers en zowel optellen als aftrekken bevatten.

Opdracht:



Voorbeeldoplossing:



Voorbeeldoplossing (in tekstformaat):

```
BEGIN
    STEP (10)
END
```

# Een draai maken

Scottie kan zich omdraaien. Gebruik hiervoor de **TURN TO THE LEFT** tegel,



of de **TURN TO THE RIGHT** tegel.

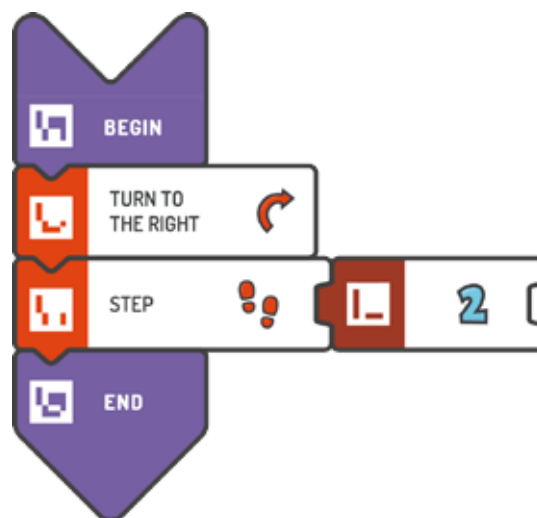


Om duidelijk te maken hoe Scottie beweegt, kun je in deze opdracht de leerlingen laten doen alsof ze Scottie zijn. Dit zal het voor hun makkelijker maken om te begrijpen hoe Scottie beweegt.

Opdracht:



Voorbeeldoplossing:



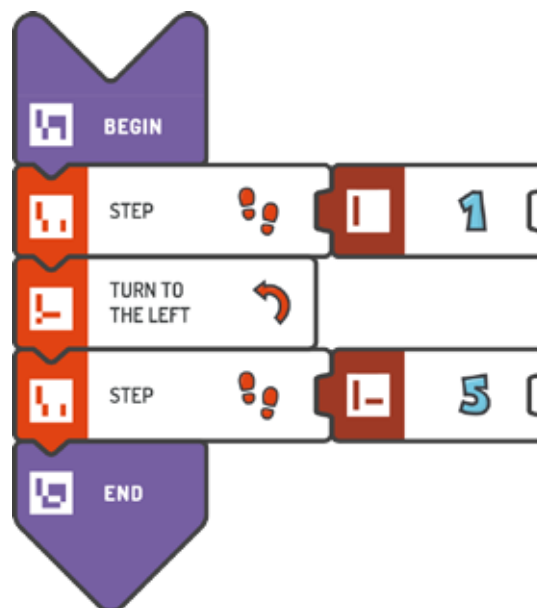
Voorbeeldoplossing (in tekstformaat):

```
BEGIN
  TURN TO THE RIGHT
  STEP (2)
END
```

Opdracht (onthoud dat Scottie langs alle velden met een "X" moet gaan):



Voorbeeldoplossing:



Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (1)**

**TURN TO THE LEFT**

**STEP (5)**

**END**

Opdracht (onthoud dat Scottie langs alle velden met een "X" moet gaan):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (2)**

**TURN TO THE LEFT**

**STEP (2)**

**TURN TO THE LEFT**

**STEP (2)**

**END**

Opdracht (onthoud dat Scottie langs alle velden met een "X" moet gaan):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (4)**

**TURN TO THE RIGHT**

**STEP (3)**

**TURN TO THE RIGHT**

**STEP (4)**

**END**

# Achteruit lopen

Scottie kan ook achteruit lopen. Gebruik hiervoor de **MIN** tegel.



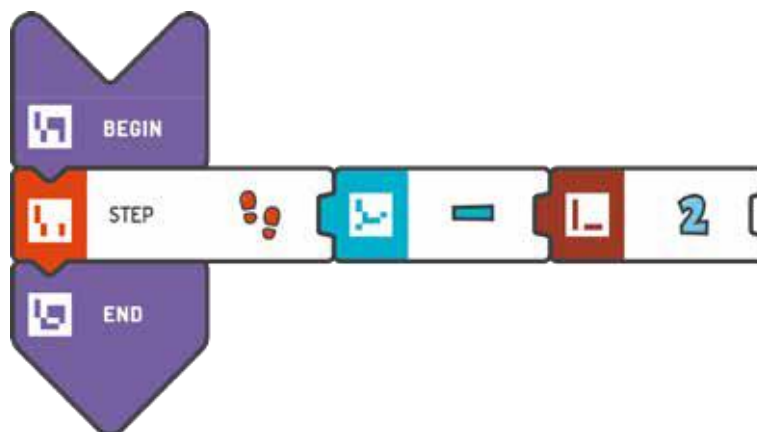
Achteruit lopen is een van Scottie's nieuwe mogelijkheden. Om het in het spel uit te laten voeren, gebruik je de **MIN** tegel vóór de tegel met het aantal stappen dat Scottie moet zetten.

Leerlingen die nog geen negatieve getallen hebben geleerd met de rekenles, zullen de **MIN** tegel toch snel kunnen gebruiken, wanneer ze de gelijkenis zien met de positieve en negatieve temperatuur die ze als vanzelfsprekend beschouwen en zonder problemen kunnen begrijpen.

Opdracht:



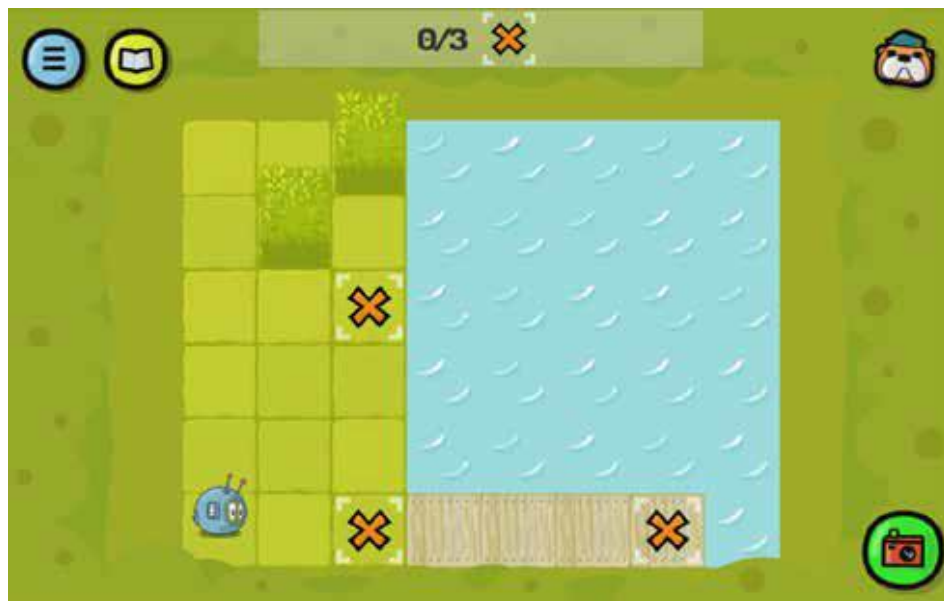
Voorbeeldoplossing:



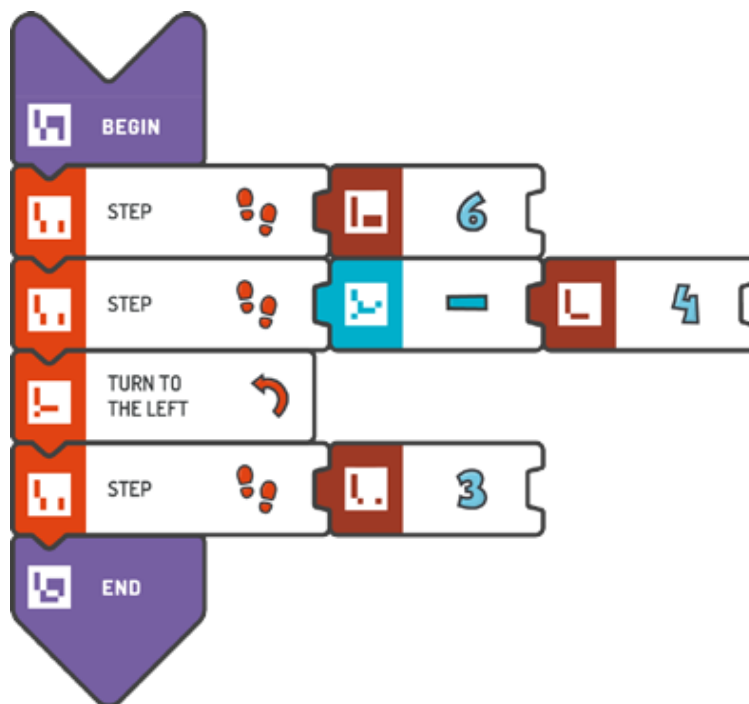
Voorbeeldoplossing (in tekstformaat):

```
BEGIN
  STEP (-2)
END
```

Opdracht (onthoud dat Scottie langs alle velden met een "X" moet gaan):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (6)**

**STEP (-4)**

**TURN TO THE LEFT**

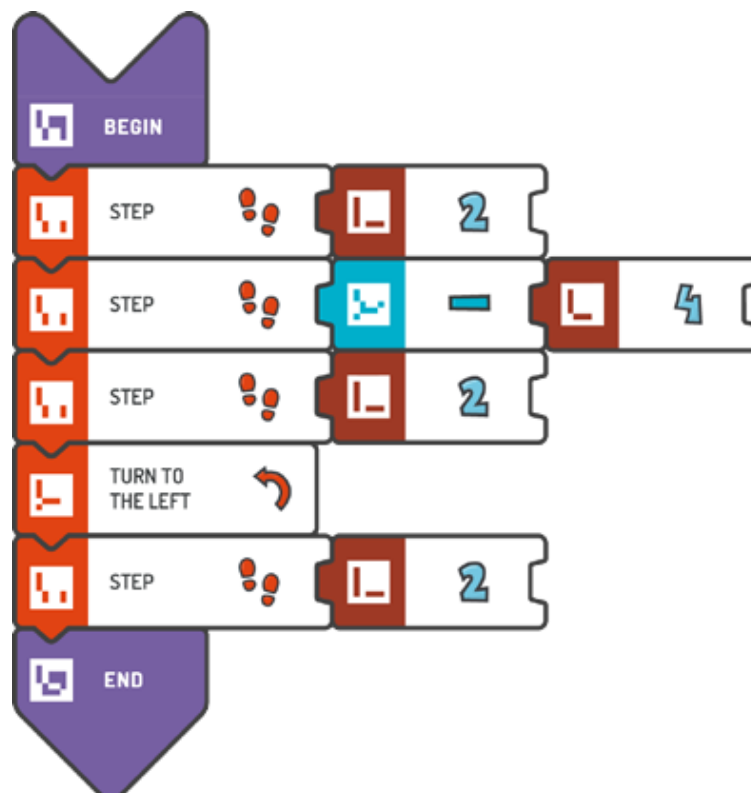
**STEP (3)**

**END**

Opdracht (onthoud dat Scottie langs alle velden met een "X" moet gaan):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (2)**

**STEP (-4)**

**STEP (2)**

**TURN TO THE LEFT**

**STEP (2)**

**END**

# Module II

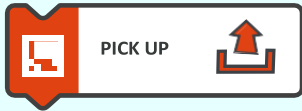
## Europa 2



**Scottie begint interactie te krijgen met zijn omgeving.  
Raap alle slakken van de grond op en neem ze mee  
naar de atletiekbaan.**

# Module introductie: objecten oprapen van het spelbord

Om objecten van het spelbord op te rapen, gebruik je het **PICK UP** commando.



Gebruik dit commando in programma's, bijvoorbeeld als je slakken wilt oprapen.

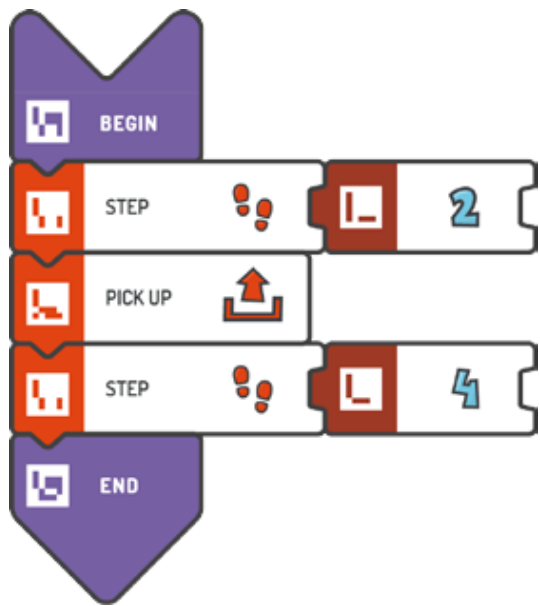
## Let op!

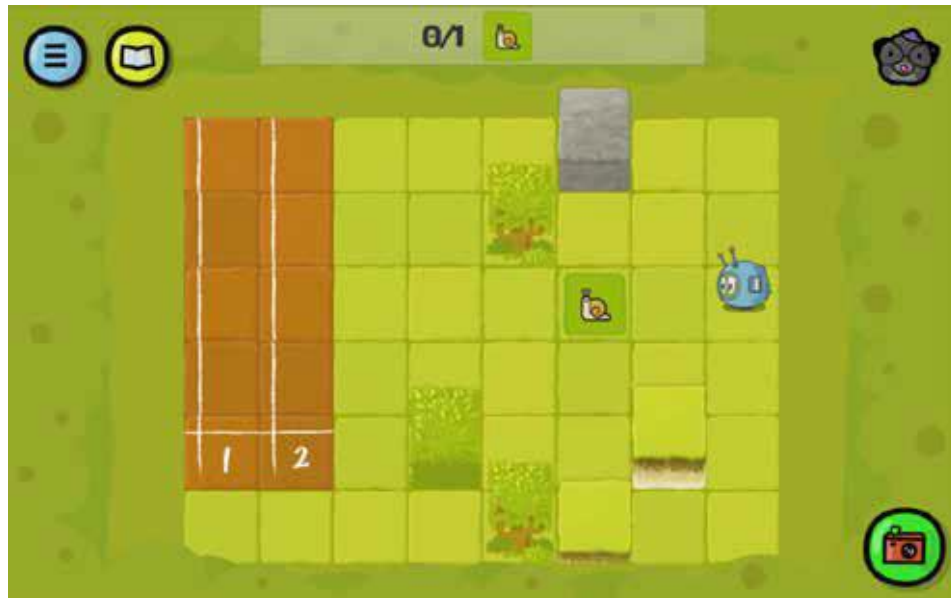
Het gebruik van het **PICK UP** commando op een leeg veld, wordt gezien als een fout. In Module II hoeft de **PLACE** tegel nog niet gebruikt te worden.

Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (2)**

**PICK UP**

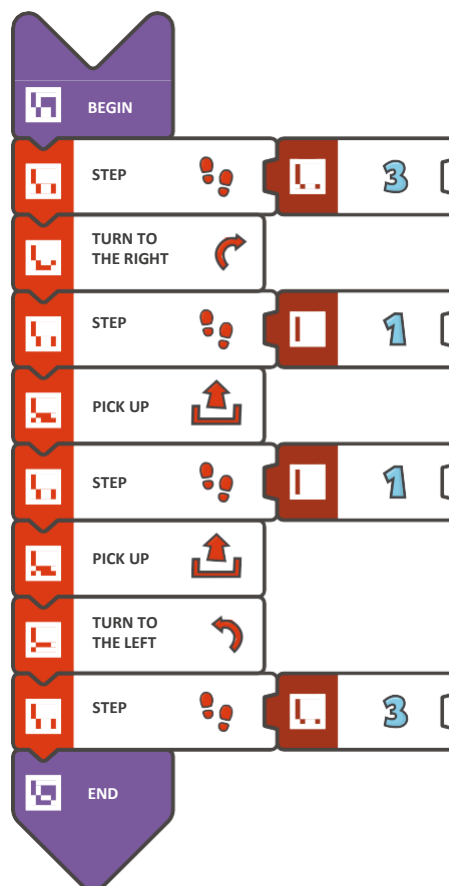
**STEP (4)**

**END**

Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (3)**

**TURN TO THE RIGHT**

**STEP (1)**

**PICK UP**

**STEP (1)**

**PICK UP**

**TURN TO THE LEFT**

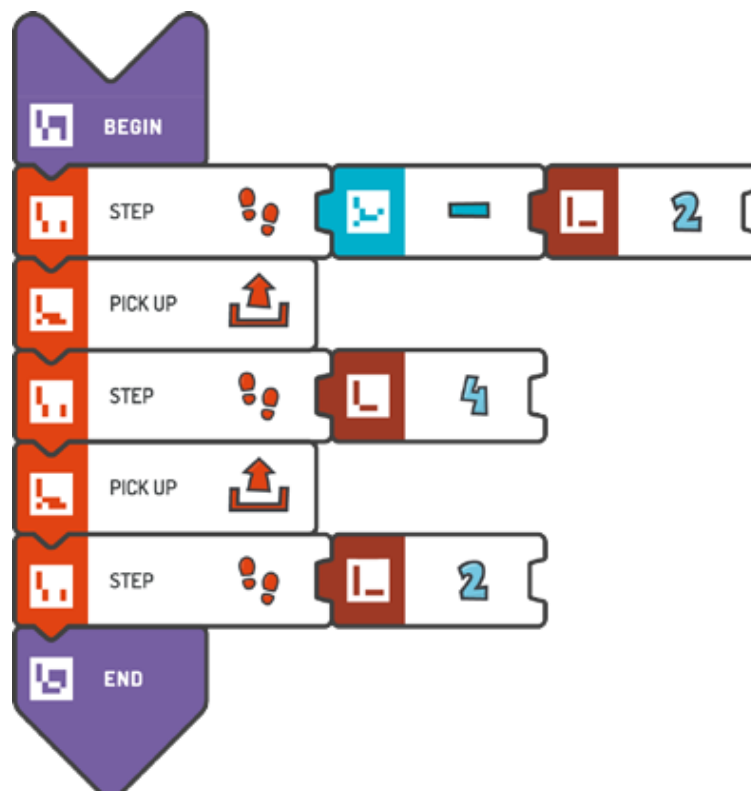
**STEP (3)**

**END**

Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (-2)**

**PICK UP**

**STEP (4)**

**PICK UP**

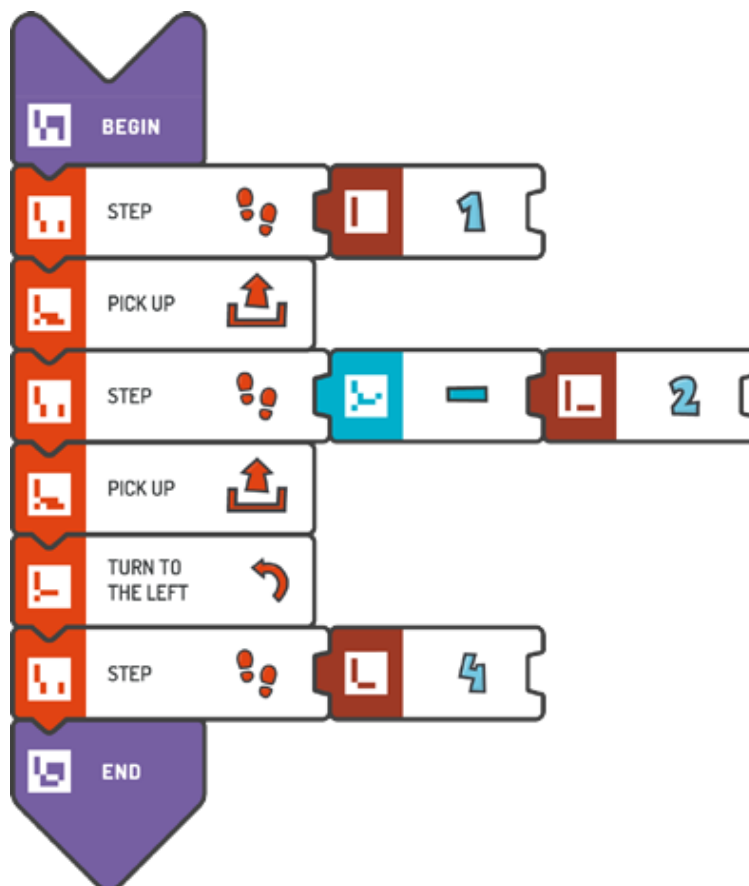
**STEP (2)**

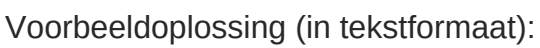
**END**

Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





## STEP (1)

## STEP (-2)

## TURN TO THE LEFT

## STEP (4)

**END**

# Springen

Soms komt Scottie klein obstakels tegen op zijn weg. Om hier overheen te springen, gebruik je de rode **JUMP** tegel. Scottie kan alleen vooruit springen.

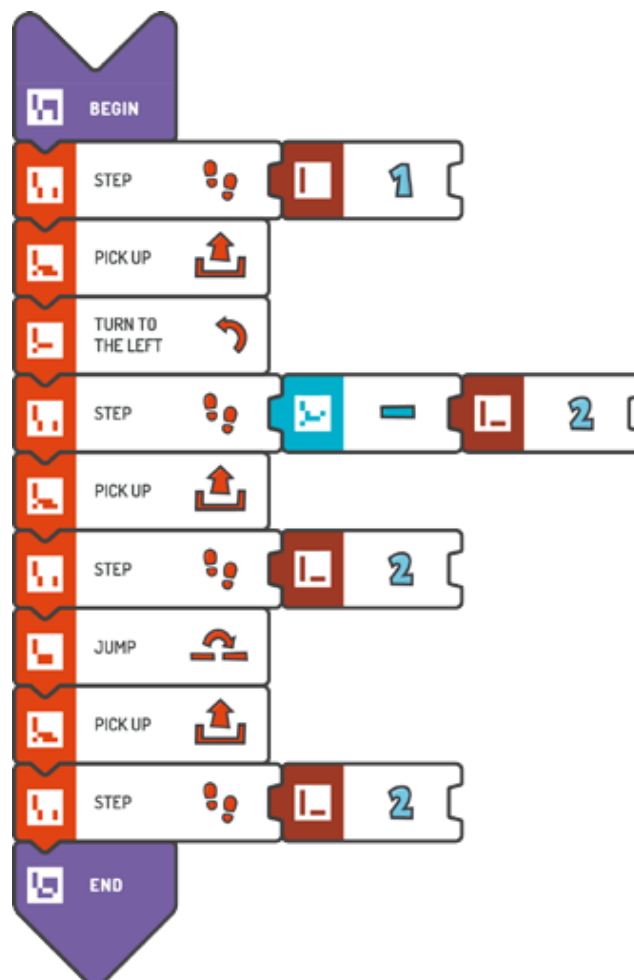


Als Scottie springt, dan springt hij van het veld waar hij op staat, over veld 2, naar veld 3.

Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (1)**

**PICK UP**

**TURN TO THE LEFT**

**STEP (-2)**

**PICK UP**

**STEP (2)**

**JUMP**

**PICK UP**

**STEP (2)**

**END**

Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (1)**

**TURN TO THE LEFT**

**STEP (1)**

**PICK UP**

**JUMP**

**PICK UP**

**JUMP**

**PICK UP**

**STEP (2)**

**END**

# LET OP!

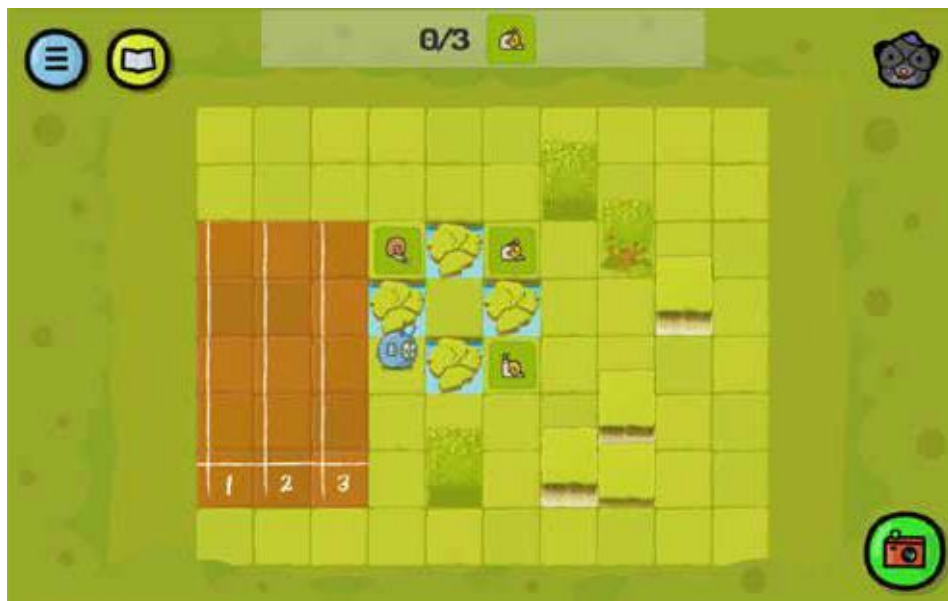
Leerlingen hoeven het 'loop' algoritme nog niet te gebruiken in deze opdracht, ook al zou dit de oplossing veel eenvoudiger maken.

Tijdens de lessen zullen leerlingen vaak intuïtief handelen en andere tegels gebruiken, zoals bijvoorbeeld de **REPEAT** tegel, ook al moet dit commando in een volgende module nog uitgelegd worden.



De **REPEAT** evenals de **END IF / REPEAT** tegels mogen wel al gebruikt worden in Module I. De juiste manier om een lus (loop) te gebruiken wordt echter in de introductie van Module III uitgelegd.

Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**JUMP**

**PICK UP**

**TURN TO THE LEFT**

**JUMP**

**PICK UP**

**TURN TO THE LEFT**

**JUMP**

**PICK UP**

**STEP (1)**

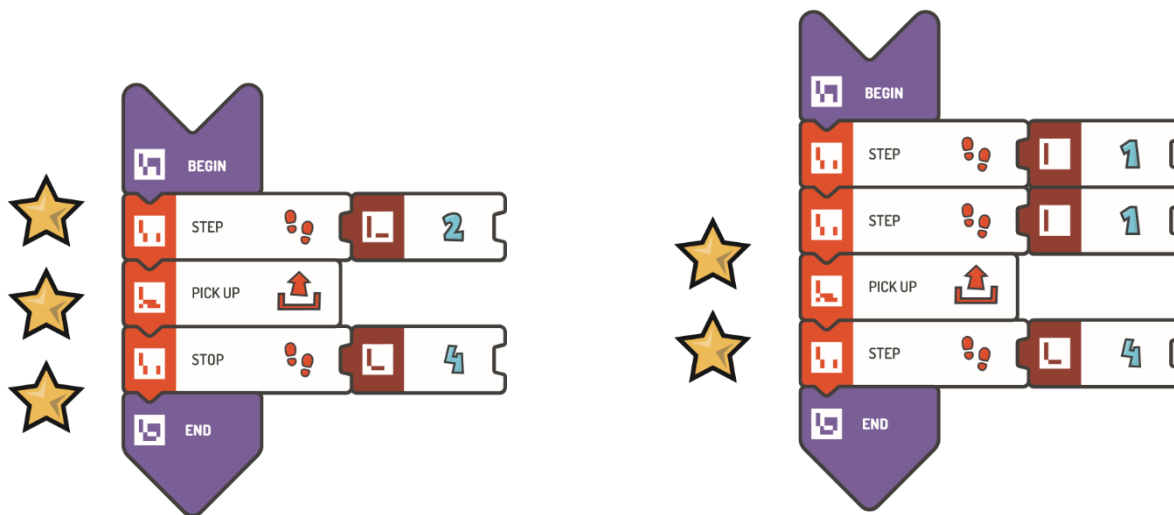
**END**

## LET OP!

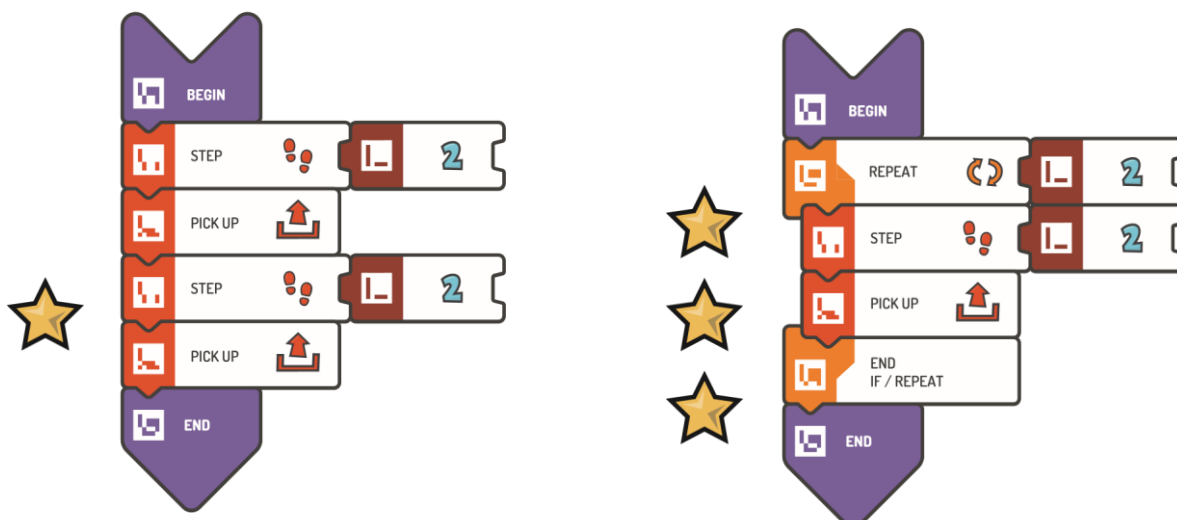
Met deze opdracht zullen leerlingen waarschijnlijk worstelen met het vinden van de optimale route voor Scottie, waarbij de code zo kort mogelijk is.

In Scottie Go! EDU krijgen de leerlingen sterren op basis van de kwaliteit van de oplossing.

Ze krijgen drie sterren als de code het meest optimaal is, twee sterren als de code goed is, maar minder efficiënt. Eén ster betekent dat de opdracht is voltooid, maar dat de code te lang was.



Voor sommige opdrachten, zullen de leerlingen specifieke tegels **moeten** gebruiken als ze meer dan één ster willen verdienen. In onderstaand voorbeeld moeten ze bijvoorbeeld de **REPEAT** tegel gebruiken in hun oplossing.



Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (1)**

**PICK UP**

**TURN TO THE LEFT**

**STEP (2)**

**TURN TO THE RIGHT**

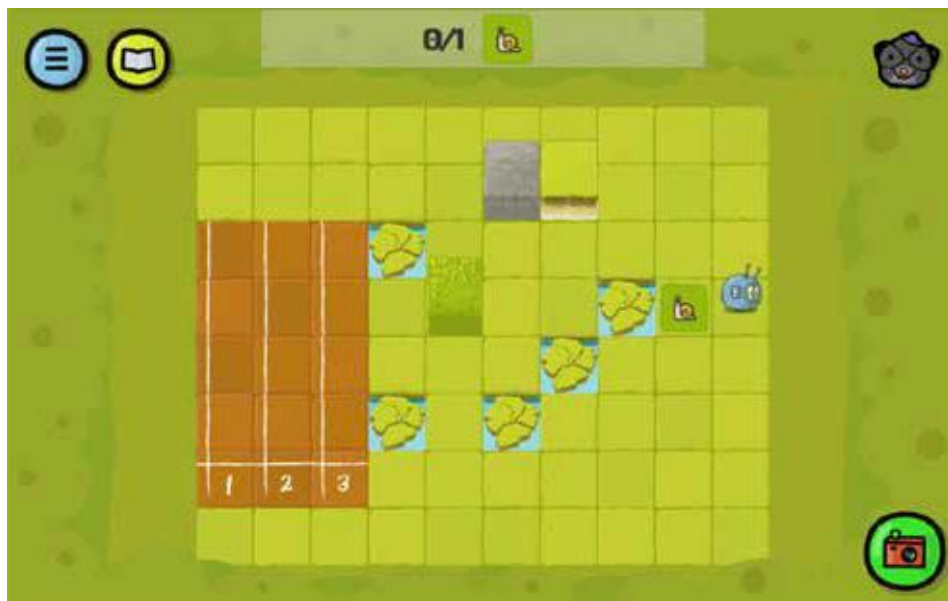
**STEP (6)**

**END**

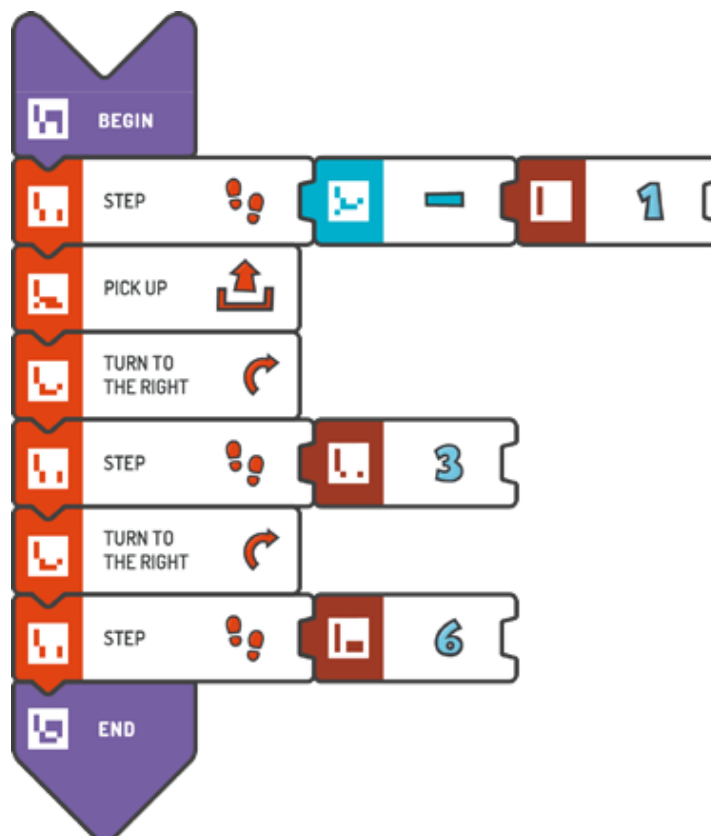
## TIP

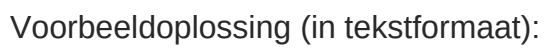
Met deze opdracht raden we aan om de leerlingen te stimuleren, om de optimale route voor Scottie te zoeken, zodat de code zo kort mogelijk is.

Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





**END**

Opdracht (onthoud dat Scottie op de atletiekbaan moet eindigen):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (2)**

**PICK UP**

**TURN TO THE LEFT**

**STEP (6)**

**TURN TO THE LEFT**

**STEP (1)**

**END**

# Module III

## Zuid-Amerika



**Terugkerende acties.**

**Pak alle veren op van het spelbord.**

# Module introductie: de REPEAT lus

Na het uitvoeren van een aantal van de opdrachten in Module I, zal het waarschijnlijk een aantal leerlingen opgevallen zijn, dat Scottie sommige acties meerdere keren moet uitvoeren. Soms moeten deze acties in exact dezelfde volgorde uitgevoerd worden. Het zou handig zijn als er een manier was om dit te automatiseren. Het zou de leerlingen ook leren om hun werk te optimaliseren door de meest efficiënte manier te zoeken. Wanneer je een eenvoudige manier zoekt om bepaalde acties vaak te herhalen, dan zijn de **lussen** het juiste stukje gereedschap voor jou en je leerlingen.

De **REPEAT lus** is een van de meest gebruikte **lussen**. In Scottie Go! wordt de eenvoudigste versie gebruikt. Wanneer je een **lus** wilt gebruiken, moet je aangeven welk deel van het programma herhaald moet worden,



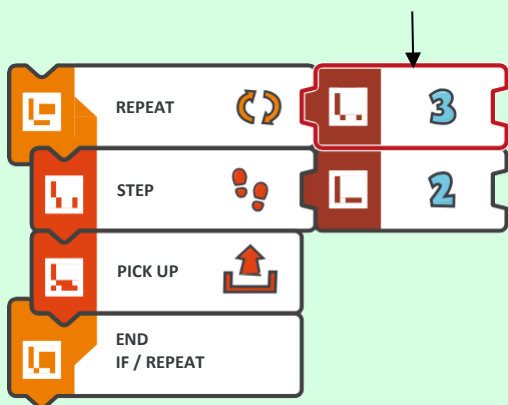
vervolgens moet aangegeven worden hoe vaak het herhaald moet worden. De code voor het herhalen van commando's bestaat uit twee tegels: **REPEAT** en **END IF / REPEAT**. Deze tegels staan gelijk aan de haakjes die gebruikt worden in populaire programmeertalen.



# De eerste lus

Een lus is een van de meest onmisbare en vaak gebruikte begrippen bij het programmeren. Het beheersen van lussen is ook een belangrijke rekenkundige competentie. Zonder de lussen, zouden talrijke digitale apparaten niet goed kunnen werken en de codes zouden te lang en daardoor minder effectief zijn. Een lus maakt het mogelijk om een reeks commando's te herhalen tot het gewenste doel behaald is.

In deze opdracht, moeten de leerlingen er aan denken om gebruik te maken van de **REPEAT** lus. Een lus moet altijd afgesloten worden met een **END IF / REPEAT** tegel. De met rood gemarkeerde tegel in onderstaande afbeelding, geeft het aantal herhalingen voor de reeks commando's in de lus aan:



De lus zoals aangegeven met de tegels hierboven, kan vertaald worden naar de volgende code in tekstformaat:

```
REPEAT (3) {  
    STEP (2)  
    PICK UP  
}
```

**REPEAT** (aantal herhalingen) {

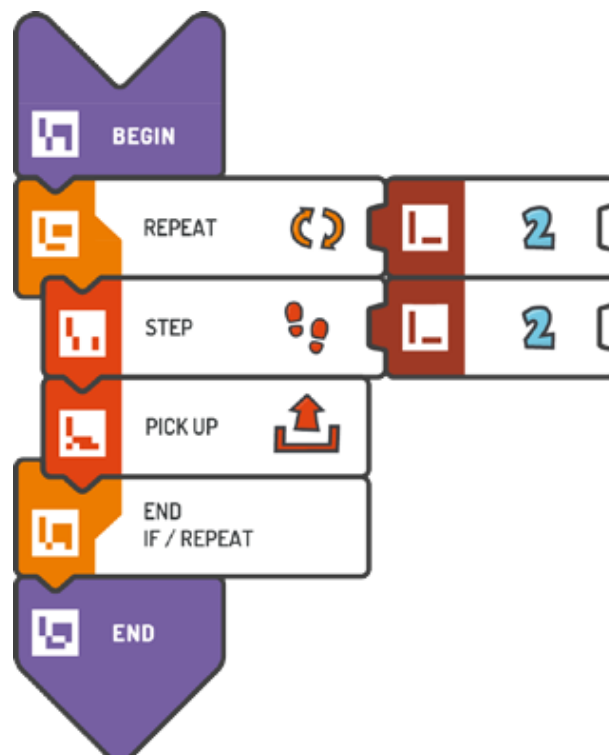
markeert de plaats waar de **REPEAT** tegel met het aantal herhalingen is gebruikt.

} markeert de plaats waar de **END IF / REPEAT** tegel is gebruikt.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (2) {**  
**STEP (2)**  
**PICK UP**

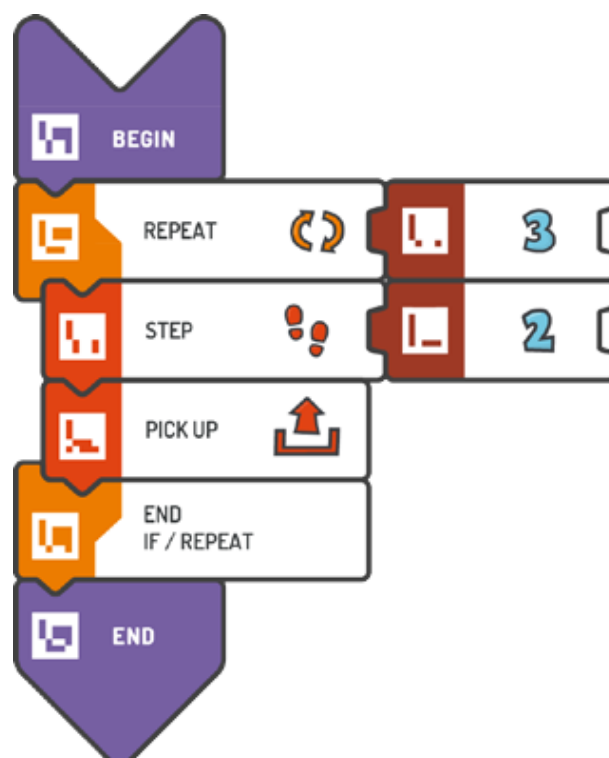
**}**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (3) {**

**STEP (2)**

**PICK UP**

**}**

**END**

## Binnen en buiten een lus werken

In sommige gevallen zullen bepaalde commando's in een code herhaald moeten worden, maar andere niet. Het is belangrijk om de leerlingen te vertellen dat ze de commando's die ze **niet** willen laten herhalen buiten de lus moeten plaatsen! In lussen is de volgorde van de gebeurtenissen heel erg belangrijk. Laat de leerlingen zich deze volgorde in hun hoofd voorstellen, voordat ze dit deel van de code gaan schrijven. De volgorde van de commando's zal Scottie's gedrag beïnvloeden.

In onderstaand voorbeeld, zal de lus twee keer herhaald moeten worden, als je de eerste veer buiten de lus oppakt.



Pak je de eerste veer binnen de **lus** op, dan moet de **lus** drie keer herhaald worden.



Beide lussen bevatten twee commando's. Bekijk welk van de twee oplossingen minder regels code heeft.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (3)**

**PICK UP**

**TURN TO THE RIGHT**

**REPEAT (3) {**

**STEP (2)**

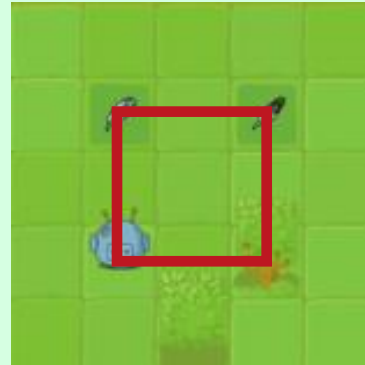
**PICK UP**

**}**

**END**

# Scottie GO! EDU als gereedschap om begrippen uit de wis- en natuurkunde te introduceren

In deze opdracht is de route die Scottie moet volgen, terwijl hij de veren oppakt, onderdeel van een vierkant. Hij beweegt zich langs de hoeken van dit vierkant, waar de veren zich in bevinden. Dit is een mooie kans om met de leerlingen te beginnen over rechte hoeken, evenwijdige en loodrechte lijnen.



Scottie Go! EDU wordt gespeeld op een rechthoekig raster wat een natuurlijk hulpmiddel is om de begrippen: vectoren, Cartesisch coördinatensysteem en resulterende krachten uit te leggen.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (2) {**

**STEP (2)**

**PICK UP**

**TURN TO THE RIGHT**

**}**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**TURN TO THE RIGHT**

**REPEAT (3) {**

**STEP (2)**

**PICK UP**

**TURN TO THE LEFT**

**}**

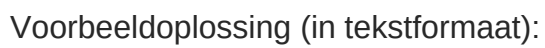
**END**

Opdracht:



Voorbeeldoplossing:



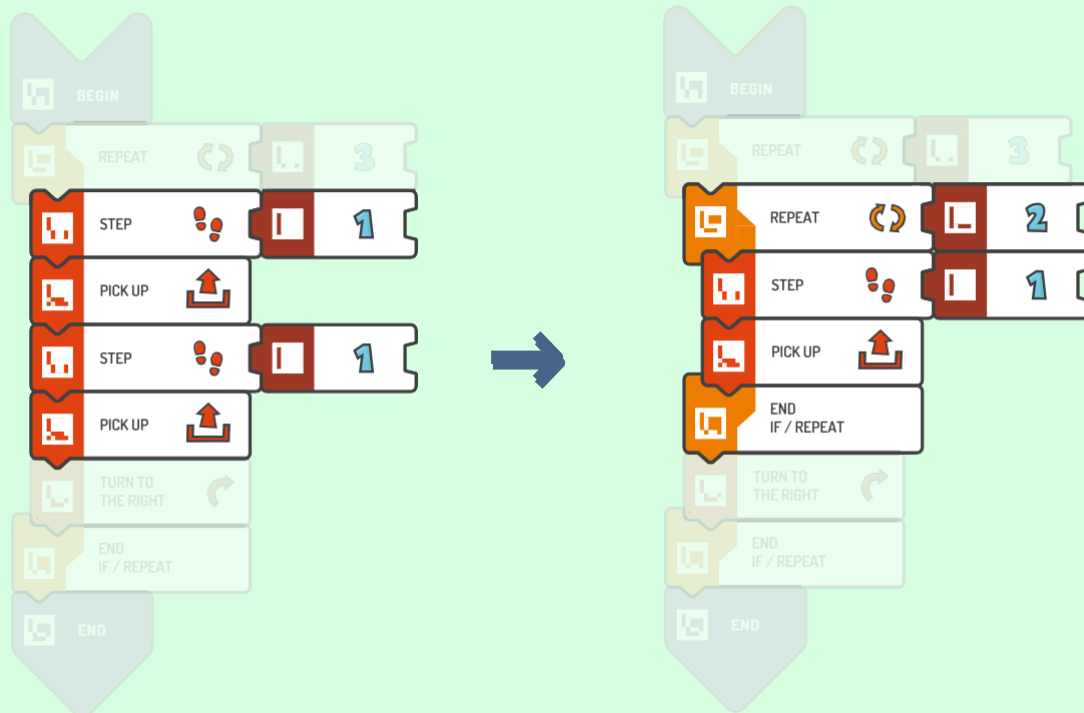


```
REPEAT (3) {  
    STEP (2)  
    PICK UP  
    TURN TO THE LEFT  
    STEP (2)  
    PICK UP  
    TURN TO THE RIGHT  
}
```

**END**

# Geneste lussen

Om het aantal regels code te beperken, kun je een andere, extra lus, in een bestaande **REPEAT** lus plaatsen. Dit noemen we geneste lussen. Deze manier van rekenkundig denken, leidt tot verdere optimalisatie van de code en beïnvloedt het niveau van de rekenkundige complexiteit. Wanneer leerlingen de geneste lussen beheersen, zullen ze niet alleen hun rekenkundige intuïtie ontwikkelen, maar ook de noodzaak van het steeds verbeteren van de code begrijpen.



Onderstaand de code in tekstformaat:

```
BEGIN
  REPEAT (3) {
    STEP (1)
    PICK UP
    STEP (1)
    PICK UP
    TURN TO THE RIGHT
  }
END
```

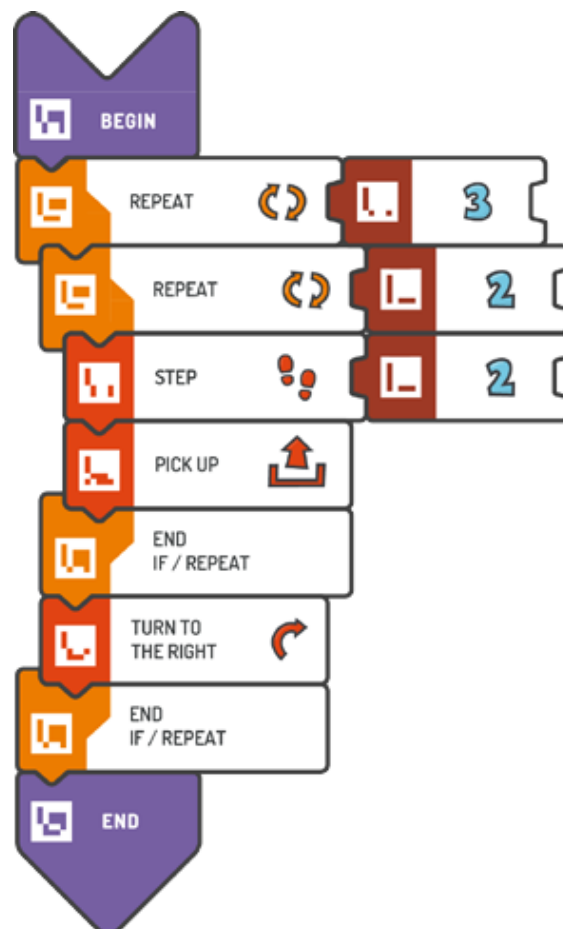


```
BEGIN
  REPEAT (3) {
    REPEAT (2) {
      STEP (1)
      PICK UP
    }
    TURN TO THE RIGHT
  }
END
```

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (3) {**

**REPEAT (2) {**

**STEP (2)**

**PICK UP**

**}**

**TURN TO THE RIGHT**

**}**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (2) {**

**REPEAT (3) {**

**STEP (2)**

**PICK UP**

**}**

**TURN TO THE LEFT**

**STEP (1)**

**TURN TO THE LEFT**

**}**

**END**

# Module IV

## Noord-Amerika



**Je kunt veel goud vinden in de nabijgelegen rotsen.  
Help Scottie om het goud te verzamelen met de hulp  
van zijn dierenvrienden – de Bevers.**

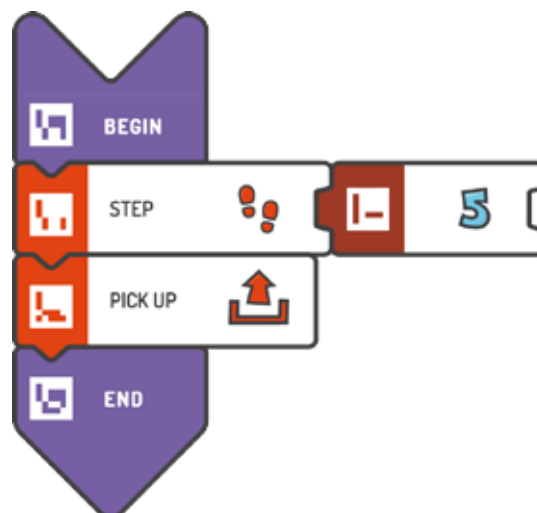
## Bever als een brug

Om de rivier over te steken, moet Scottie op de Bever zijn hoofd stappen. De leerlingen zullen de tegels die ze al eerder geleerd hebben, moeten gebruiken voor een nieuw resultaat.

Opdracht:



Voorbeeldoplossing:



Voorbeeldoplossing (in tekstformaat):

**BEGIN**  
**STEP (5)**  
**PICK UP**  
**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (5)**

**PICK UP**

**TURN TO THE RIGHT**

**STEP (1)**

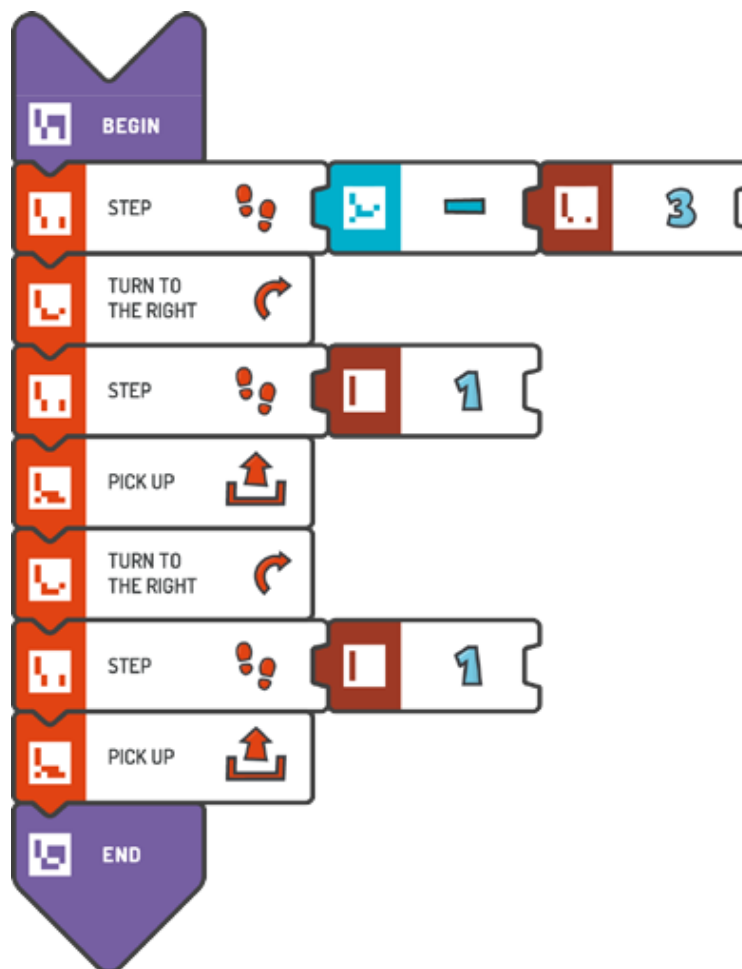
**PICK UP**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (-3)**

**TURN TO THE RIGHT**

**STEP (1)**

**PICK UP**

**TURN TO THE RIGHT**

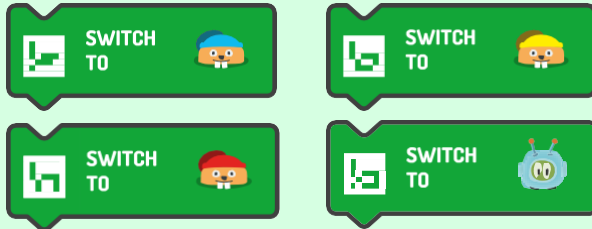
**STEP (1)**

**PICK UP**

**END**

# Bever in beweging

Je kunt Bever ook als brug of vlot gebruiken. Hoe Bever zwemt kun je programmeren door gebruik te maken van de nieuwe tegels:



## LET OP!

Elke Bever heeft een andere kleur pet (**geel, blauw of rood**), het is dus belangrijk om de juiste tegel uit de doos te pakken.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**SWITCH TO - BLUE BEAVER:**

**STEP (4)**

**SWITCH TO - SCOTTIE:**

**STEP (5)**

**PICK UP**

**STEP (1)**

**PICK UP**

**END**

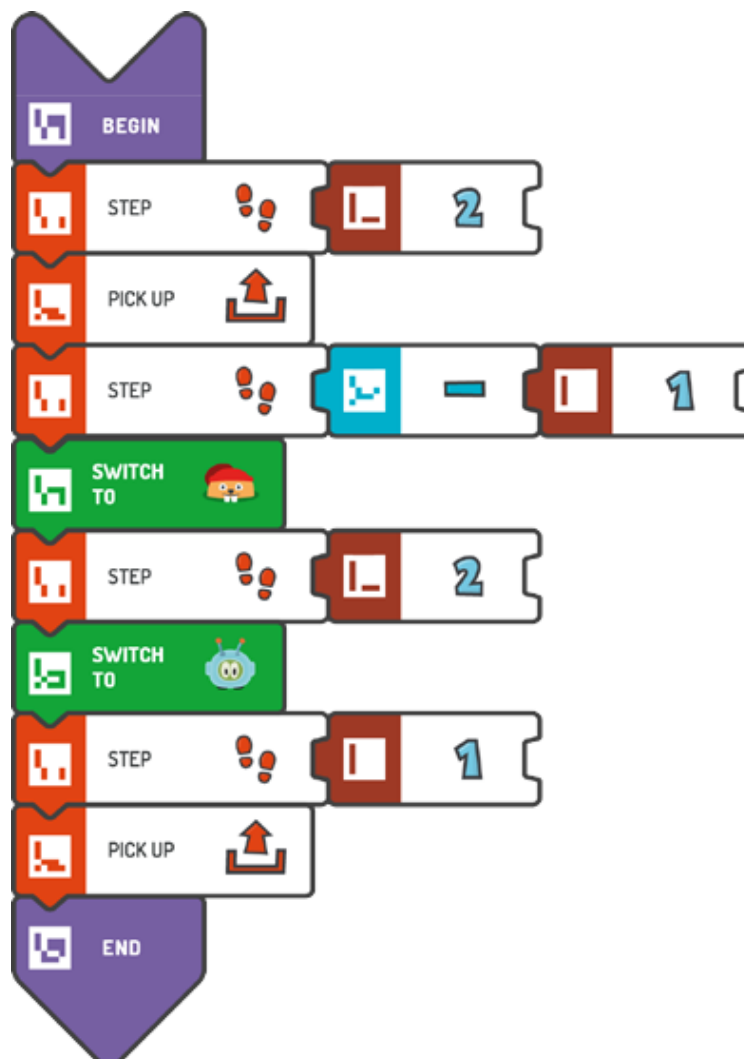
## TIP

Wanneer je wilt dat Scottie sneller op zijn bestemming aankomt, laat hem dan op een bever zwemmen. Dit is een mooie mogelijkheid om de meest optimale route voor Scottie te zoeken en daardoor de code zo kort mogelijk te maken.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (2)**

**PICK UP**

**STEP (-1)**

**SWITCH TO - RED BEAVER:**

**STEP (2)**

**SWITCH TO - SCOTTIE:**

**STEP (1)**

**PICK UP**

**END**

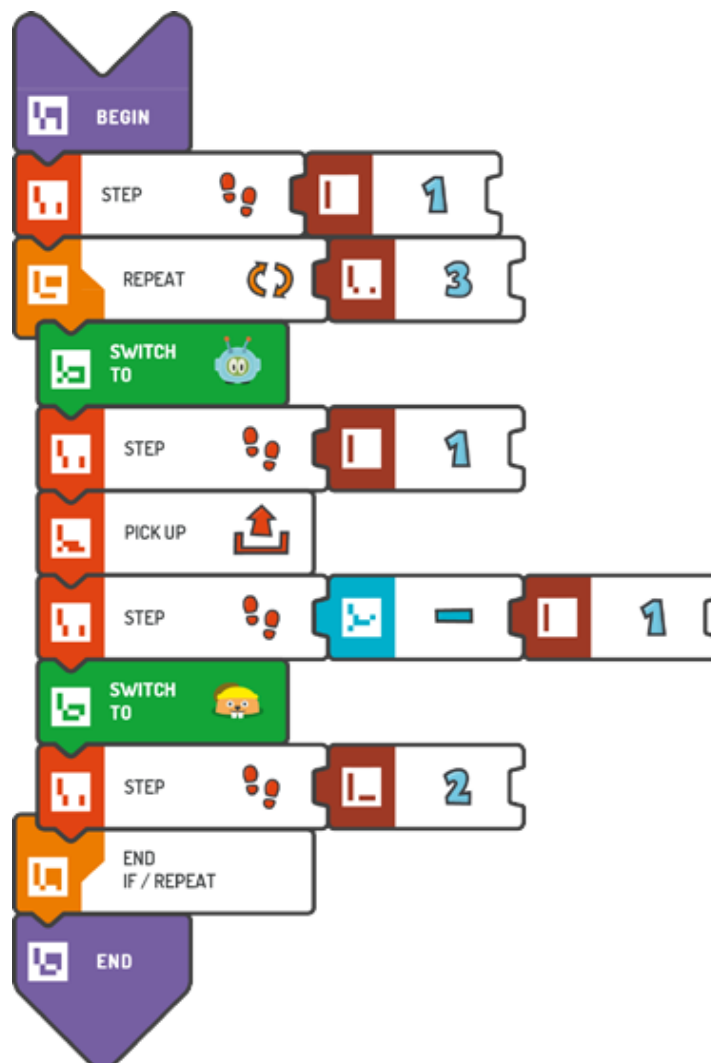
# Gebruik de REPEAT lus en de SWITCH TO tegels samen

In deze opdracht dienen de leerlingen de **REPEAT** lus én de **SWITCH TO** tegels samen te gebruiken om de spelkarakters om de beurt te laten bewegen. Het zal ze helpen om voort te bouwen op de reeds bekende begrippen en hun vaardigheden te ontwikkelen met betrekking tot het combineren van losse stukjes code.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (1)**

**REPEAT (3) {**

**SWITCH TO - SCOTTIE:**

**STEP (1)**

**PICK UP**

**STEP (-1)**

**SWITCH TO - YELLOW BEAVER:**

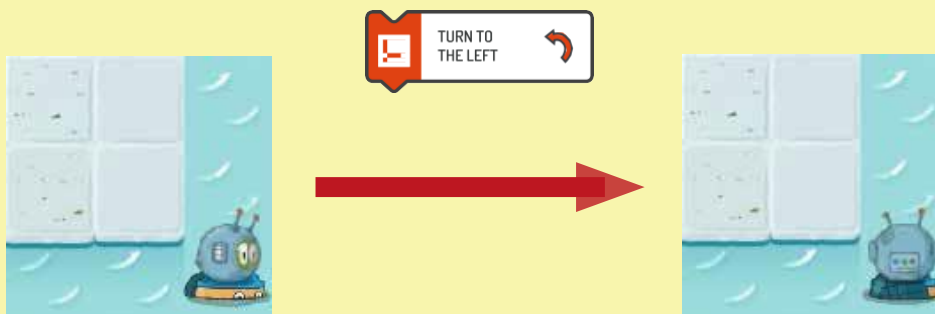
**STEP (2)**

**}**

**END**

# Bever maakt een draai

Bever kan zich, net als Scottie, omdraaien. Wanneer Scottie op Bever zijn hoofd zwemt op het moment dat Bever draait, dan zal Scottie met Bever meedraaien.

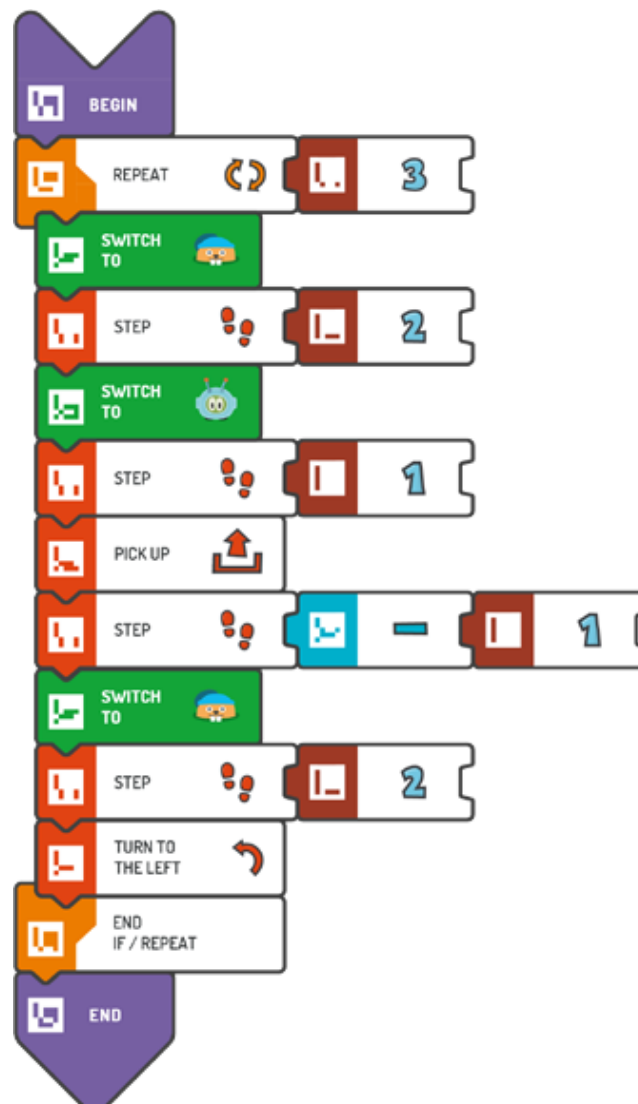


Je kunt nu Bever zijn hulp gebruiken. Voltooi de volgende opdrachten door gebruik te maken van de **REPEAT** lus en laat Bever draaien.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (3) {**

**SWITCH TO - BLUE BEAVER:**

**STEP (2)**

**SWITCH TO - SCOTTIE:**

**STEP (1)**

**PICK UP**

**STEP (-1)**

**SWITCH TO - BLUE BEAVER:**

**STEP (2)**

**TURN TO THE LEFT**

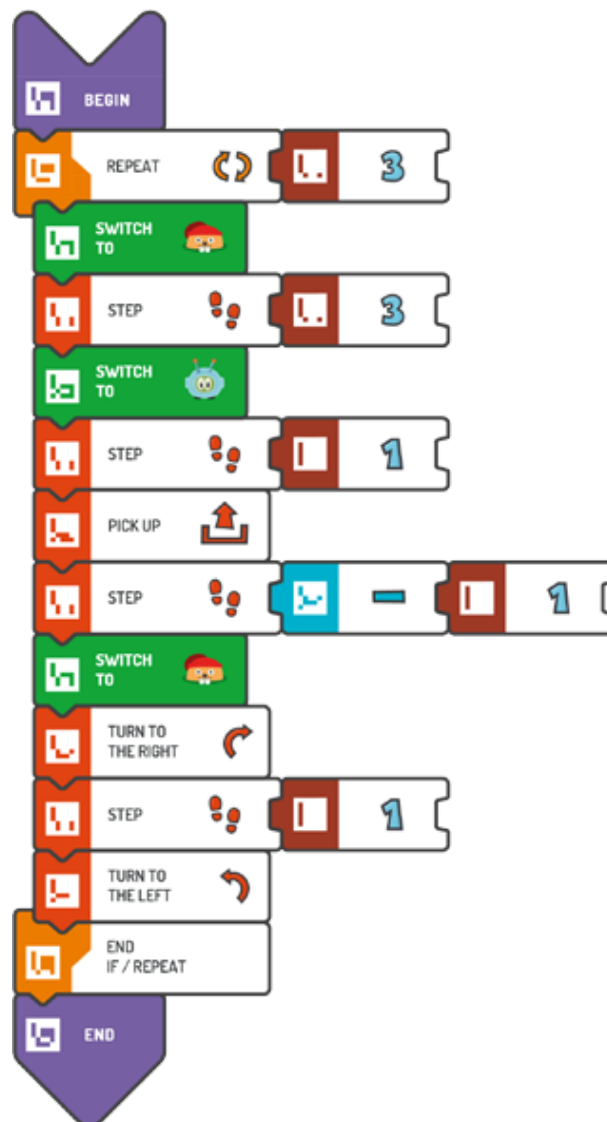
**}**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

```

REPEAT (3) {
  SWITCH TO - RED BEAVER:
  STEP (3)
  SWITCH TO - SCOTTIE:
  STEP (1)
  PICK UP
  STEP (-1)
  SWITCH TO - RED BEAVER:
  TURN TO THE RIGHT
  STEP (1)
  TURN TO THE LEFT

```

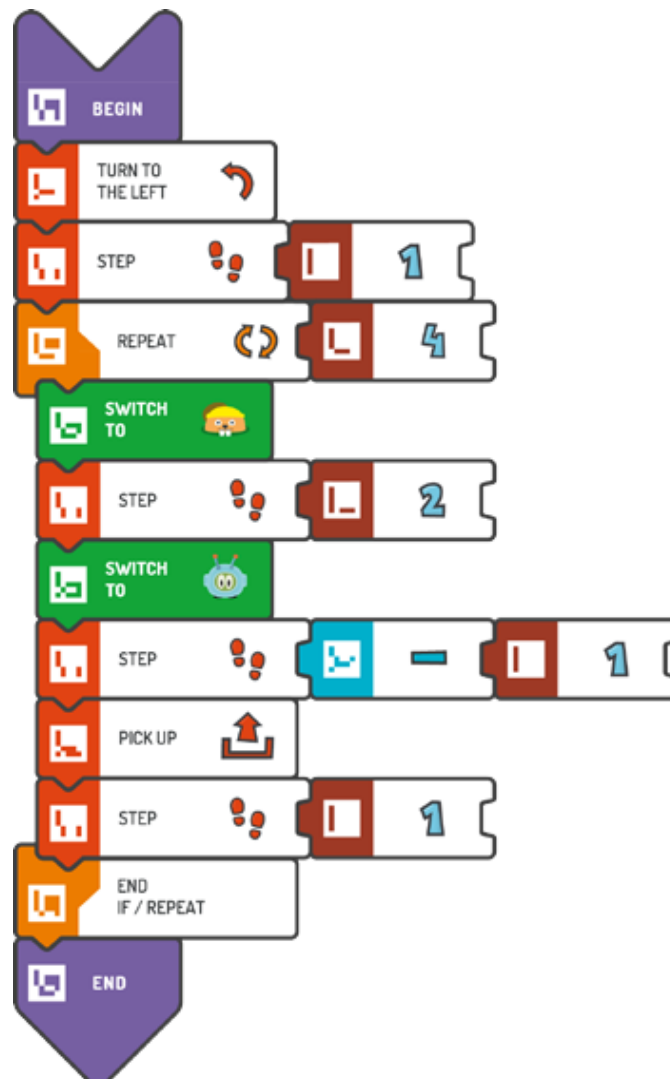
}

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**TURN TO THE LEFT**

**STEP (1)**

**REPEAT (4) {**

**SWITCH TO - YELLOW BEAVER:**

**STEP (2)**

**SWITCH TO - SCOTTIE:**

**STEP (-1)**

**PICK UP**

**STEP (1)**

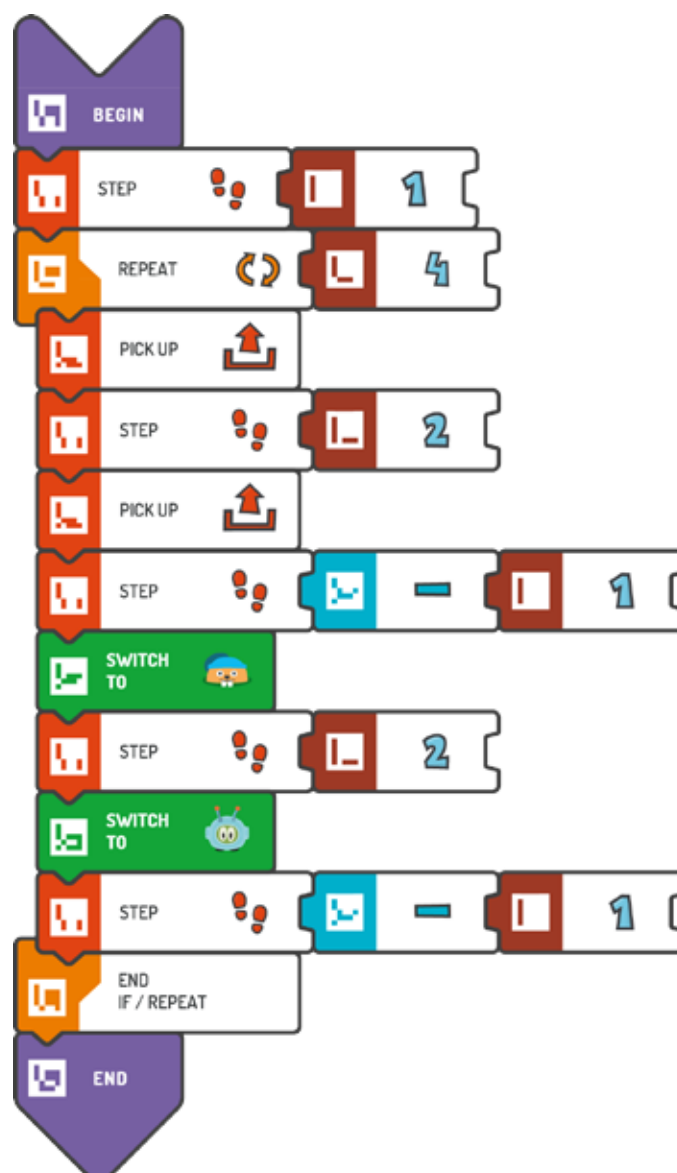
**}**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (1)**

**REPEAT (4) {**

**PICK UP**

**STEP (2)**

**PICK UP**

**STEP (-1)**

**SWITCH TO - BLUE BEAVER:**

**STEP (2)**

**SWITCH TO - SCOTTIE:**

**STEP (-1)**

**}**

**END**

# Module V

## Antarctica



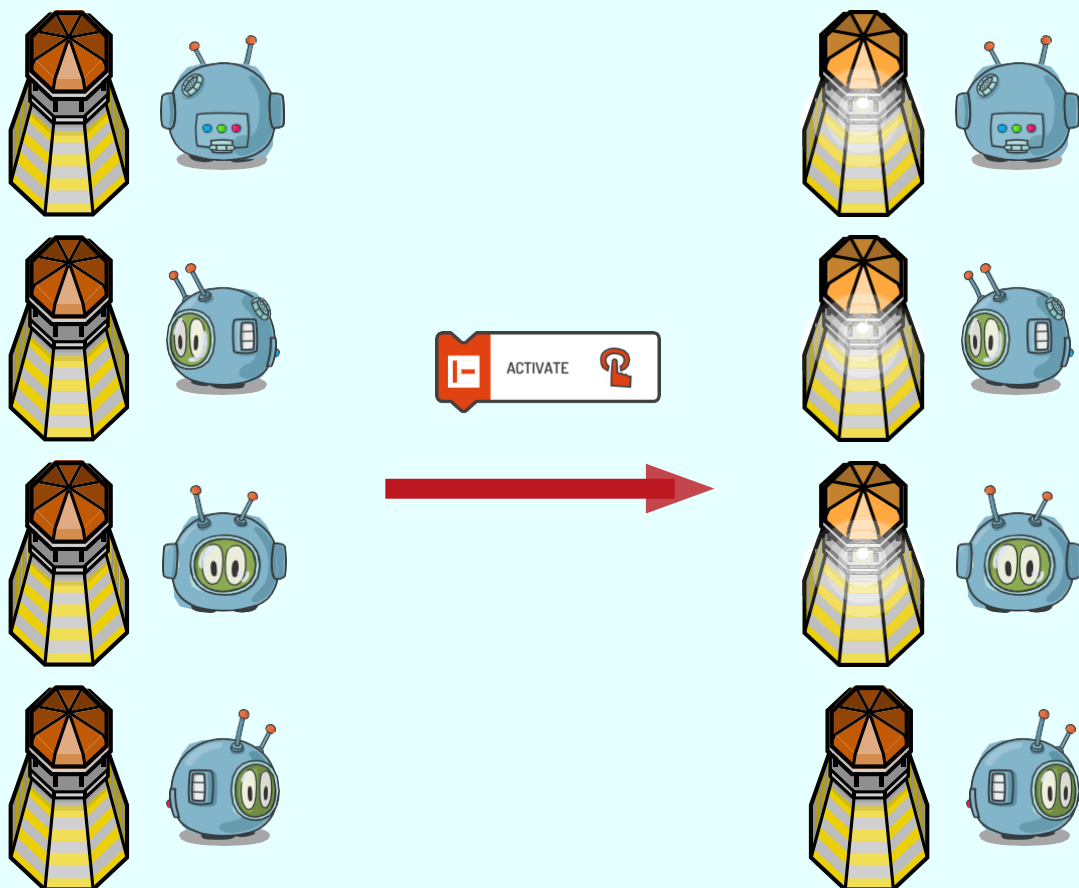
Scottie helpt de zeemannen. In deze opdracht zet Scottie alle lichten van de vuurtorens van het schiereiland aan, om de schepen die de onderdelen van zijn ruimteschip vervoeren naar de haven te leiden.

# MODULE INTRODUCTIE: interactie met Scottie zijn omgeving - ACTIVATE en PLACE tegels

In deze module blijft Scottie bezig met objecten en hij begint zijn omgeving te veranderen. De **ACTIVATE** en **PLACE** tegels worden uitgelegd.



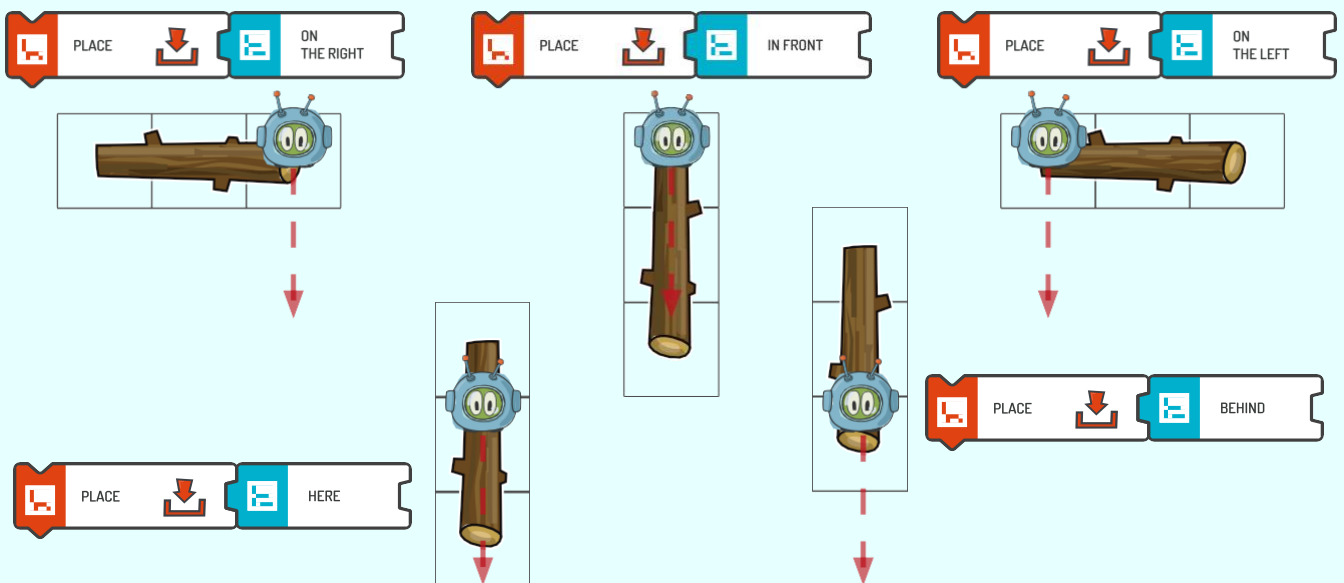
De **ACTIVATE** tegel kan in de code gebruikt worden om Scottie de vuurtoren aan te laten zetten, als hij er naast staat. Dit werkt alleen als Scottie **niet** met zijn rug naar de vuurtoren staat.



Deze module introduceert ook boomstammen die Scottie kan oppakken (met behulp van de **PICK UP** tegel).



Als Scottie de boomstam heeft opgepakt, kan hij deze vervolgens in vijf verschillende posities op het spelbord plaatsen. Gebruik hierbij de **HERE**, **IN FRONT**, **ON THE RIGHT**, **ON THE LEFT**, **BEHIND** tegels als parameters voor de **PLACE** tegels.



Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

# BEGIN

## STEP (1)

## TURN TO THE RIGHT

## STEP (4)

## ACTIVATE

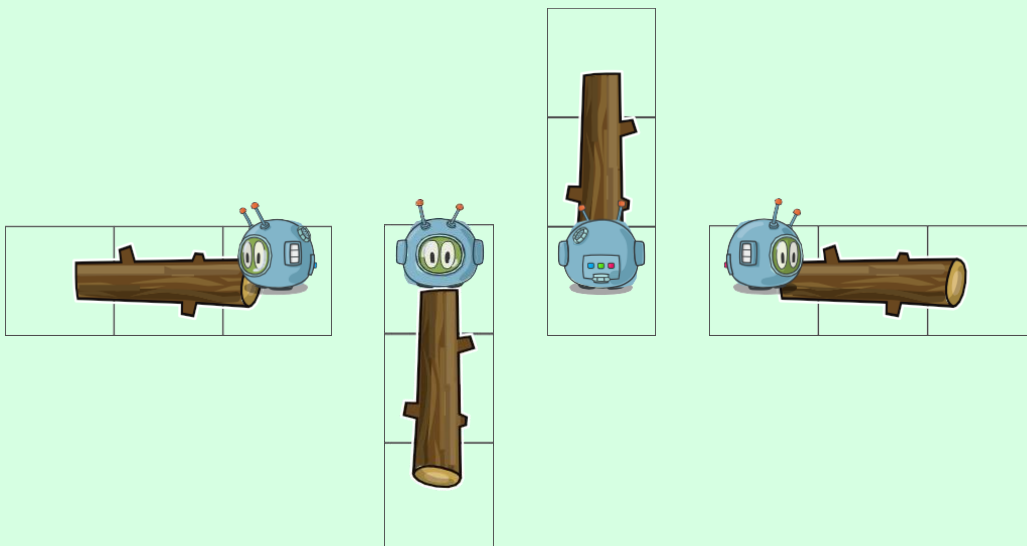
**END**

## De PLACE tegel

Om Scottie over een ravijn te krijgen, pak je een boomstam op en plaats je hem op de juiste plaats in de juiste richting.



Om een boomstam over de ravijn te leggen, moet Scottie één veld voor het ravijn staan. Verder moet je de juiste parameter voor de **PLACE** tegel gebruiken bijvoorbeeld **PLACE (IN FRONT)** als Scottie met zijn gezicht naar het ravijn staat.

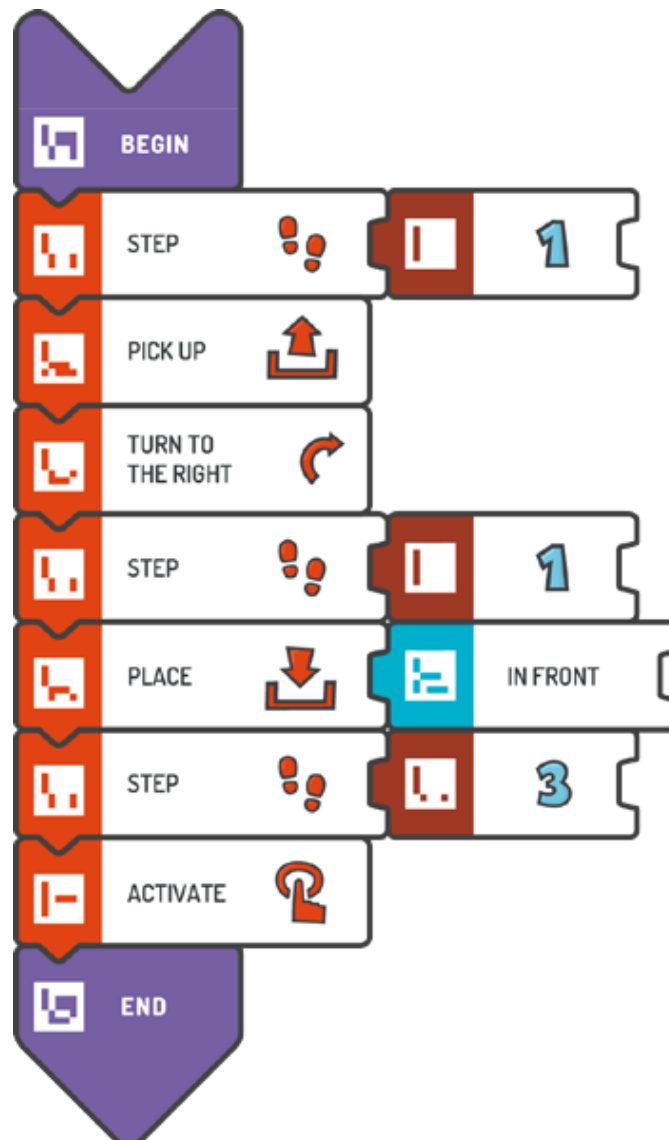


Voor het oppakken van een boomstam, moet Scottie op één van de drie velden staan waar de boomstam overheen ligt.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (1)**

**PICK UP**

**TURN TO THE RIGHT**

**STEP (1)**

**PLACE (IN FRONT)**

**STEP (3)**

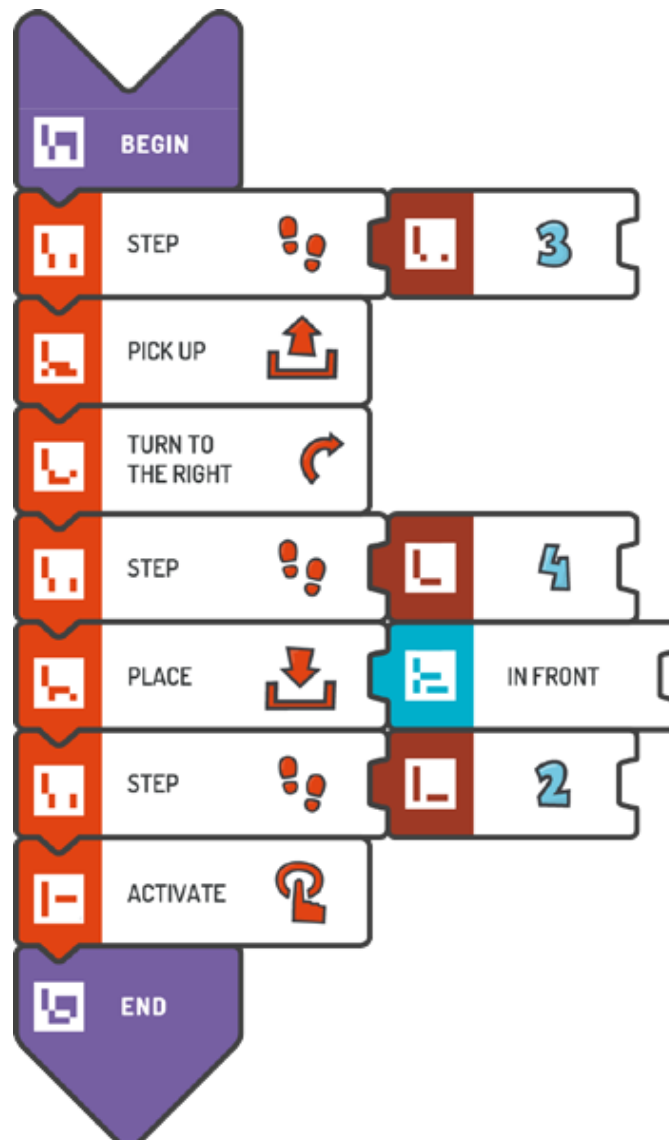
**ACTIVATE**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (3)**

**PICK UP**

**TURN TO THE RIGHT**

**STEP (4)**

**PLACE (IN FRONT)**

**STEP (2)**

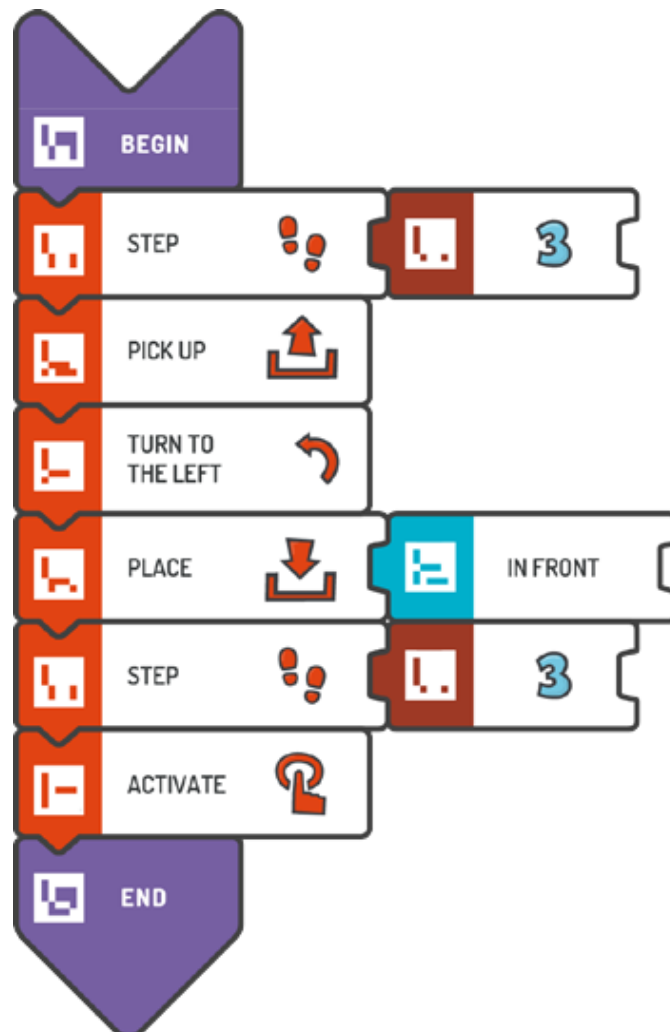
**ACTIVATE**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (3)**

**PICK UP**

**TURN TO THE LEFT**

**PLACE (IN FRONT)**

**STEP (3)**

**ACTIVATE**

**END**

# LET OP

Zorg er in de volgende opdrachten voor dat de leerlingen de **REPEAT** tegel gebruiken, nieuwe lussen maken en hun code steeds optimaliseren.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (1)**

**REPEAT (2) {**

**STEP (2)**

**PICK UP**

**PLACE (IN FRONT)**

**}**

**STEP (2)**

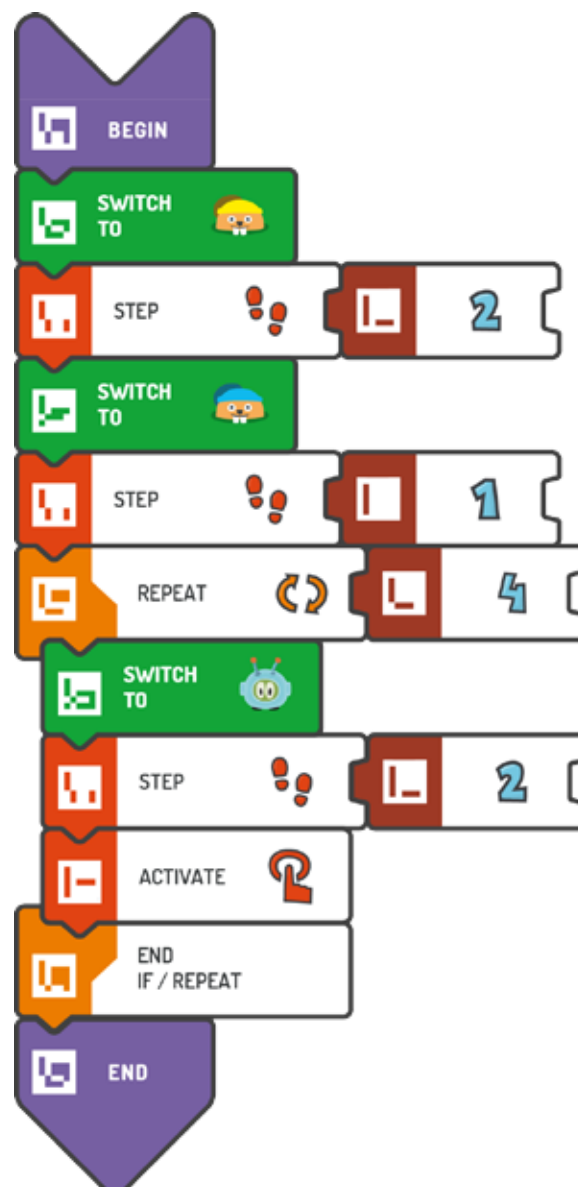
**ACTIVATE**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**SWITCH TO - YELLOW BEAVER:**

**STEP (2)**

**SWITCH TO - BLUE BEAVER:**

**STEP (1)**

**REPEAT (4) {**

**SWITCH TO - SCOTTIE:**

**STEP (2)**

**ACTIVATE**

**}**

**END**

## SWITCH TO de bevers en verplaats ijsschotsen

Wanneer je het omgaan met de bevers beheerst, kun je kleine ijsschotsen verplaatsen en nieuwe routes creëren voor Scottie. Gebruik Bever in deze opdracht.

Zorg ervoor dat de leerlingen de **REPEAT** lus gebruiken.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**SWITCH TO - YELLOW BEAVER:**

**STEP (2)**

**SWITCH TO - SCOTTIE:**

**TURN TO THE RIGHT**

**STEP (1)**

**REPEAT (2) {**

**STEP (1)**

**JUMP**

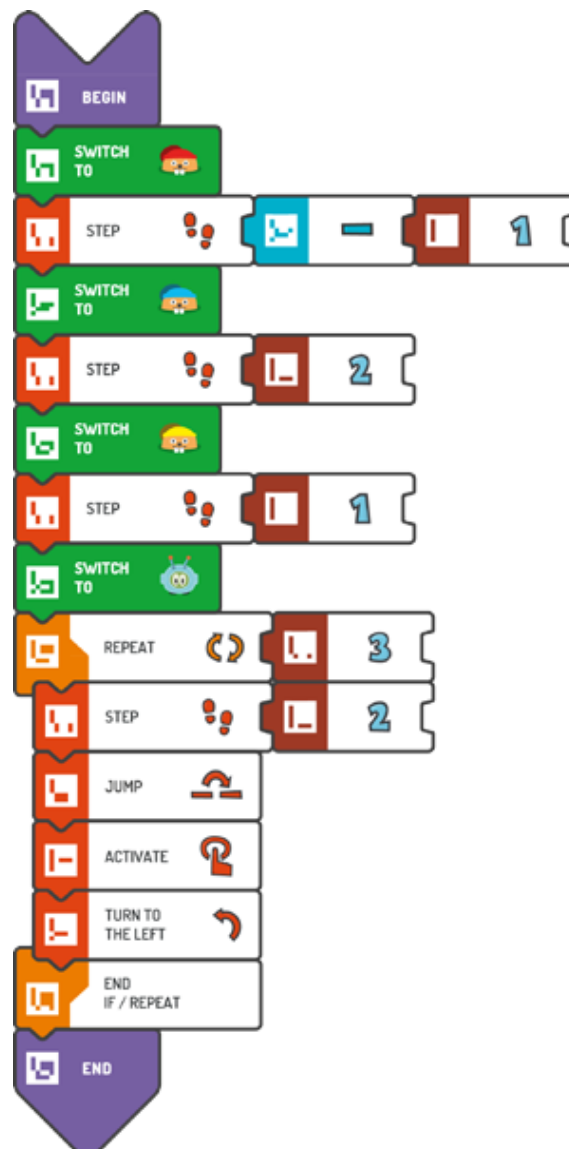
**}**

**ACTIVATE**

**END**



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

## BEGIN

SWITCH TO - RED BEAVER:

STEP (-1)

SWITCH TO - BLUE BEAVER:

STEP (2)

SWITCH TO - YELLOW BEAVER:

STEP (1)

SWITCH TO - SCOTTIE:

REPEAT (3) {

STEP (2)

JUMP

ACTIVATE

TURN TO THE LEFT

}

END

## **HINT:**

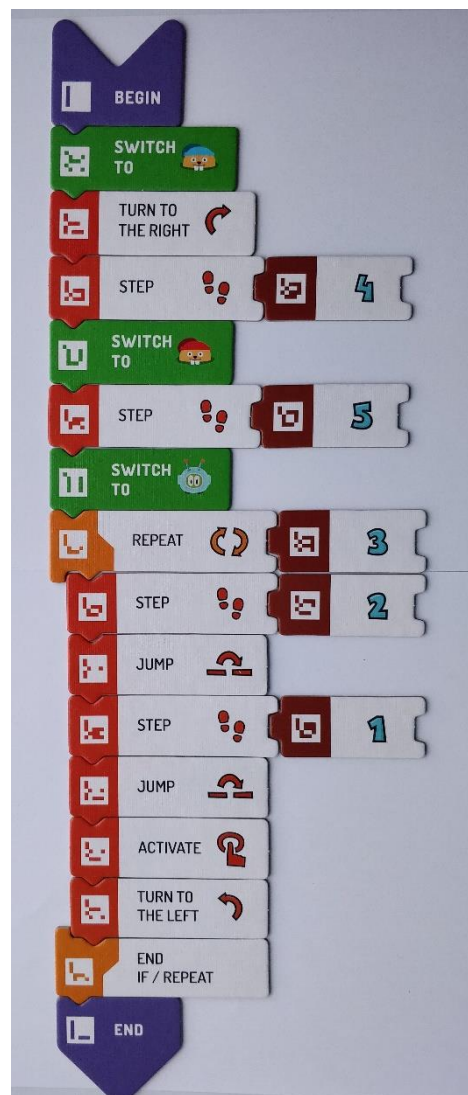
### **selecteer een karakter**

Selecteer een karakter bijvoorbeeld Bever om te programmeren, om de code te optimaliseren (zo kort mogelijk te maken).

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

## BEGIN

SWITCH TO - BLUE BEAVER:

TURN TO THE RIGHT

STEP (4)

SWITCH TO - RED BEAVER:

STEP (5)

SWITCH TO - SCOTTIE:

REPEAT (3) {

STEP (1)

JUMP

STEP (1)

JUMP

ACTIVATE

TURN TO THE LEFT

}

END

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

## BEGIN

SWITCH TO - BLUE BEAVER:

TURN TO THE RIGHT

STEP (2)

SWITCH TO - RED BEAVER:

STEP (2)

SWITCH TO - SCOTTIE:

REPEAT (3) {

JUMP

JUMP

ACTIVATE

TURN TO THE LEFT

}

END

# Module VI

## Azië



**Ik heb je hulp nodig. De bijzonder flexibele en sterke bamboestengels zullen een perfecte steiger maken voor het repareren van mijn ruimteschip. Verzamel alsjeblieft alle bamboestengels voor me.**

# MODULE INTRODUCTIE: voorwaardelijk commando - REPEAT WHILE

Net als de **REPEAT** lus, laat de **REPEAT WHILE** lus Scottie (of een ander karakter) commando's meerdere keren herhalen binnen de lus. Bij de **REPEAT WHILE** lus zullen ze echter herhaald worden, tot aan een bepaalde voorwaarde voldaan is. Zo kunnen we voorwaardelijke lussen maken.

Een alledaags voorbeeld van een **voorwaardelijke lus**:

Stel je voor dat de ouders van onze leerlingen ze met de auto van school naar huis brengen. Soms is het moeilijk om met zekerheid te bepalen, welke stappen ze moeten uitvoeren en hoe vaak ze deze stappen moeten uitvoeren om thuis te komen. In de meeste gevallen gebruiken we dan onbewust de **REPEAT WHILE** lus. Hoe komt dat?

1. We controleren de voorwaarde "is er een rechte weg voor me?"
2. Als dit zo is, gaan we rechtdoor en weer terug naar punt (1) – we controleren bijvoorbeeld opnieuw of de weg voor ons een rechte weg is.
3. Als er niet aan de voorwaarde voldaan wordt, bijvoorbeeld als er een bocht komt, is de lus doorbroken. We gaan niet meer door met het controleren van de voorwaarde van punt (1) en gaan door met het uitvoeren van het volgende commando buiten de lus. In dit geval slaan we af de juiste richting in.

## Voorwaardelijke lussen in Scottie Go!

Laten we kijken hoe de voorwaardelijke lus gebruikt is in dit voorbeeld. In deze opdracht mag Scottie alleen de combinatie van **STEP** tegels en de nummer **1** tegel gebruiken. Dit maakt het makkelijker om uit te leggen hoe deze lus gebruikt wordt.

1. We controleren de voorwaarde "is er een vrije weg voor me?". Aan deze voorwaarde wordt voldaan, het veld voor Scottie is leeg.



2. Scottie begint met het uitvoeren van de commando's die binnen de lus geplaatst zijn. **Hij gaat een stap naar voren.** Het programma gaat opnieuw de voorwaarde controleren (punt 1.)



3. Scottie controleert opnieuw de voorwaarde "is er een vrije weg voor me?". **Er wordt weer aan de voorwaarde voldaan.**



4. Scottie voert opnieuw het commando binnen de lus uit, te weten - **STEP 1**. Dan gaat hij terug naar het controleren van de voorwaarde in punt 1.



5. Scottie controleert opnieuw de voorwaarde "is er een vrije weg voor me?". **Deze keer is dat niet het geval, er wordt niet aan de voorwaarde voldaan.**



6. **De lus is doorbroken.** Scottie begint de code buiten de **lus** te volgen: **STEP (1)** en **PICK UP**.



# De REPEAT WHILE tegel gebruiken

Gebruik in de volgende opdrachten alleen de combinatie van de **STEP (1)** tegels en de **REPEAT WHILE** lus.

Soms kun je in plaats van de **REPEAT WHILE (IN FRONT < EMPTY)** tegel combinatie, de **REPEAT WHILE (ON THE RIGHT < OBSTACLE)** of **REPEAT WHILE (HERE < EMPTY)** gebruiken.



In het laatste voorbeeld kun je eventueel het programma nog een regel inkorten.

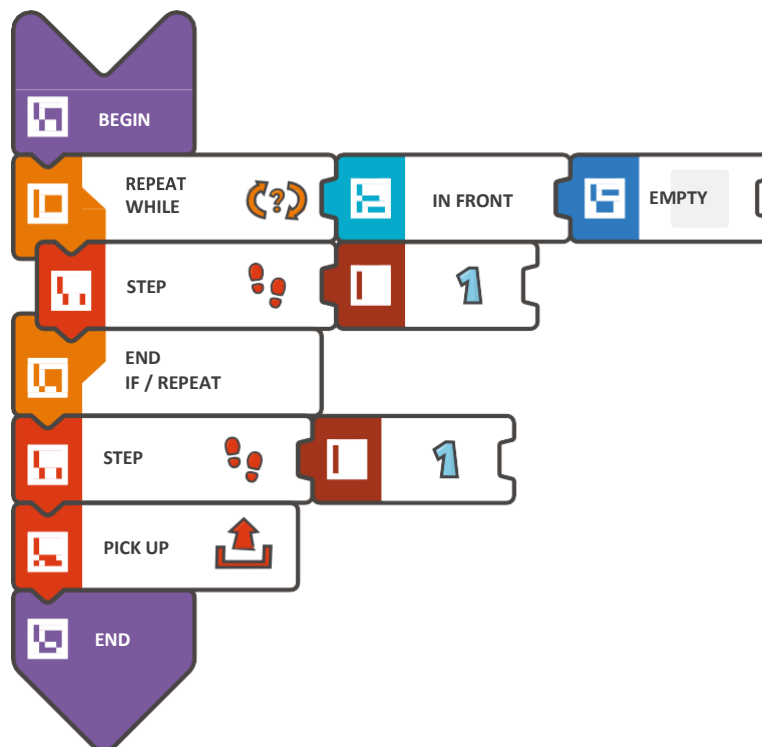
Gebruik voor het introduceren van de **REPEAT WHILE** lus, het "real life" voorbeeld zoals aangegeven in de introductie van de module, of bedenk een eigen voorbeeld.

Wanneer je consequent bent in het gebruik van de combinatie van de **STEP (1)** tegels met de **REPEAT WHILE** tegel in de volgende opdrachten, zorgt dit ervoor dat de leerlingen deze lus structuur steeds meer als vanzelfsprekend zullen zien. Denk eraan om de leerlingen alleen de **STEP (1)** tegels te laten gebruiken.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT WHILE (IN FRONT < EMPTY) {**

**STEP (1)**

**}**

**STEP (1)**

**PICK UP**

**END**

# HINT

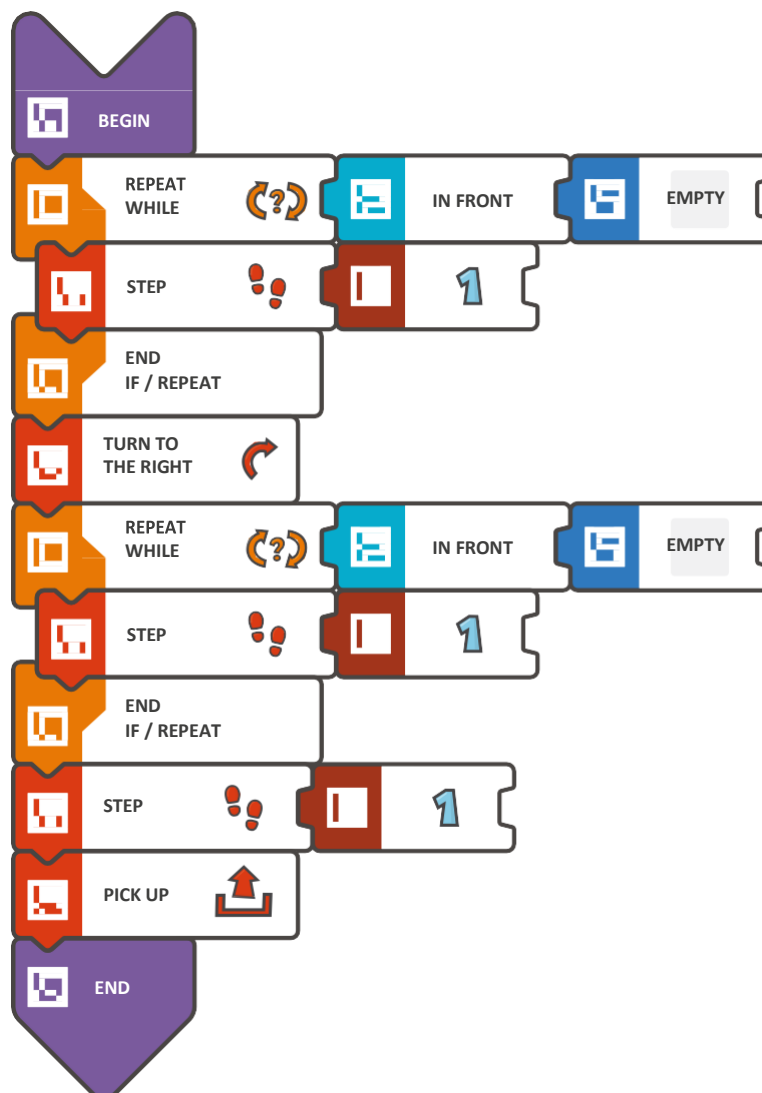
Gebruik twee **REPEAT WHILE** lussen in deze opdracht.

Optioneel, kunnen de leerlingen in de tweede lus van deze opdracht in plaats van de **REPEAT WHILE (IN FRONT < EMPTY)** combinatie, de **REPEAT WHILE (HERE < EMPTY)** combinatie gebruiken. Na een aantal aanpassingen zou de code dan een regel korter moeten worden.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT WHILE (IN FRONT < EMPTY) {**

**STEP (1)**

**}**

**TURN TO THE RIGHT**

**REPEAT WHILE (IN FRONT < EMPTY){**

**STEP (1)**

**}**

**STEP (1)**

**PICK UP**

**END**

# HINT

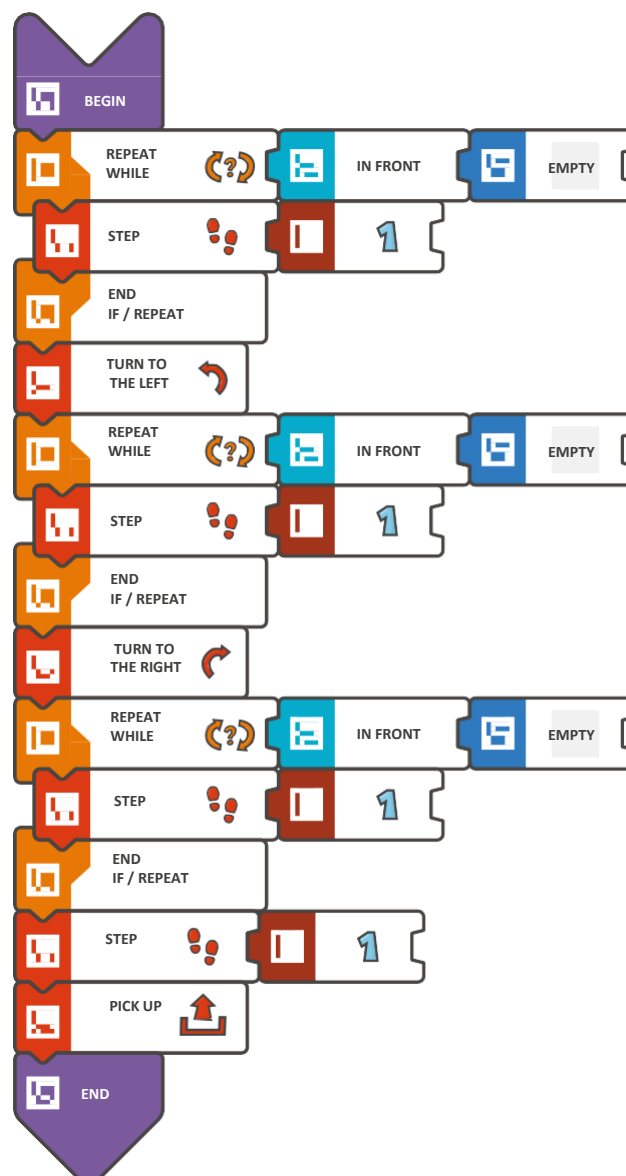
Zoals we in de vorige opdrachten konden zien, is de **REPEAT WHILE** lus een slimmere versie van de **REPEAT** lus, omdat de reeks commando's binnen de lus uitgevoerd worden, zolang er aan de voorwaarde voldaan wordt. Wanneer er niet meer aan de voorwaarde voldaan wordt, wordt de lus doorbroken.

Gebruik in deze opdracht drie **REPEAT WHILE** lussen.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT WHILE (IN FRONT < EMPTY) {**

**STEP (1)**

**}**

**TURN TO THE LEFT**

**REPEAT WHILE (IN FRONT < EMPTY) {**

**STEP (1)**

**}**

**TURN TO THE RIGHT**

**REPEAT WHILE (IN FRONT < EMPTY) {**

**STEP (1)**

**}**

**STEP (1)**

**PICK UP**

**END**

# REPEAT WHILE met andere voorwaarden

Je kunt de **REPEAT WHILE** lussen gebruiken met verschillende voorwaarden om er voor te zorgen dat de commando's binnen de lus uitgevoerd kunnen worden.

## Waar (vanuit Scottie's gezichtspunt)?



## Wat?



LEEG

VERZAMELOBJECT

OBSTAKEL

HEUVEL

(essentieel in latere modules)



GEMARKEERD  
VIERKANT  
(essentieel in  
latere modules)

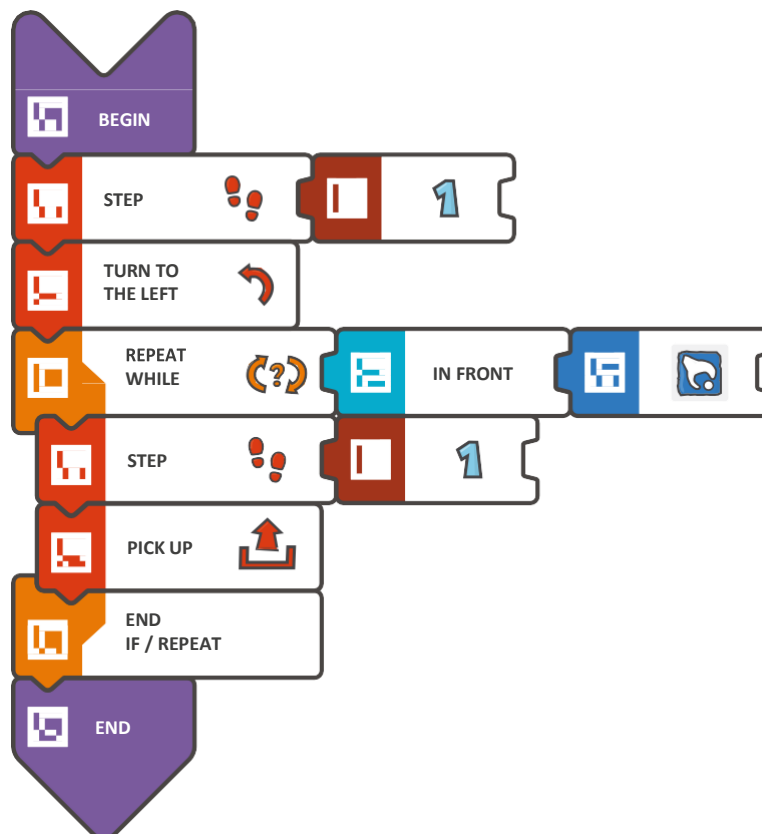
GEMARKEERD  
KRUIS  
(essentieel in  
latere modules)

ACTIEVELD  
(essentieel in  
latere modules)

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**STEP (1)**

**TURN TO THE LEFT**

**REPEAT WHILE (IN FRONT < COLLECTABLE OBJECT) {**

**STEP (1)**

**PICK UP**

**}**

**END**

## REPEAT WHILE genest in een lus

Indien nodig, kun je de **REPEAT WHILE** lussen meerdere keren gebruiken. Creëer een **REPEAT** lus en plaats hier een andere **REPEAT WHILE** lus in. Een lus in een andere lus – probeer het maar, het is niet zo moeilijk!

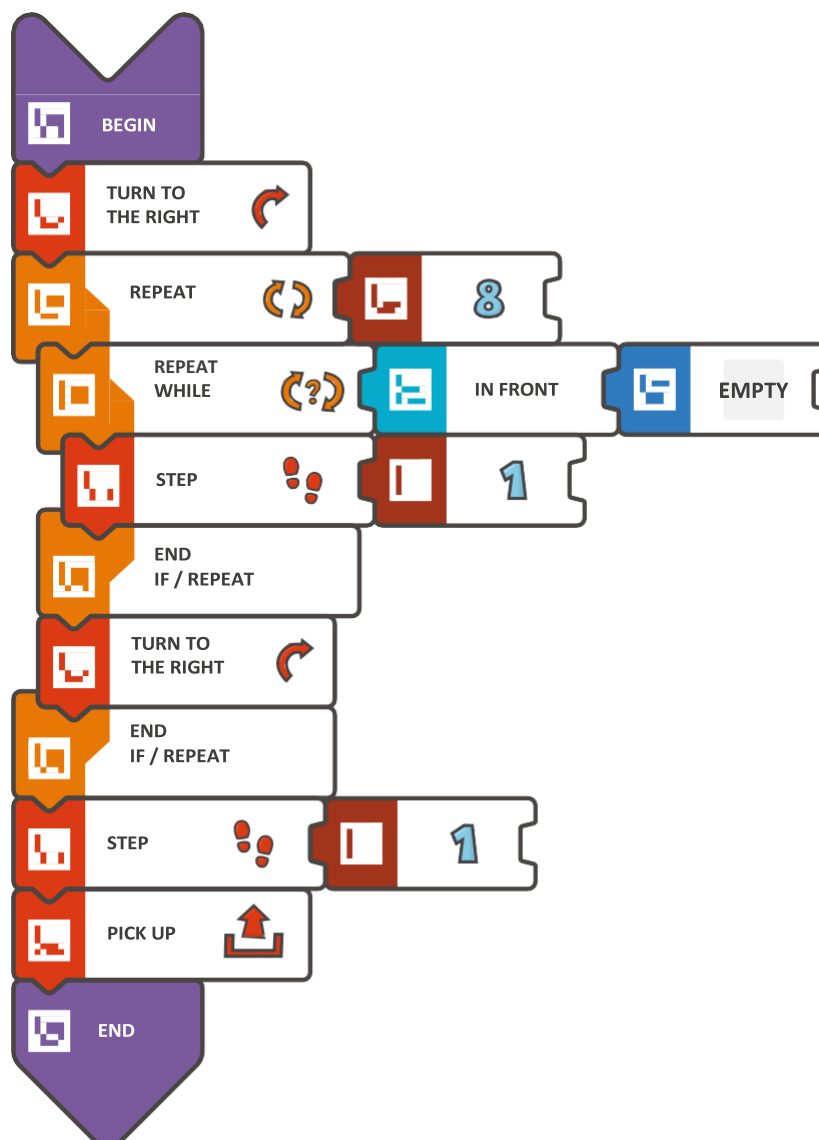
Meer gevorderde leerlingen kunnen proberen om deze opdracht op te lossen door voorwaardelijke commando's te gebruiken zoals deze:

```
IF (ON THE RIGHT < EMPTY){  
    TURN TO THE RIGHT  
}
```

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**TURN TO THE RIGHT**

**REPEAT (8) {**

**REPEAT WHILE (IN FRONT < EMPTY) {**

**STEP (1)**

**}**

**TURN TO THE RIGHT**

**}**

**STEP (1)**

**PICK UP**

**END**

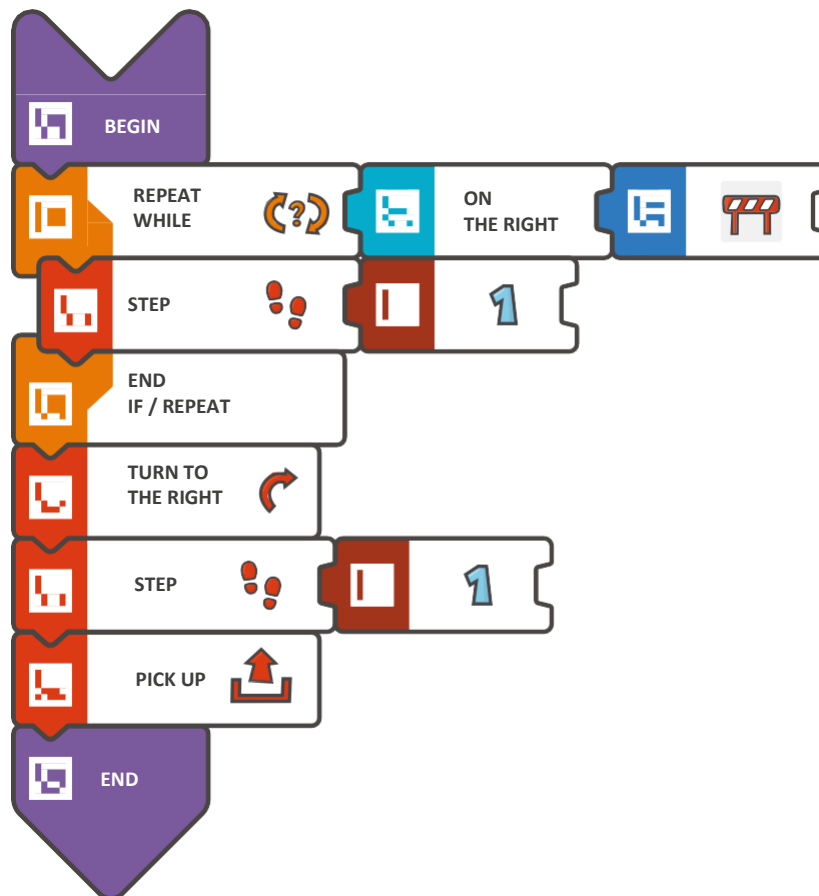
# LET OP!

Gebruik de **REPEAT WHILE** lus maar één keer.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT WHILE (ON THE RIGHT < OBSTACLE) {**

**STEP (1)**

**}**

**TURN TO THE RIGHT**

**STEP (1)**

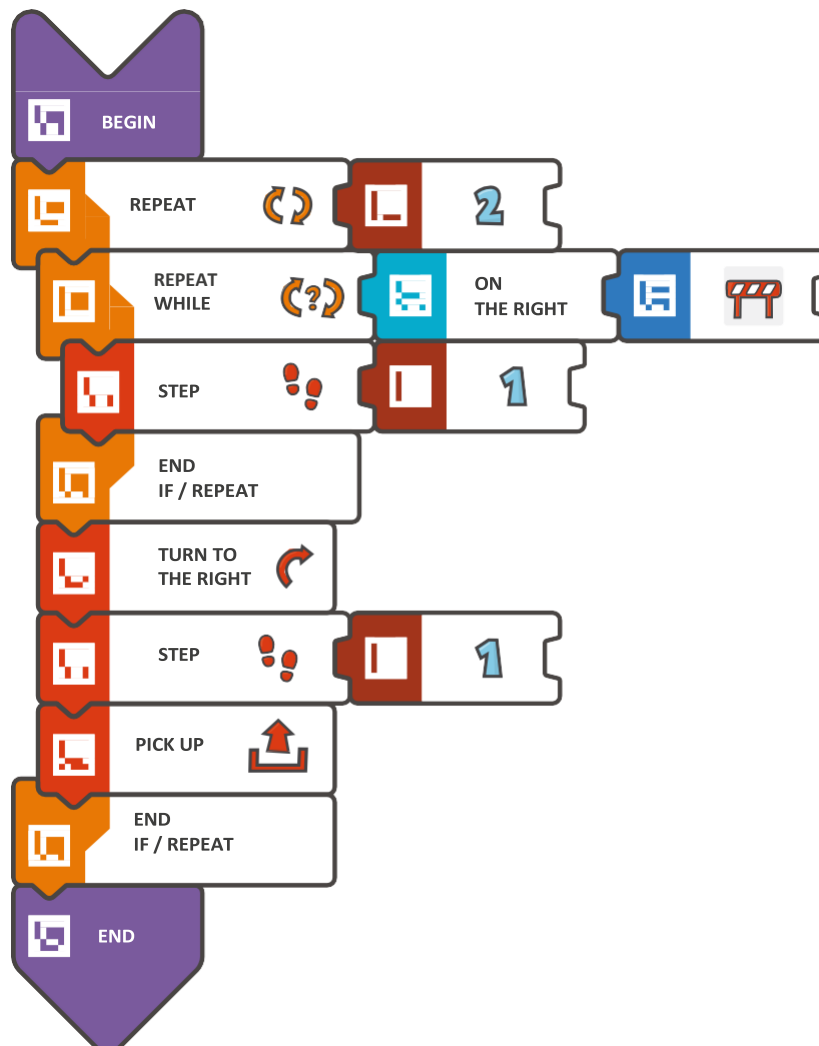
**PICK UP**

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (2) {**

**REPEAT WHILE (ON THE RIGHT < OBSTACLE) {**

**STEP (1)**

**}**

**TURN TO THE RIGHT**

**STEP (1)**

**PICK UP**

**}**

**END**

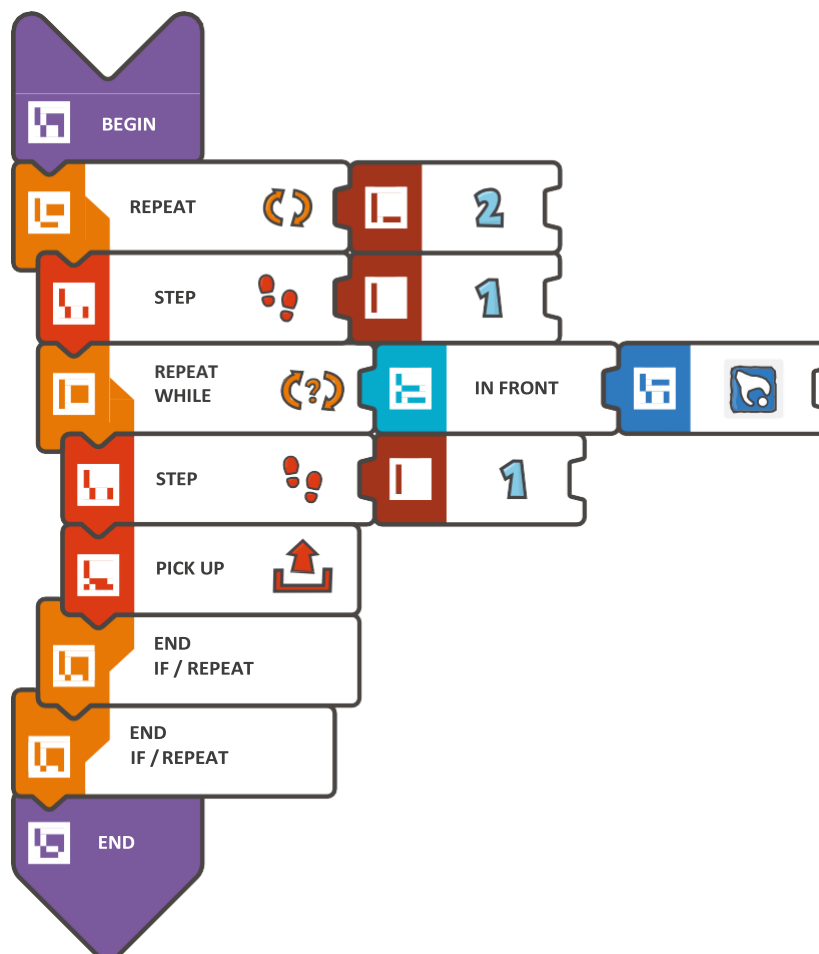
# LET OP!

Gebruik zowel de **REPEAT WHILE** lus en de **REPEAT** lus maar één keer in deze opdracht.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

```
BEGIN
  REPEAT (2) {
    STEP (1)
    REPEAT WHILE (IN FRONT < COLLECTABLE OBJECT) {
      STEP (1)
      PICK UP
    }
  }
END
```

# Module VII

## Australië



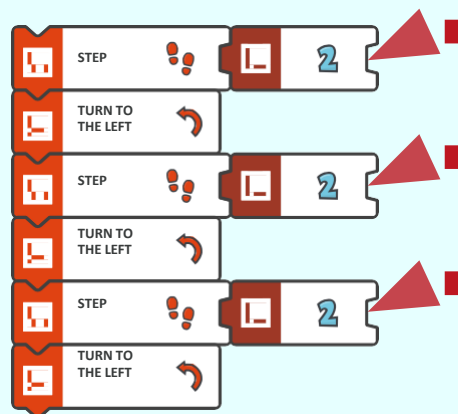
In deze module wordt het begrip van een **VARIABLE** uitgelegd. Bouw asfaltwegen voor Scottie zodat hij in het dorp onderdelen voor zijn ruimteschip kan verzamelen.

# MODULE INTRODUCTIE: VARIABLES (VARIABLEN)

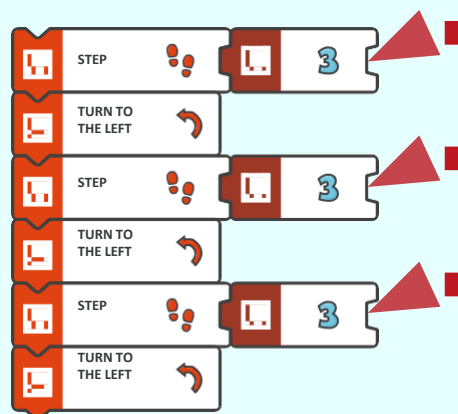
Je kunt **variabelen** voorstellen als **containers**. Je kunt verschillende gegevens in zo'n container stoppen, bijvoorbeeld getallen en deze opslaan, zodat je ze later kunt gebruiken.

Terwijl een computerprogramma uitgevoerd wordt, kan het in deze containers kijken om de huidige inhoud te controleren en te lezen, of om de inhoud te veranderen. Om het programma te laten weten waar het moet kijken (het is makkelijk voor te stellen dat er meer dan één **variabele** in een programma is), moeten **variabelen** namen hebben.

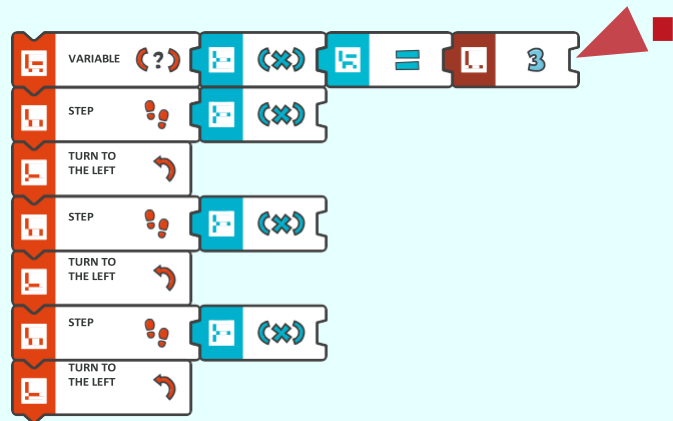
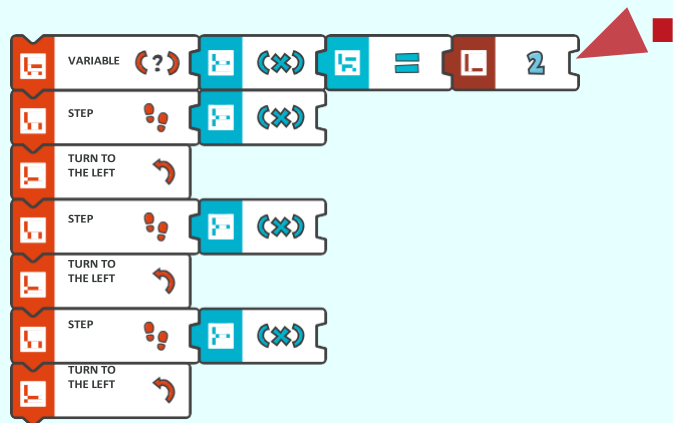
Je kunt bijvoorbeeld een **variabele** gebruiken om te bepalen hoeveel stappen Scottie moet zetten. Stel voor dat Scottie in onderstaande opdracht steeds opnieuw twee commando's uit moet voeren – twee stappen vooruit en **TURN TO THE LEFT**.



In de volgende opdracht (zie onderstaande afbeelding) zijn Scotties bewegingen bijna hetzelfde. Het enige wat verandert, is het aantal stappen (drie stappen en **TURN TO THE LEFT**). Het is duidelijk dat je de zojuist gemaakte code op drie plaatsen moet aanpassen om hem voor onderstaande opdracht te laten werken.



Ga terug naar de eerste opdracht die we als voorbeeld hebben gebruikt (zie onderstaande afbeelding), maar deze keer vereenvoudig je de oplossing door **variabelen** te gebruiken. Gebruik met elke stap, in plaats van een getal, een **variabele**. In het eerste deel van het programma benoem je de **variabele** met behulp van het '=' teken. Je definieert de waarde als 2 (het aantal stappen).



Wanneer de **variabele** is gebruikt, hoef je voor het oplossen van de volgende opdracht alleen de eerste regel code te veranderen, waar de waarde voor **variabele X** is bepaald.

Zo zie je, dat het gebruik van **variabelen** vaak de tijd voor het aanpassen van code, kan verkorten.

**Oftewel: door het gebruik van variabelen wordt de code netter en makkelijker aan te passen.**

Om te zien hoe variabelen werken, dien je de volgende stappen uit te voeren:

1. Benoem een **variabele** door deze een naam te geven en definieer de oorspronkelijke waarde.



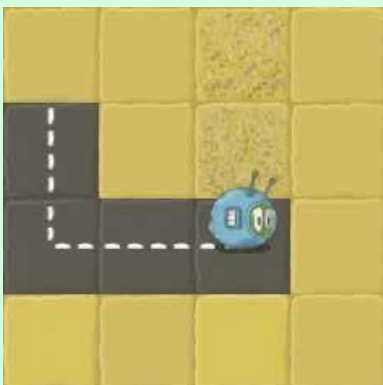
2. Gebruik deze **variabele** in de code door de naam van de variabele te gebruiken in plaats van de waarde.



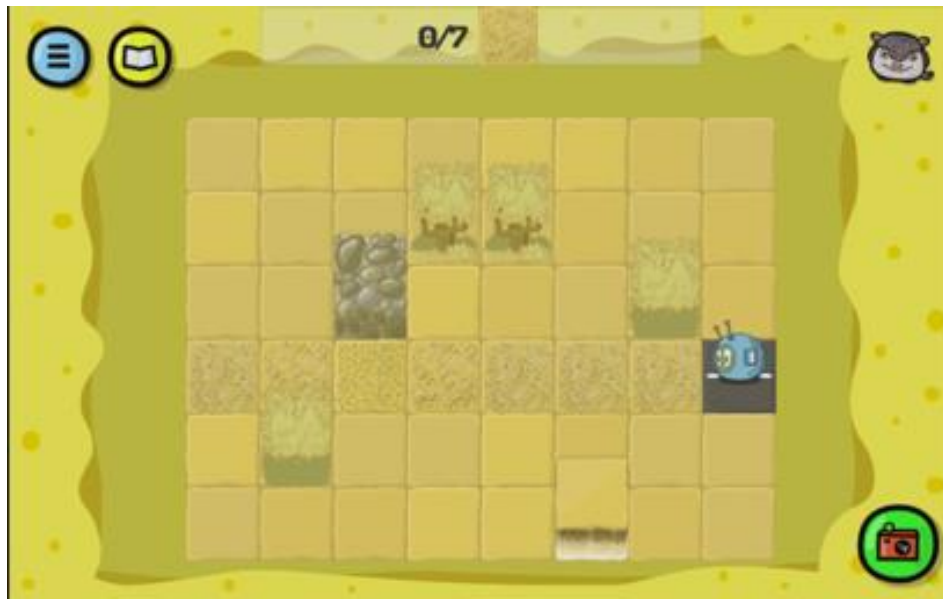
# Asfaltwegen bouwen

Als Scottie over grind loopt, bouwt hij asfaltwegen.

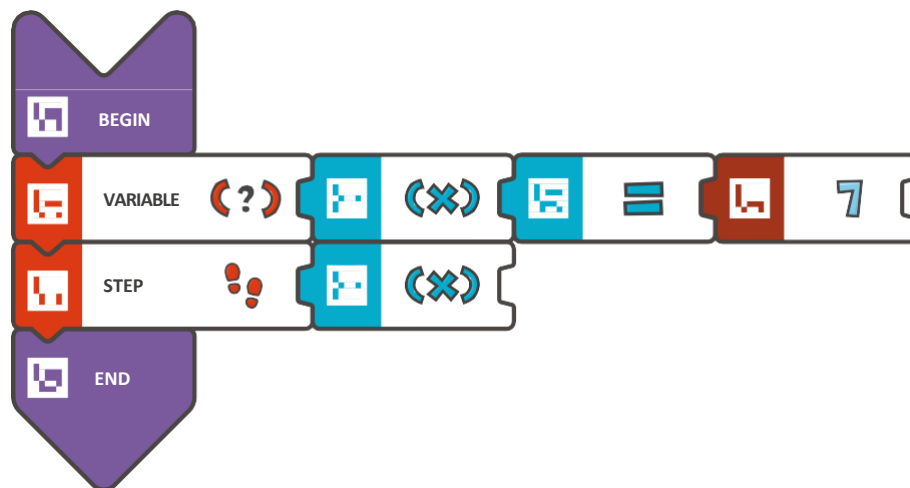
Denk er aan om een variabele X = aantal stappen te maken en roep deze variabele aan in de code.



Opdracht:



Voorbeeldoplossing:





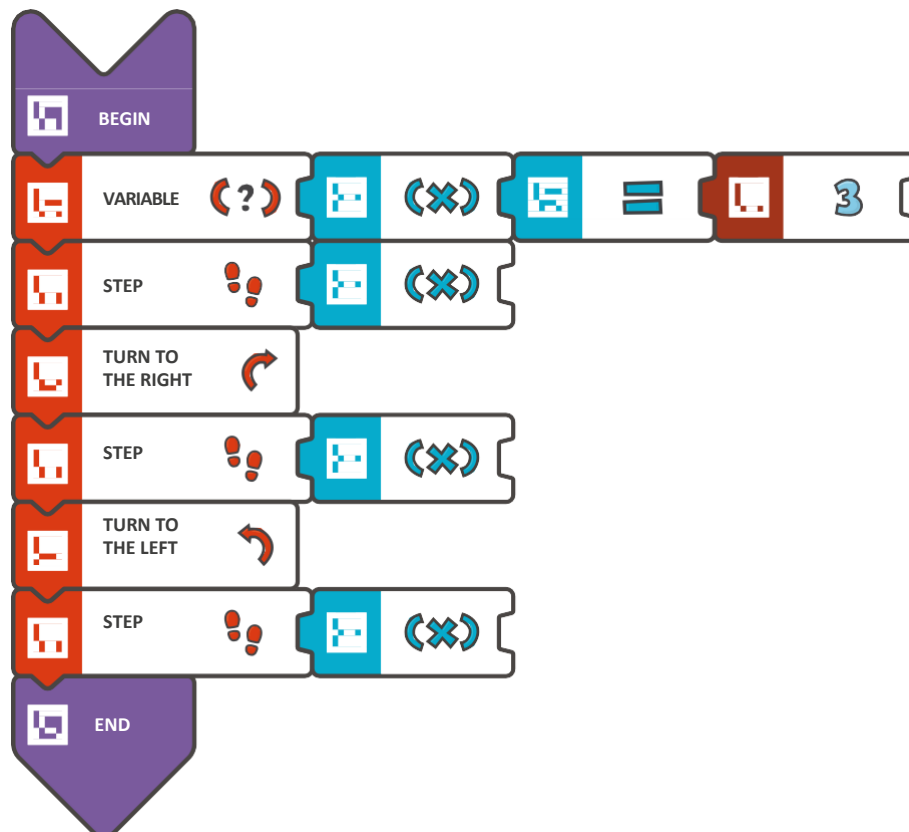
Voorbeeldoplossing (in tekstformaat):

```
BEGIN
    VARIABLE X=7
    STEP (X)
END
```

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**VARIABLE X=3**

**STEP (X)**

**TURN TO THE RIGHT**

**STEP (X)**

**TURN TO THE LEFT**

**STEP (X)**

**END**

# LET OP!

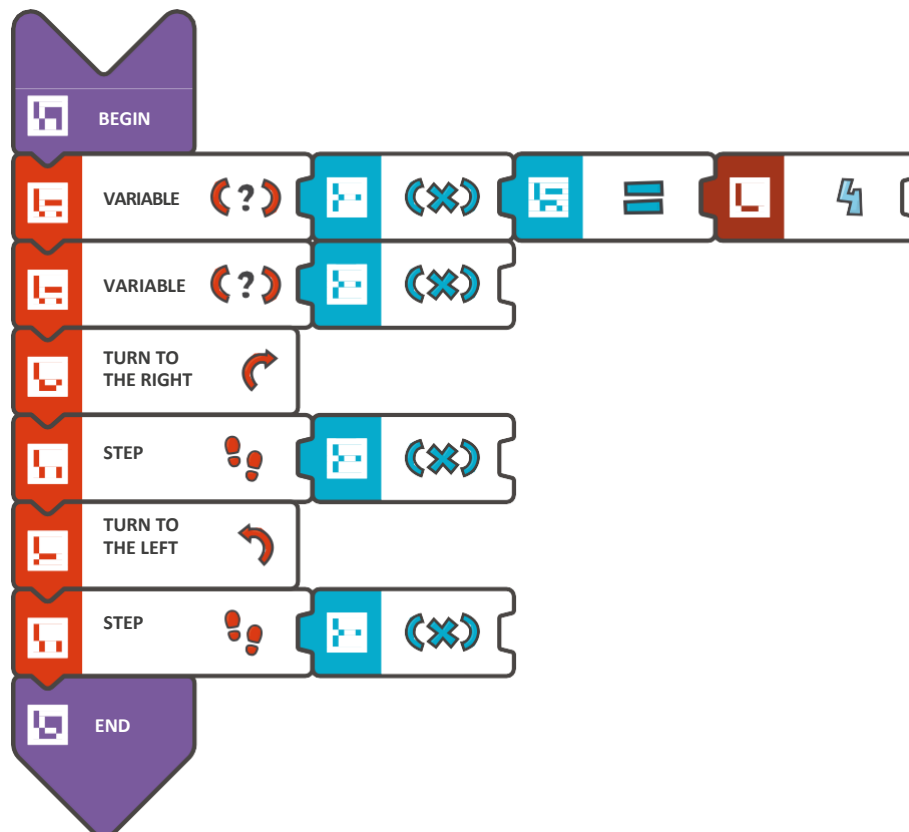
Je vorige code kan deze opdracht ook oplossen.

**Je moet alleen de waarde van variabele X aanpassen.**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**VARIABLE X=4**

**STEP (X)**

**TURN TO THE RIGHT**

**STEP (X)**

**TURN TO THE LEFT**

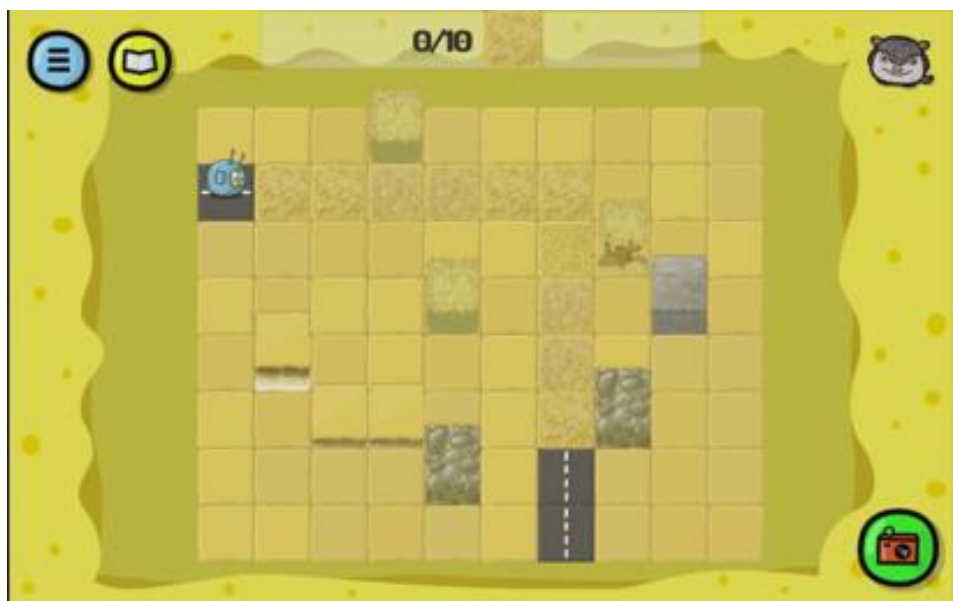
**STEP (X)**

**END**

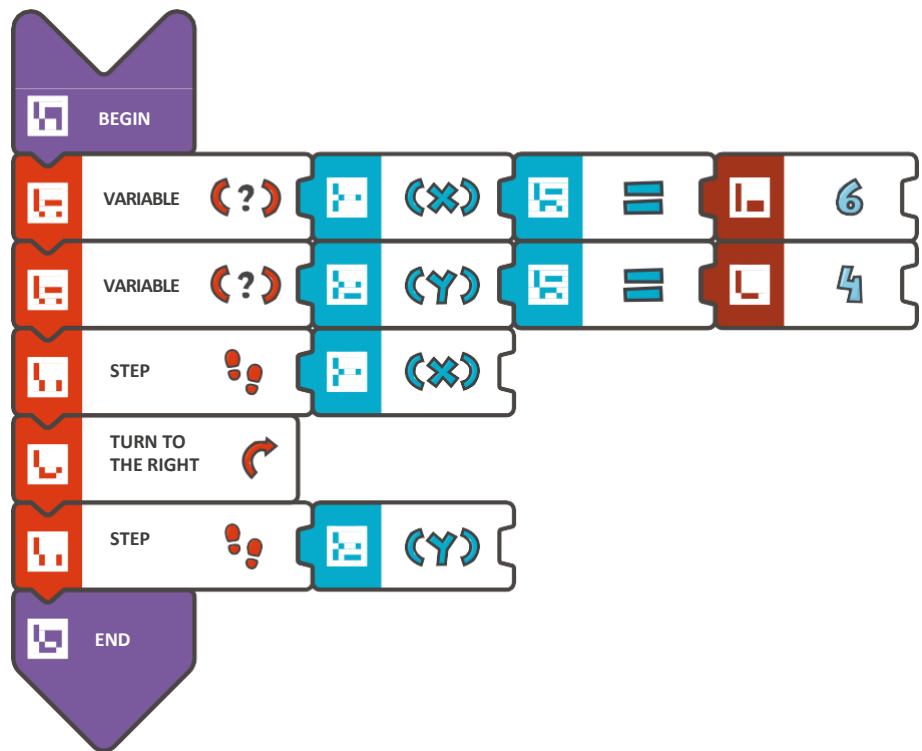
# X en Y variabelen

In Scottie Go! kan je twee variabelen: **X** en **Y** maken in dezelfde code.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

# BEGIN

**VARIABLE X=6**

**VARIABLE Y=4**

## STEP (X)

## TURN TO THE RIGHT

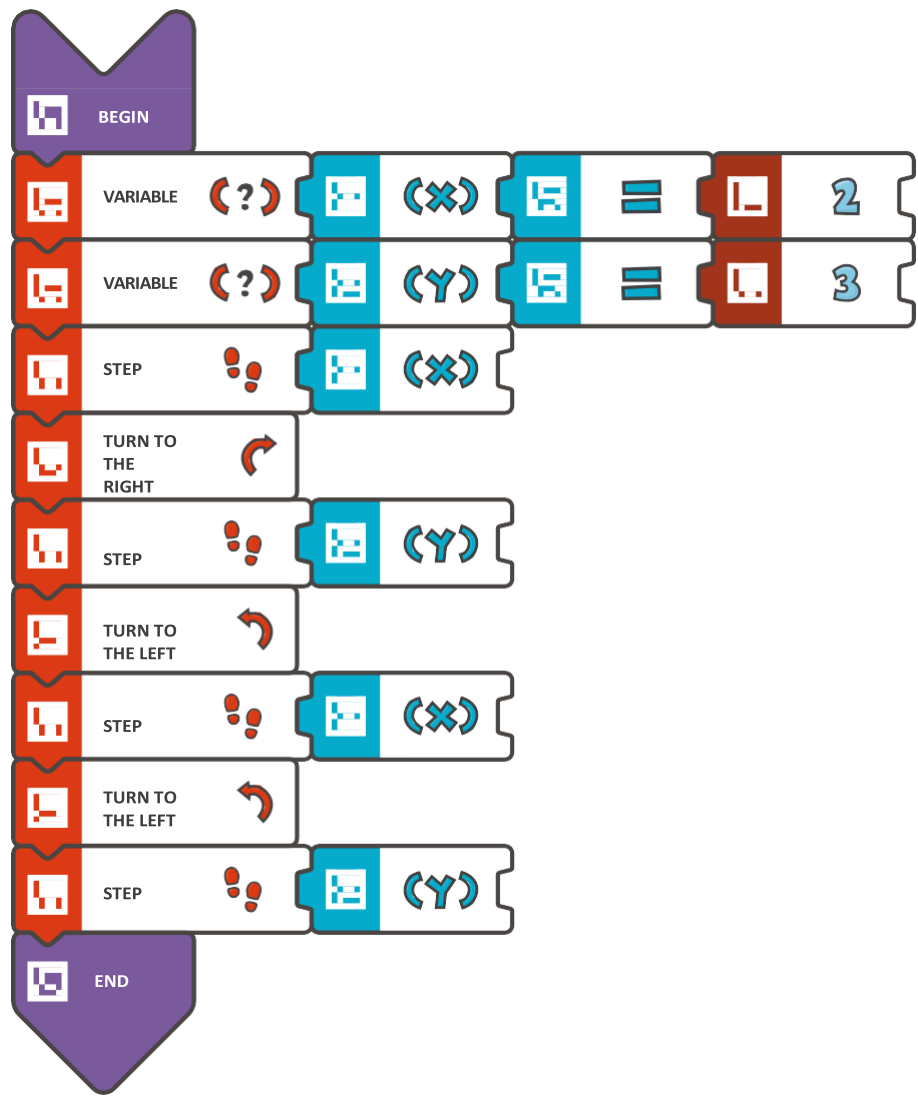
## STEP (Y)

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**VARIABLE X=2**

**VARIABLE Y=3**

**STEP (X)**

**TURN TO THE RIGHT**

**STEP (Y)**

**TURN TO THE LEFT**

**STEP (X)**

**TURN TO THE LEFT**

**STEP (Y)**

**END**

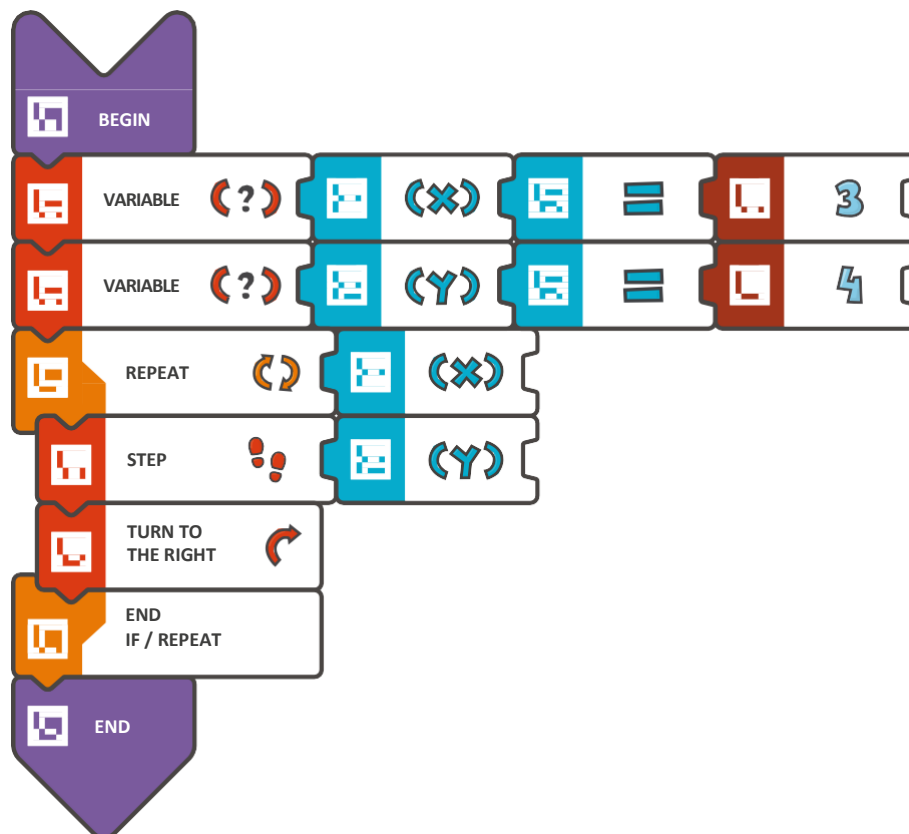
# Variabelen en lussen

Gebruik twee variabelen: **X** en **Y** en een **REPEAT** lus in deze opdracht. Gebruik variabele **X** voor het aantal herhalingen van de lus, en gebruik variabele **Y** voor het aantal stappen.

Opdracht:



Voorbeeldoplossing:





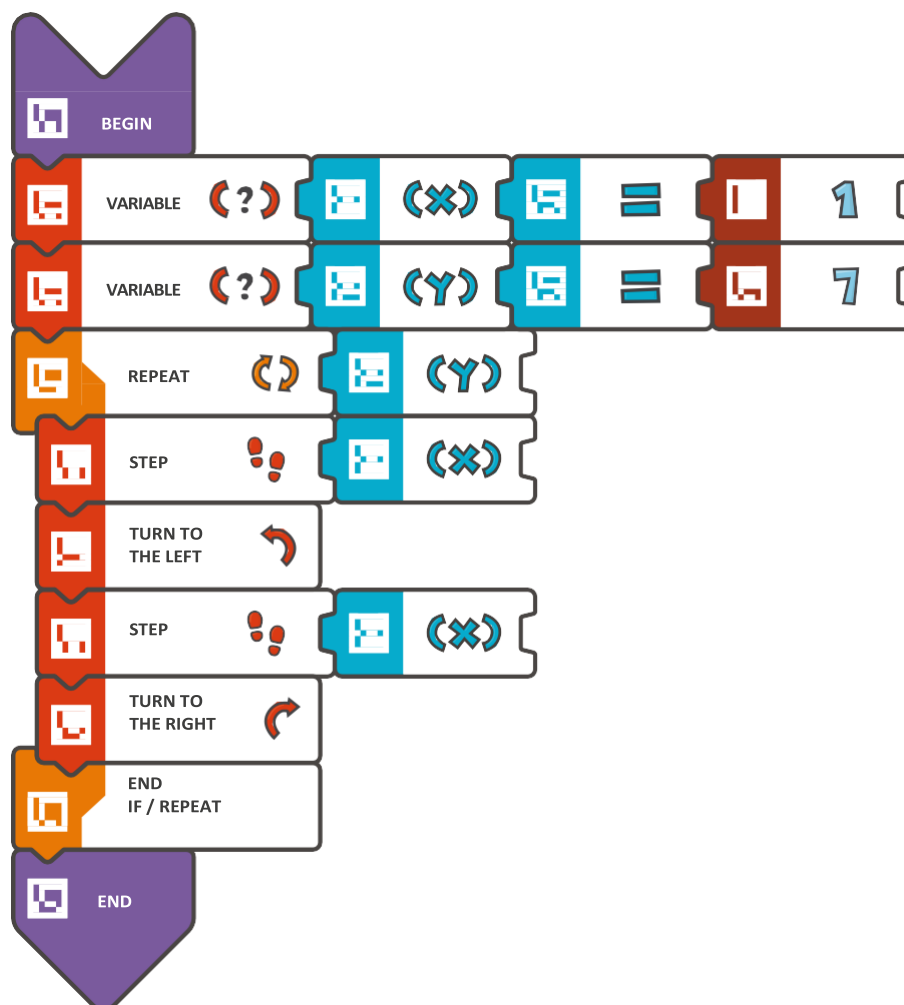
Voorbeeldoplossing (in tekstformaat):

```
BEGIN
  VARIABLE X=3
  VARIABLE Y=4
  REPEAT (X) {
    STEP (Y)
    TURN TO THE RIGHT
  }
END
```

Opdracht (in deze opdracht is het de bedoeling dat de leerlingen twee variabelen gebruiken.  
In dit geval dient een ervan het aantal herhalingen van de lus te bevatten):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**VARIABLE X=1**

**VARIABLE Y=7**

**REPEAT (Y) {**

**STEP (X)**

**TURN TO THE LEFT**

**STEP (X)**

**TURN TO THE RIGHT**

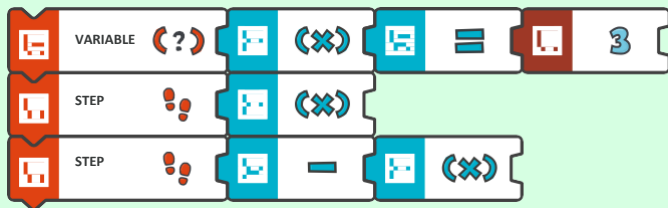
**}**

**END**

# Wiskundige bewerkingen op variabelen

Je kunt ook wiskundige bewerkingen op variabelen laten uitvoeren.

In onderstaand deel van het programma, veranderen we het aantal stappen naar de negatieve waarde van de variabele door de **MIN** tegel voor variabele **X** te zetten, als deze voor de tweede keer wordt aangeroepen.

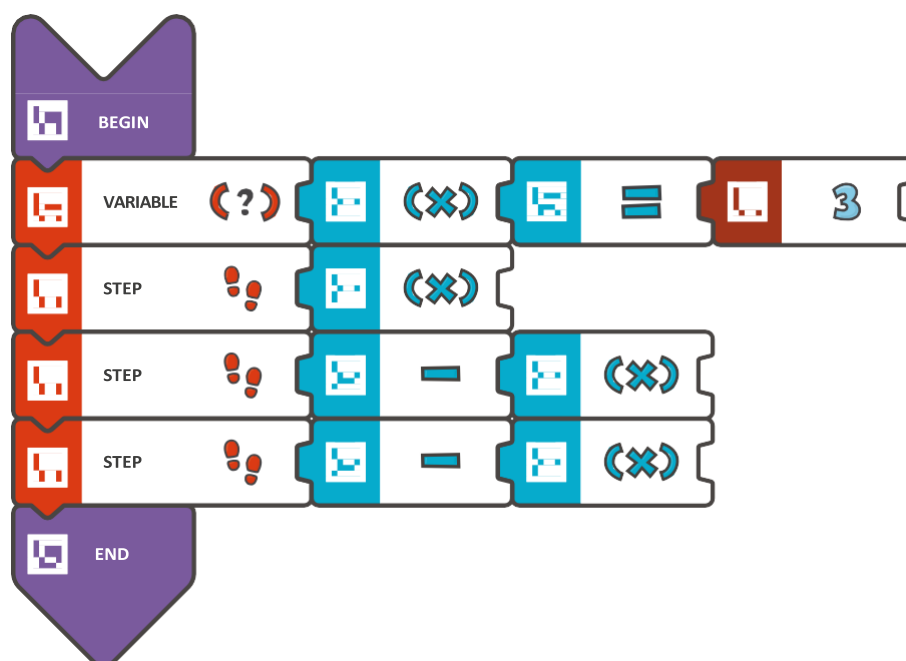


In deze opdracht is het de bedoeling dat de leerlingen de tegel met het getal 3 maar één keer gebruiken.

Opdracht (in deze opdracht is het de bedoeling dat de leerlingen de tegel met het getal 3 maar één keer gebruiken):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**VARIABLE X=3**

**STEP (X)**

**STEP (-X)**

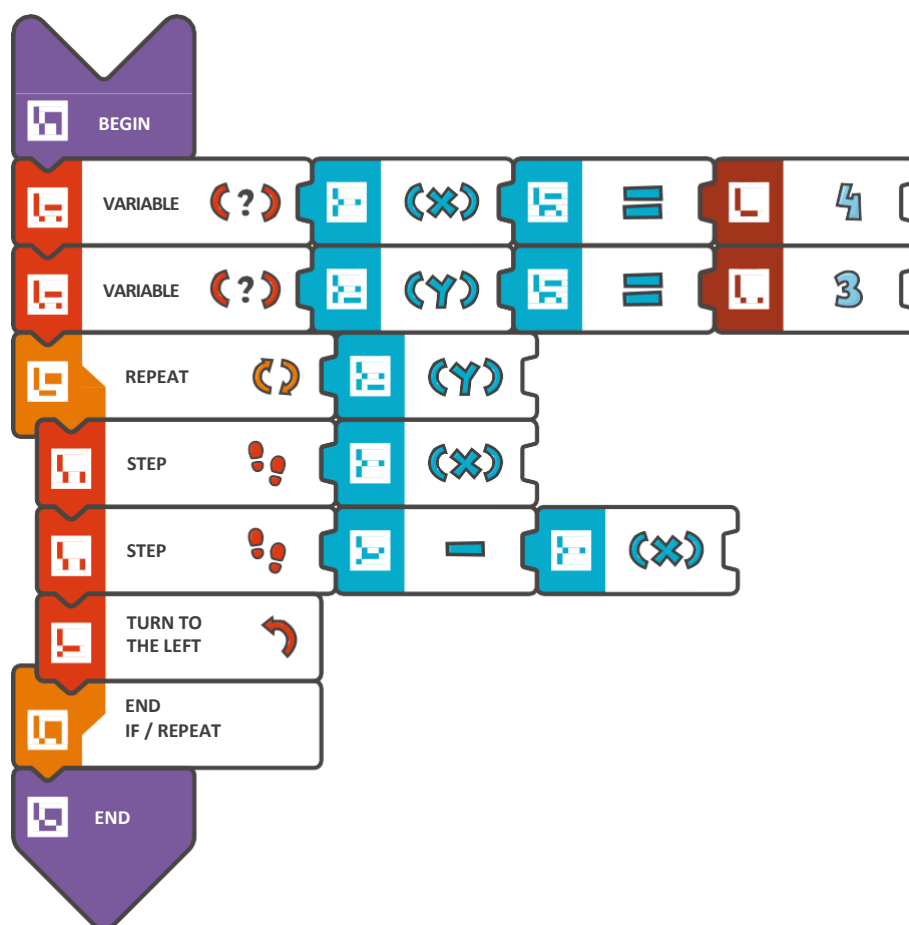
**STEP (-X)**

**END**

Opdracht: (in deze opdracht is het de bedoeling dat de leerlingen twee variabelen gebruiken: een voor het aantal stappen en de andere voor het aantal herhalingen van de lus):



Voorbeeldoplossing:

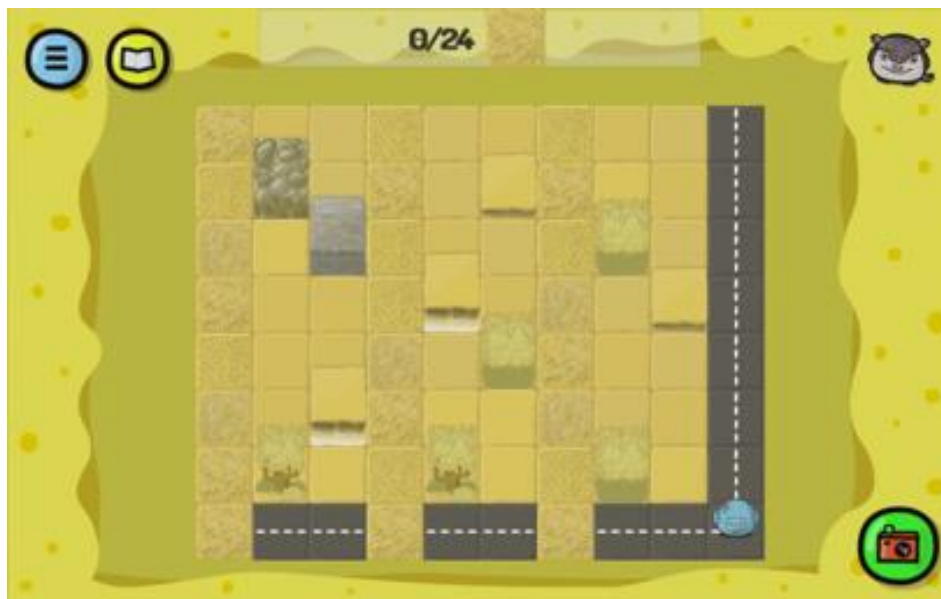




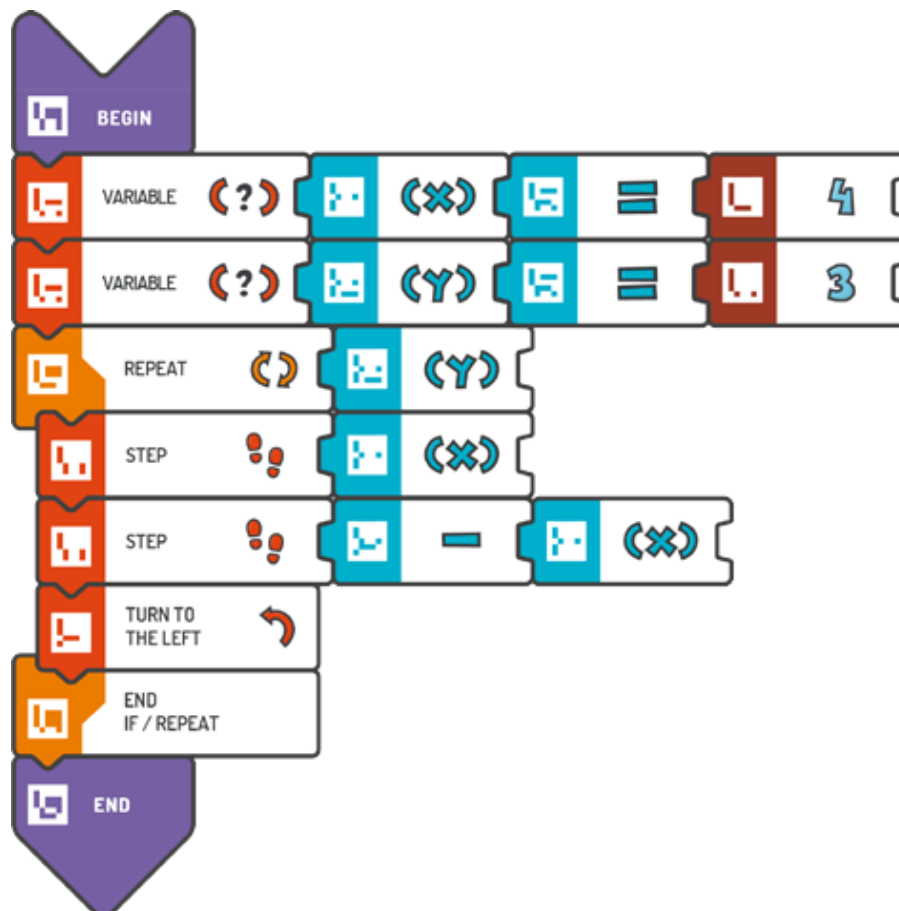
Voorbeeldoplossing (in tekstformaat):

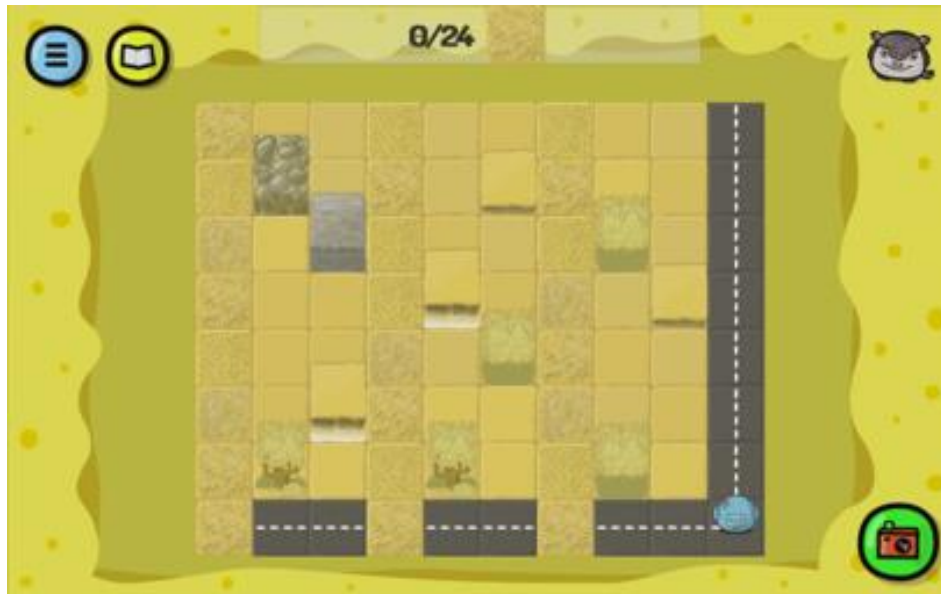
```
BEGIN
  VARIABLE X=4
  VARIABLE Y=3
  REPEAT (Y) {
    STEP (X)
    STEP (-X)
    TURN TO THE LEFT
  }
END
```

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**VARIABLE Y=3**

**VARIABLE X=7**

**REPEAT (Y) {**

**TURN TO THE LEFT**

**STEP (3)**

**TURN TO THE RIGHT**

**STEP (X)**

**STEP (-X)**

**}**

**END**

# Module VIII

## Australië 2



### Voorwaardelijke commando's - IF, ELSE

Plant de ontbrekende boompjes in de actievelden.

In deze module is het de bedoeling dat de leerlingen een voorwaardelijk commando gebruiken om de opdracht te voltooien.

# MODULE INTRODUCTIE: voorwaardelijke commando's IF, ELSE

We nemen allemaal, elke dag, een aantal beslissingen. Zal ik een jas of een trui aantrekken naar school? Hoeveel boterhammen zal ik als ontbijt eten? Zal ik met de fiets of de bus naar school gaan? Computer programma's controleren altijd iets, berekenen iets en nemen constant beslissingen. Wanneer je een programma schrijft en besluit om een voorwaardelijk commando te gebruiken, dan moet je alles zo duidelijk uitleggen aan de computer, dat het programma precies weet wat het moet doen.

## De IF, ELSE commando combinatie

**IF** en **ELSE** zijn twee commando's die vaak samen gebruikt worden en ze vormen samen een van de meest basis logische structuren, bij het nemen van beslissingen. Een voorwaardelijk commando is een krachtig stukje gereedschap en komt daarom in de meeste computer programma's voor.

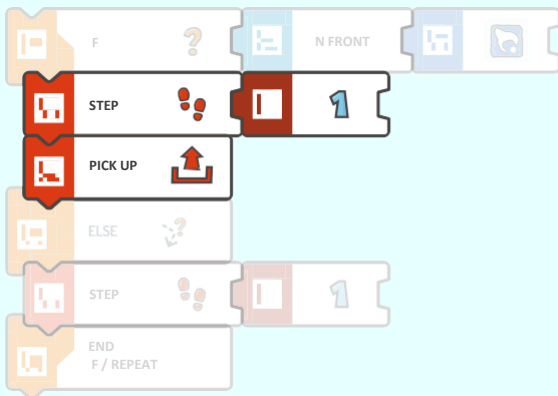


**Een typisch voorwaardelijk commando bestaat uit de volgende sleutelementen:**

1. Een voorwaarde waarvan gekeken moet worden of er wel of niet aan voldaan wordt,



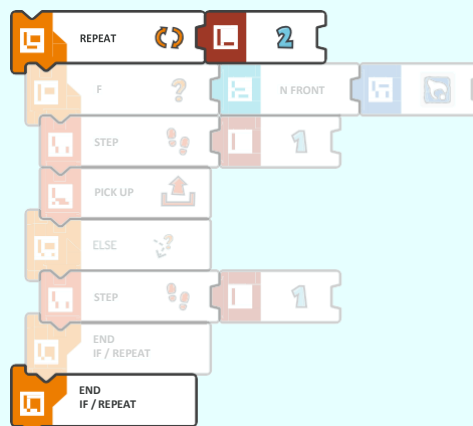
2. Commando's die uitgevoerd worden wanneer er aan de voorwaarde voldaan wordt.



3. Commando's die uitgevoerd worden wanneer er niet aan de voorwaarde voldaan wordt.

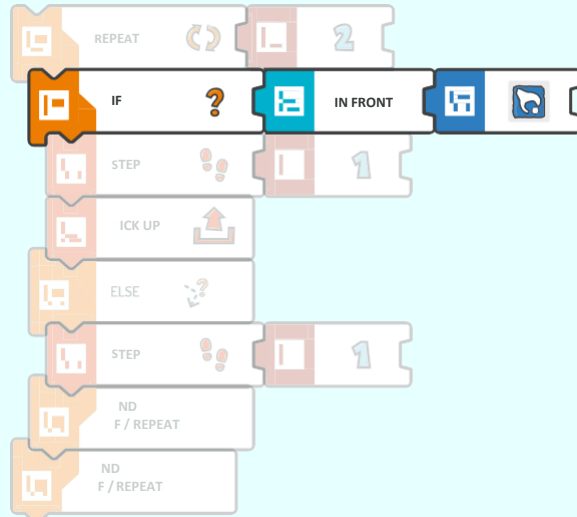


Laten we het voorbeeld hieronder bekijken. Om het begrip 'voorwaardelijk commando' makkelijker te laten begrijpen, zullen we het voorwaardelijke commando twee keer laten aanroepen door het in een **REPEAT (2)** lus te plaatsen.

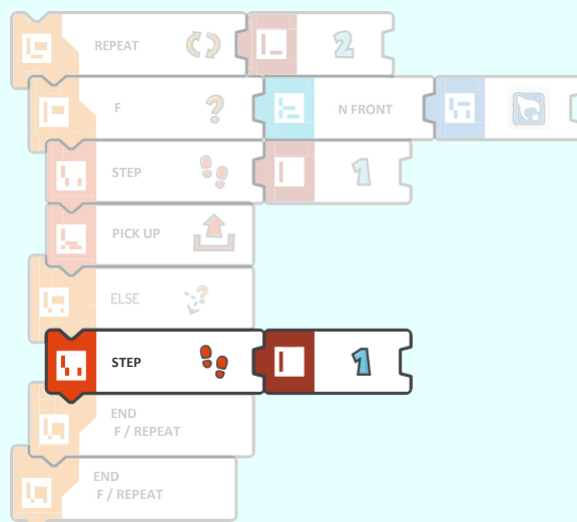


## Een voorwaardelijk commando in actie:

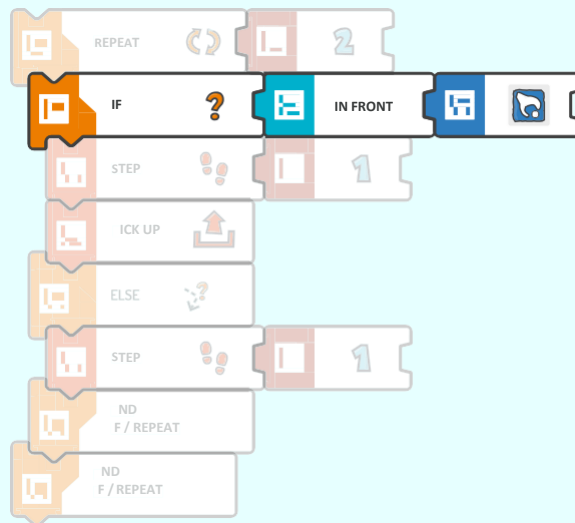
1. Scottie controleert de voorwaarde: "Is er een **COLLECTABLE OBJECT** voor me?" voor de eerste keer. In het voorbeeld hieronder is dit niet het geval.



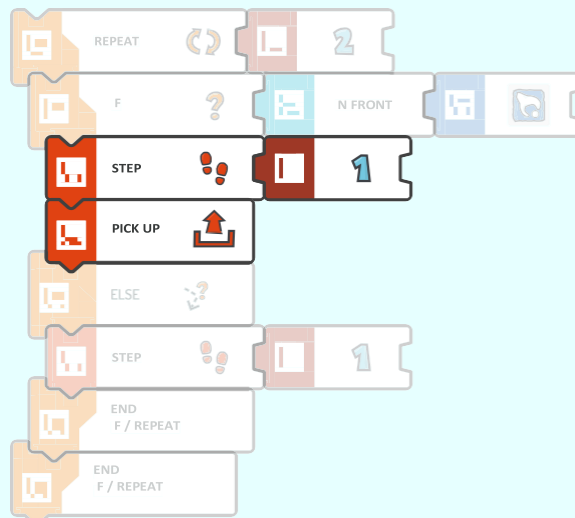
2. Als er niet aan de voorwaarde voldaan wordt, wordt het **ELSE** commando uitgevoerd. Hierdoor zet Scottie één stap - **STEP (1)**.



3. De **REPEAT (2)** lus vereist dat we de voorwaarde: "Is er een **COLLECTABLE OBJECT** voor me?" nog een keer controleren. Deze keer wordt er wel aan de voorwaarde voldaan.



4. Aangezien er aan de voorwaarde voldaan wordt, zal Scottie de commando's direct onder de **IF** regel gaan uitvoeren, namelijk **STEP (1)** en **PICK UP**.



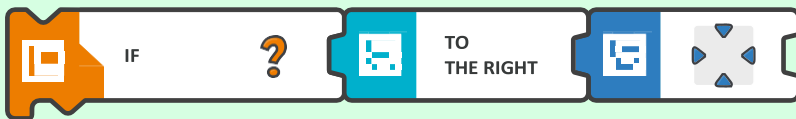
### LET OP:

Wanneer er aan de eerste voorwaarde van een voorwaardelijk commando voldaan wordt, worden de andere voorwaarden niet meer gecontroleerd of uitgevoerd.

# Actieveld

Een van Scottie's dierenvrienden heeft een prachtige boomgaard. In ruil voor de hulp van zijn vriend, beloofd Scottie om een paar bomen voor hem te planten.

Plant de ontbrekende bomen in de daarvoor bestemde actievelden en gebruik hiervoor de nieuwe tegels. In deze opdracht wordt gefocust op het voorwaardelijke commando **IF** in combinatie met de **ACTIEVELD** tegel. De voorwaarde die gecontroleerd wordt, is de locatie van een actieveld ten opzichte van Scottie, bijvoorbeeld:



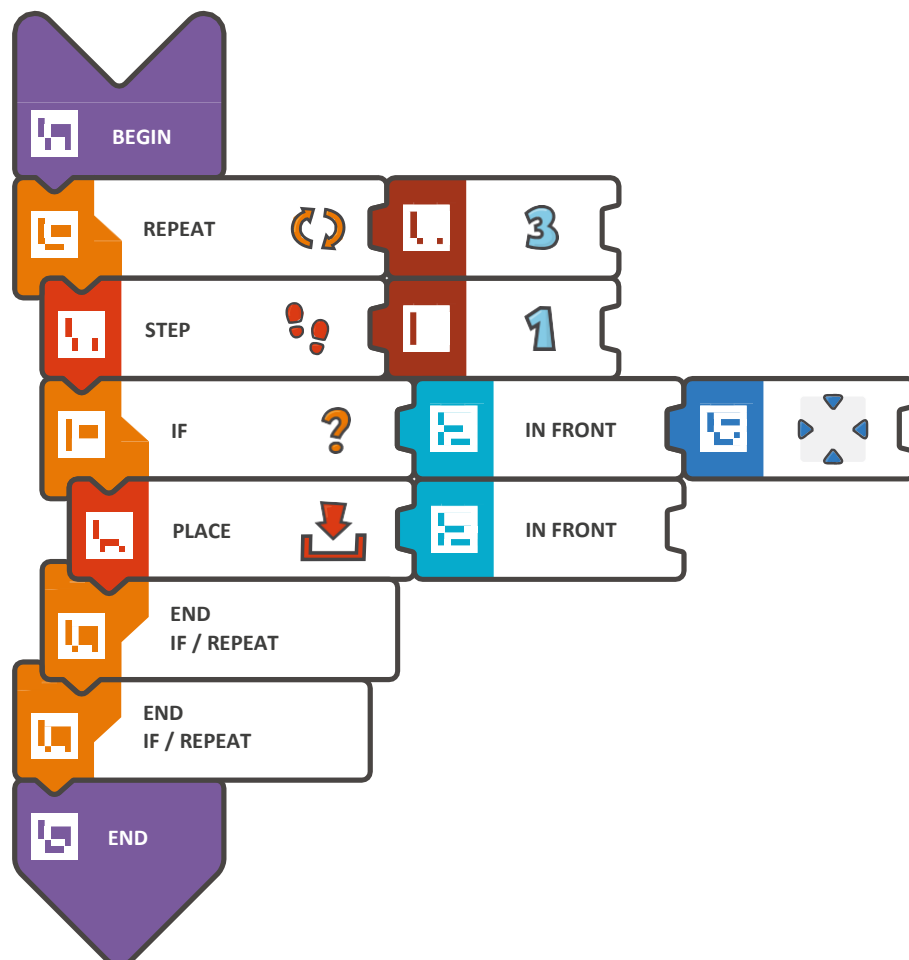
## ONTHOUD!

Je moet altijd een **END IF / REPEAT** tegel plaatsen, aan het einde van een voorwaardelijk commando.

Opdracht (denk er aan om een voorwaardelijk commando te gebruiken in deze opdracht):



Voorbeeldoplossing:





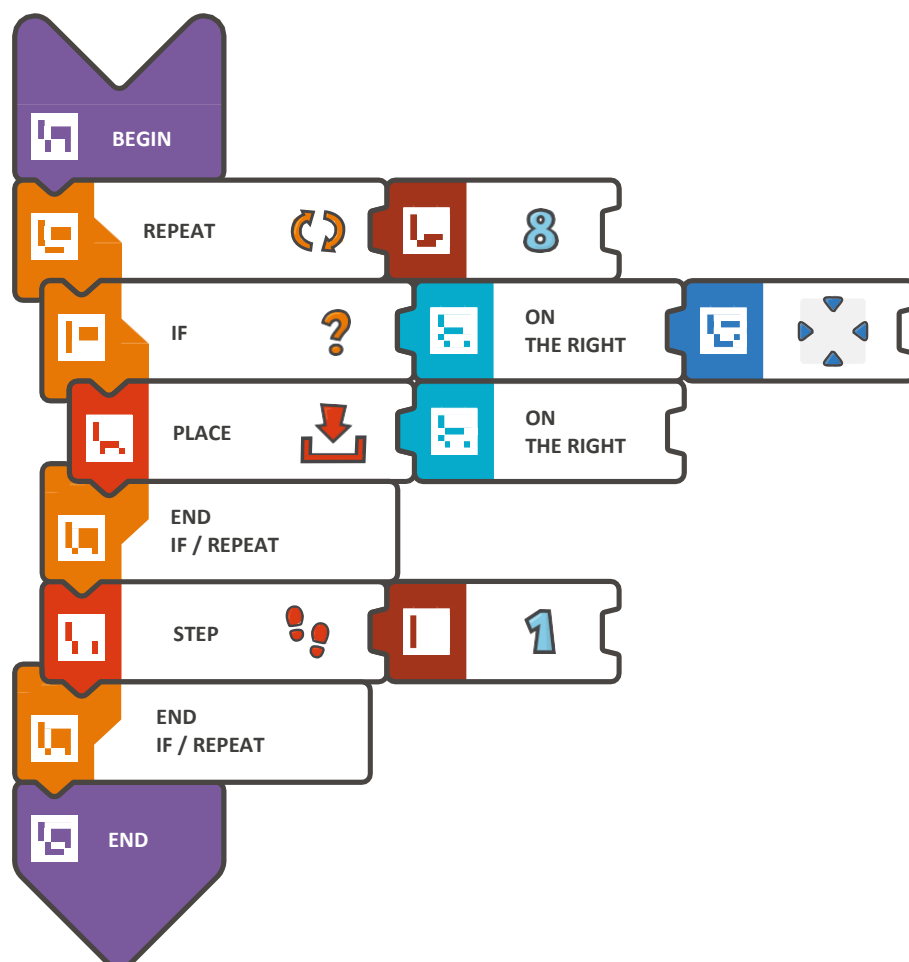
Voorbeeldoplossing (in tekstformaat):

```
BEGIN
  REPEAT (3) {
    STEP (1)
    IF (IN FRONT < ACTION SQUARE) {
      PLACE (IN FRONT)
    }
  }
END
```

Opdracht (denk er aan om een voorwaardelijk commando te gebruiken in deze opdracht):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (8) {**

**IF (ON THE RIGHT < ACTION SQUARE) {**

**PLACE (ON THE RIGHT)**

**}**

**STEP (1)**

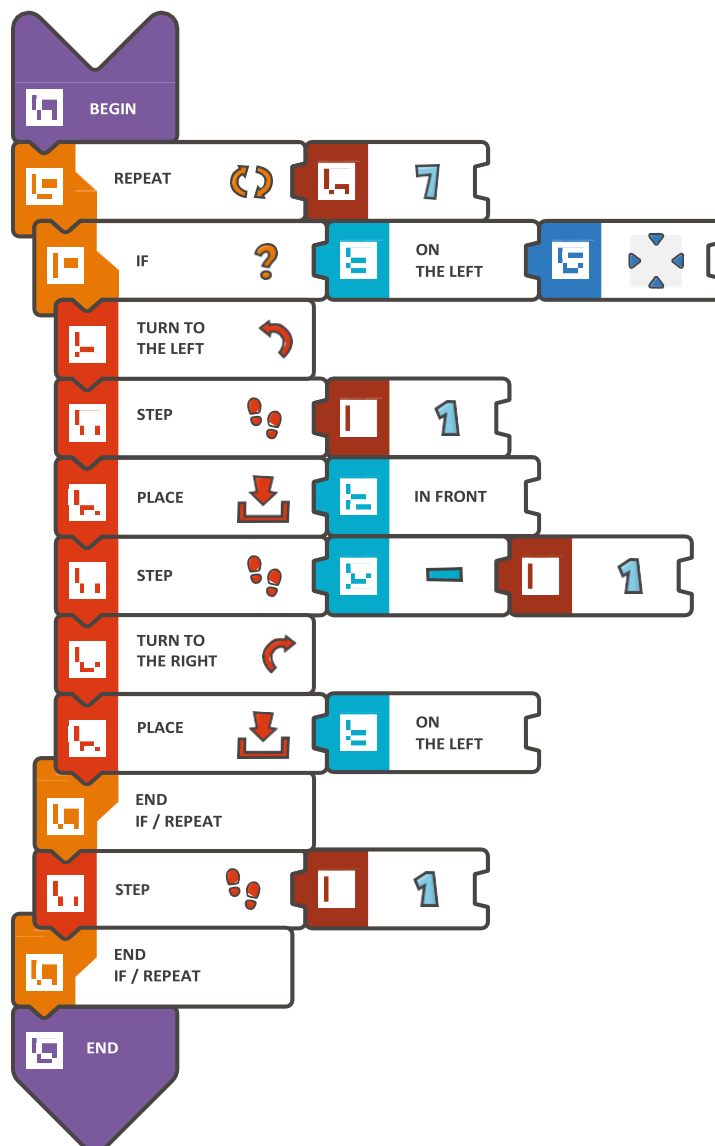
**}**

**END**

Opdracht (denk er aan om een voorwaardelijk commando te gebruiken in deze opdracht):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (7) {**

**IF (ON THE LEFT < ACTION SQUARE) {**

**TURN TO THE LEFT**

**STEP (1)**

**PLACE (IN FRONT)**

**STEP (-1)**

**TURN TO THE RIGHT**

**PLACE (ON THE LEFT)**

**}**

**STEP (1)**

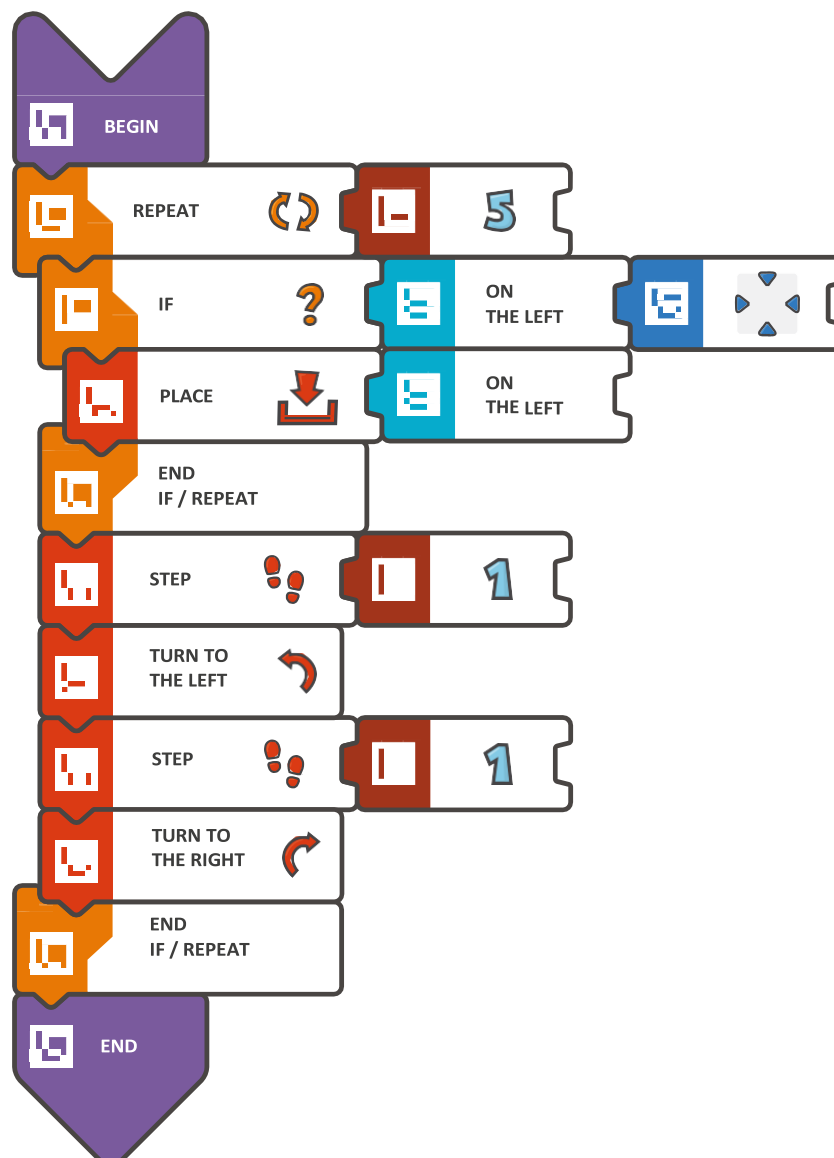
**}**

**END**

Opdracht: (denk er aan om een voorwaardelijk commando te gebruiken in deze opdracht):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (5) {**

**IF (ON THE LEFT < ACTION SQUARE) {**

**PLACE (ON THE LEFT)**

**}**

**STEP (1)**

**TURN TO THE LEFT**

**STEP (1)**

**TURN TO THE RIGHT**

**}**

**END**

# HINT

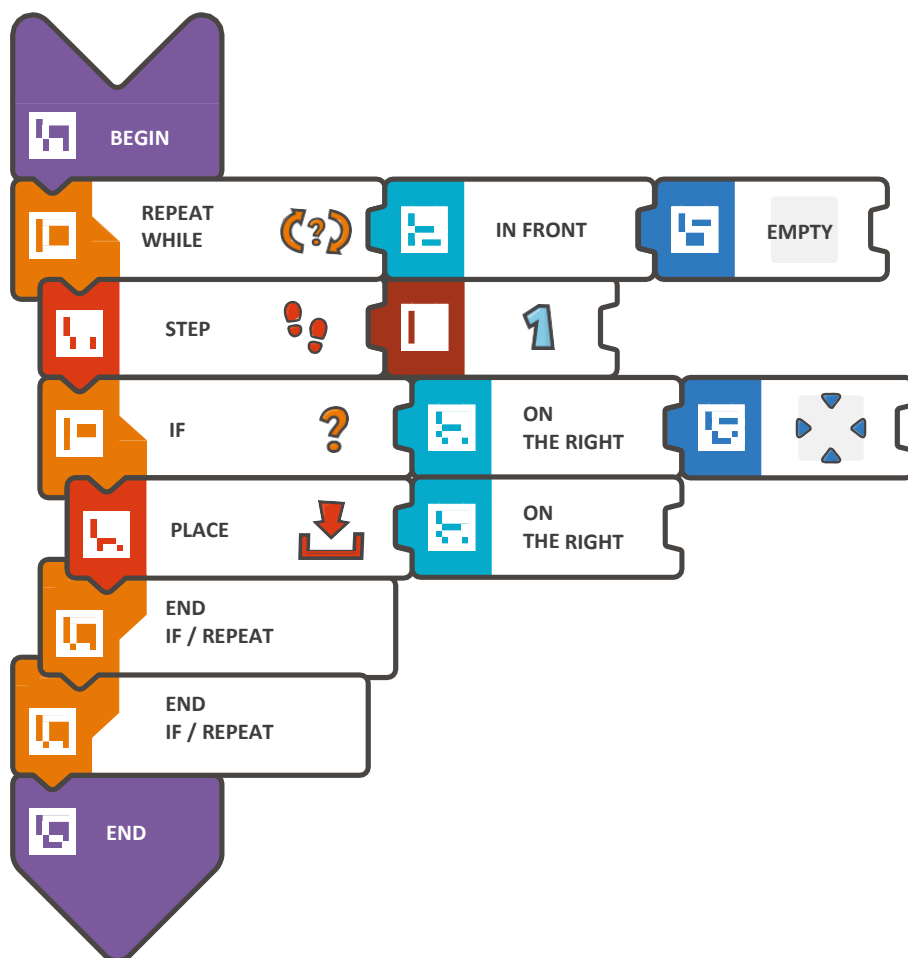
In deze opdracht moet je een combinatie maken van een voorwaardelijke lus met het **IF** commando, dat een andere voorwaarde zal toevoegen.

Wanneer er zich een obstakel op Scottie's weg bevindt, kun je de **REPEAT WHILE** lus gebruiken in plaats van de standaard **REPEAT** lus.

Opdracht: (er bevindt zich een obstakel aan het einde van het spelbord. Gebruik het in je voordeel en gebruik de **REPEAT WHILE** lus in plaats van de standaard **REPEAT** lus):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT WHILE (IN FRONT < EMPTY) {**

**STEP (1)**

**IF (ON THE RIGHT < ACTION SQUARE) {**

**PLACE (ON THE RIGHT)**

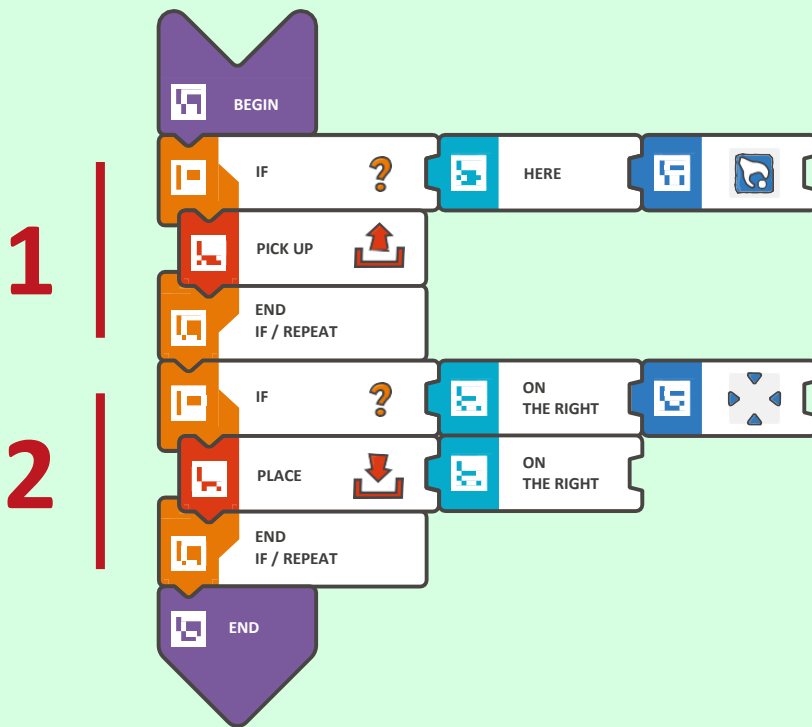
**}**

**}**

**END**

# Meerdere voorwaardelijke commando's gebruiken

Je kunt meer dan één voorwaardelijk commando gebruiken in een programma. Dit is een uitstekende manier om je logica te testen en te verbeteren. Het is ook een geweldige manier om te oefenen met dit sleutelonderdeel van het coderen in elke populaire programmeertaal. Elk voorwaardelijk commando kan een andere voorwaarde laten controleren:



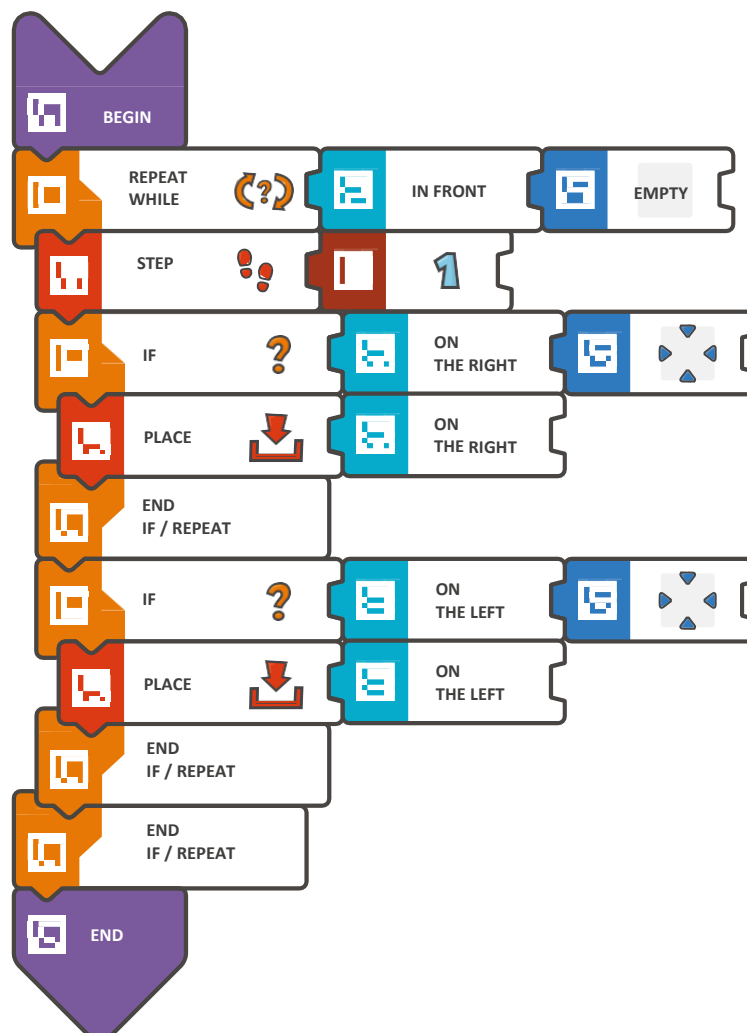
In het programma hierboven voert Scottie de volgende acties uit:

- Eerst controleert hij of er een verzamelobject ligt op het veld waar hij staat:
  - Indien dit het geval is, zal Scottie het object oppakken (en doorgaan met het controleren van de volgende voorwaarde),
  - Indien dit niet het geval is, zal Scottie het commando direct onder de **IF** regel overslaan (en doorgaan met het controleren van de volgende voorwaarde).
- Dan controleert Scottie of het veld aan zijn rechterkant een actieveld is:
  - Indien dit het geval is, zal Scottie het object uit zijn voorraad op dit veld plaatsen (en doorgaan met het controleren van de volgende voorwaarde)
  - Indien dit niet het geval is, zal Scottie het commando direct onder de **IF** regel overslaan (en in dit geval, het programma beëindigen met de **END** tegel).

Opdracht: (gebruik meer dan een voorwaardelijk commando in deze opdracht. Onthoud dat elk voorwaardelijk commando een andere voorwaarde kan controleren):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT WHILE (IN FRONT < EMPTY) {**

**STEP (1)**

**IF (ON THE RIGHT < ACTION SQUARE) {**

**PLACE (ON THE RIGHT)**

**}**

**IF (ON THE LEFT < ACTION SQUARE) {**

**PLACE (ON THE LEFT)**

**}**

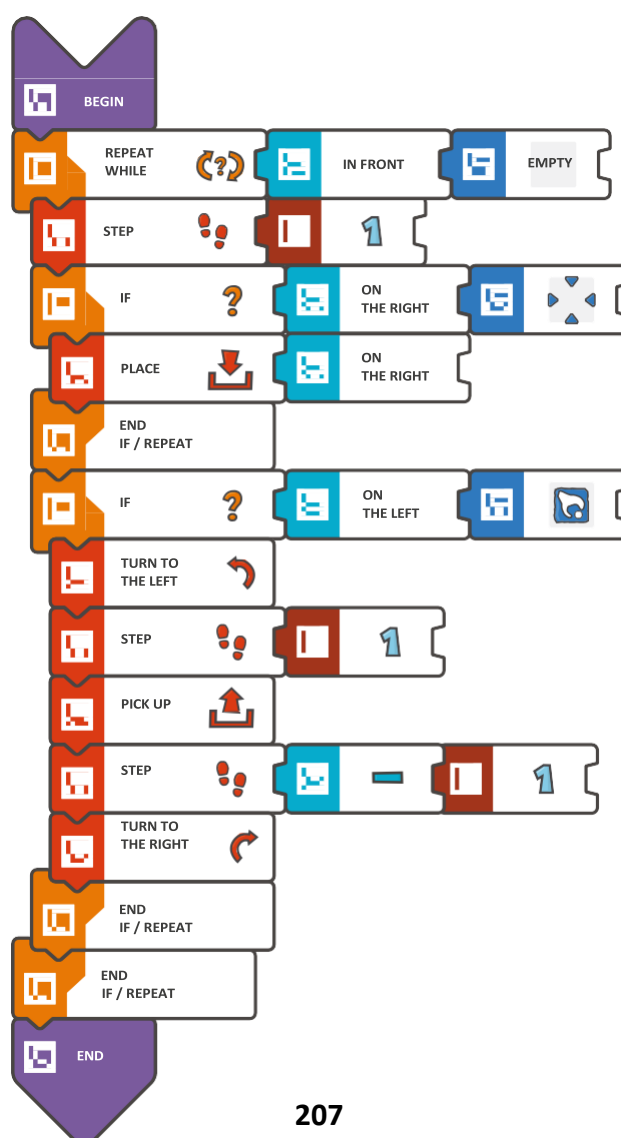
**}**

**END**

Opdracht: (Scottie heeft geen nieuwe boompjes meer in zijn voorraad, hij moet dus eerst een boompje oppakken voordat hij deze op een nieuwe plek kan planten):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT WHILE (IN FRONT < EMPTY) {**

**STEP (1)**

**IF (ON THE RIGHT < ACTION SQUARE) {**

**PLACE (ON THE RIGHT)**

**}**

**IF (ON THE LEFT < COLLECTABLE OBJECT) {**

**TURN TO THE LEFT**

**STEP (1)**

**PICK UP**

**STEP (-1)**

**TURN TO THE RIGHT**

**}**

**}**

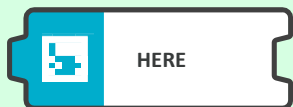
**END**

## HEUVEL en HERE tegels

Scottie zijn vriend woont in een ander deel van Australië, een rots- en heuvelachtig gebied. Laat Scottie zien hoe hij over de heuvels moet gaan. Gebruik de **HEUVEL** tegel



en de **HERE** tegel.



Gebruik de nieuwe tegels in combinatie met de voorwaardelijke commando's die je hiervoor geleerd hebt.

Opdracht: (gebruik een voorwaardelijk commando in deze module):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT WHILE (ON THE LEFT < ELEVATED SQUARE){**

**STEP (1)**

**IF (HERE < COLLECTABLE OBJECT) {**

**PICK UP**

**}**

**IF (ON THE RIGHT < OBSTACLE) {**

**STEP (1)**

**TURN TO THE RIGHT**

**STEP (1)**

**PLACE (IN FRONT)**

**STEP (-1)**

**TURN TO THE LEFT**

**}**

**}**

**END**

# Module IX

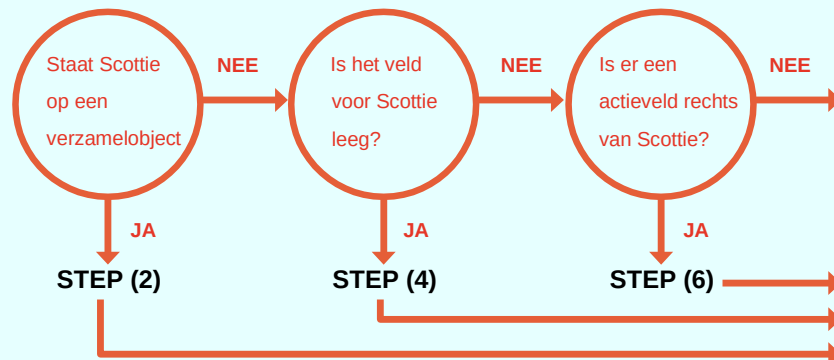
## Noord-Amerika 2



Lussen en voorwaardelijke commando's: IF, ELSE IF, ELSE. Scottie zijn ruimteschip is bijna klaar! Maar... de arbeiders hebben een verschrikkelijke puinhoop achtergelaten in de werkplaats. Help Scottie om de schroeven en het gereedschap dat verspreid ligt over de vloer, te verzamelen.

# MODULE INTRODUCTIE (A): ELSE IF

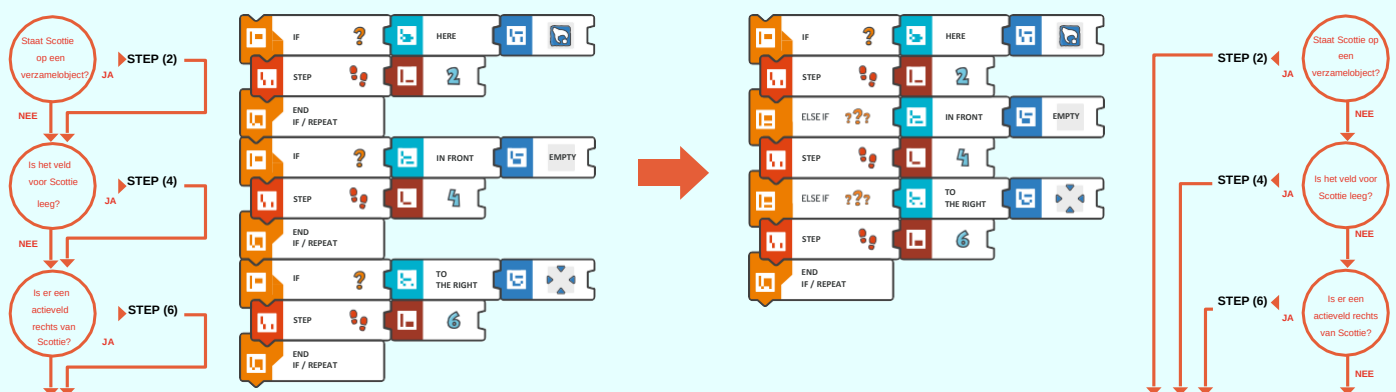
In **MODULE VIII** zagen we dat het vaak nodig is om meerdere voorwaardelijke commando's te gebruiken in een programma. Deze behoefte resulteerde in een nieuwe, zich ontwikkelende, codeerstructuur, waarbij slechts één uit een aantal voorwaarden zal worden uitgevoerd. Laten we kijken naar een voorbeeld met meerdere voorwaarden:



In dergelijke gevallen moet het voorwaardelijke commando **ELSE IF** gebruiken.



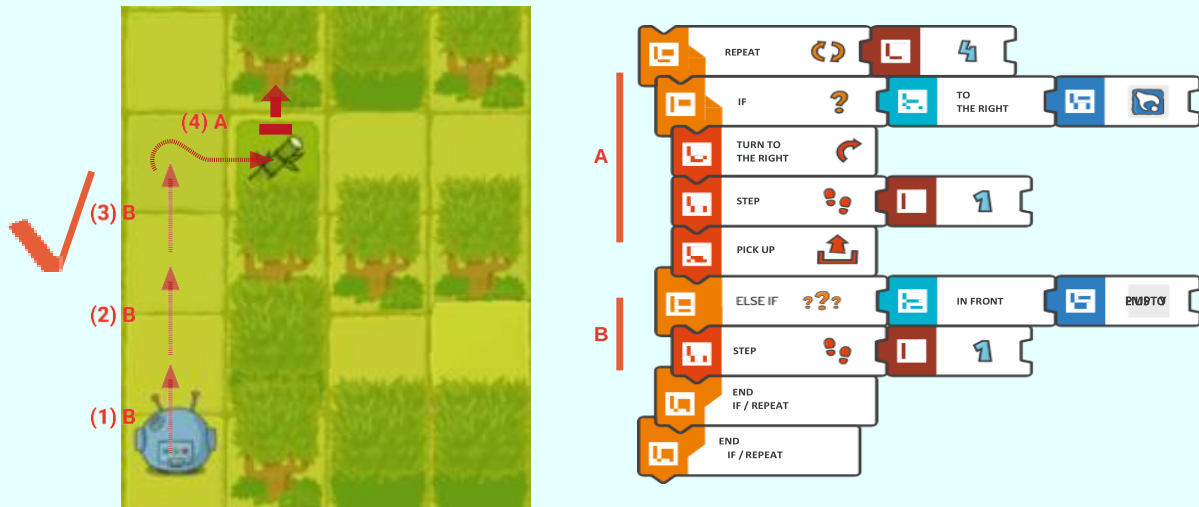
Door de **ELSE IF** tegel te gebruiken, kun je de code versimpelen door een voorwaardelijk commando uit te breiden, in plaats van het combineren van meerdere voorwaardelijke commando's.



Als we de **ELSE IF** tegel gebruiken, is de volgorde van de voorwaardes van cruciaal belang. Wanneer aan de eerste voorwaarde voldaan wordt (als deze waar is), zullen de tweede en derde voorwaarde niet gecontroleerd of uitgevoerd worden. Wordt er niet aan de eerste voorwaarde voldaan, dan gaat het programma verder met het controleren van de tweede voorwaarde en als hier aan voldaan wordt (als deze waar is), wordt de derde voorwaarde niet gecontroleerd of uitgevoerd.

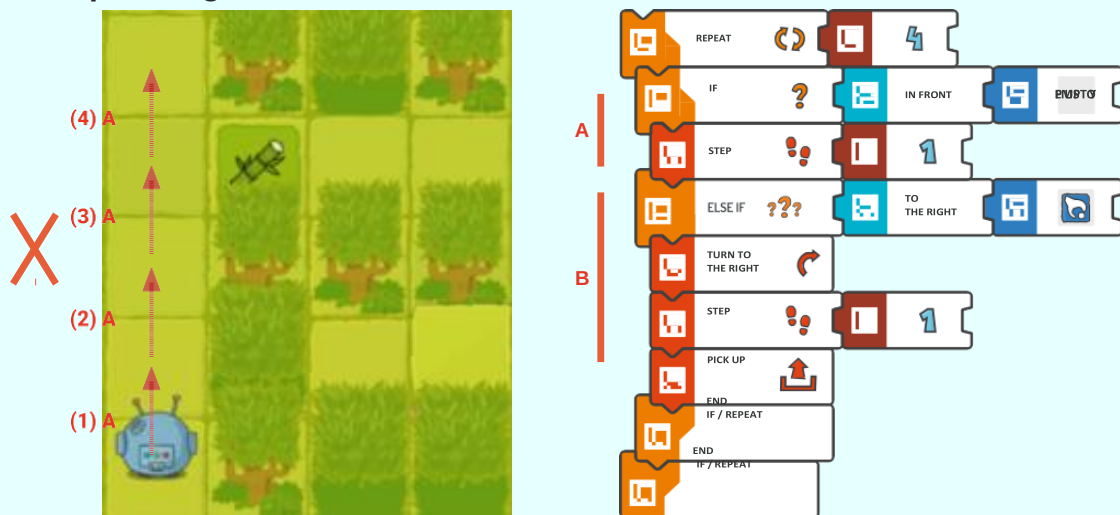
**Voorbeeld I** - het **ELSE IF** commando gebruiken. Scottie moet de bamboe oppakken:

**Goede oplossing:**



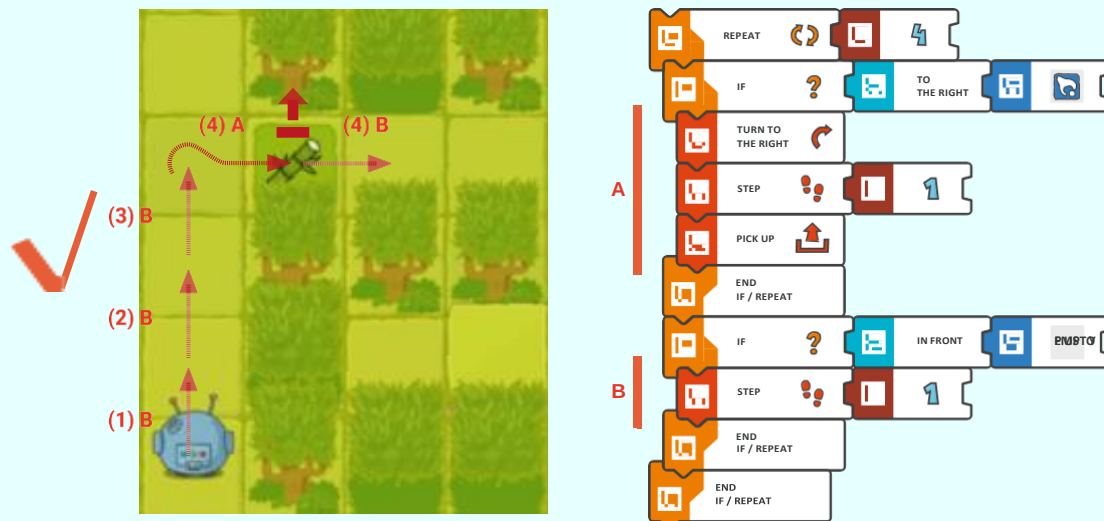
Het verschil tussen het goede en het foute antwoord is alleen de volgorde van de te controleren voorwaarden (en de commando's die bij deze voorwaarden horen). Merk op, dat elke herhaling van de loop slechts de commando's uitvoert, die bij één voorwaarde horen.

**Foute oplossing:**



**Voorbeeld II** - twee **IF** commando's gebruiken. Scottie moet de bamboe oppakken:

**Goede oplossing:**



De goede oplossing zonder de **ELSE IF** tegel te gebruiken is een regel langer dan hiervoor. Merk op, dat in de vierde herhaling van de loop aan beide voorwaarden voldaan wordt (ze zijn waar). De volgorde van de voorwaarden blijft uiteraard van essentieel belang in dergelijke situaties.

**Foute oplossing:**



# MODULE INTRODUCTIE (B): de ELSE tegel

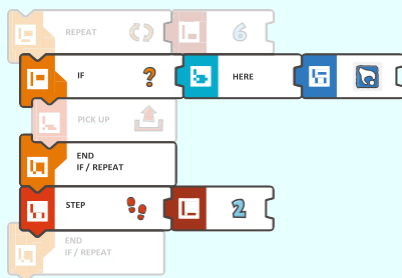
Naast het werken met meerdere voorwaardelijke commando's, zal Scottie soms alleen iets moeten doen, als er niet aan de voorgaande voorwaarden is voldaan. Hoe codeer je bijvoorbeeld 2 alternatieve acties en verder niets?



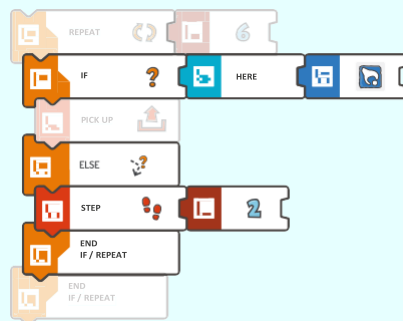
Om de hierboven aangegeven structuur in code te bouwen, moeten we het **ELSE** commando gebruiken.



Dit is een effectieve manier om Scottie zijn "standaard" commando's te schrijven. Bijvoorbeeld commando's die hij, elke keer dat aan een eerdere voorwaarde niet voldaan is, moet uitvoeren. Als sommige commando's buiten het voorwaardelijke commando zijn geplaatst, zullen ze altijd uitgevoerd worden. Dit kun je met name goed zien in een lus.



Met elke herhaling van de lus, zal Scottie **ALTIJD** twee stappen zetten.

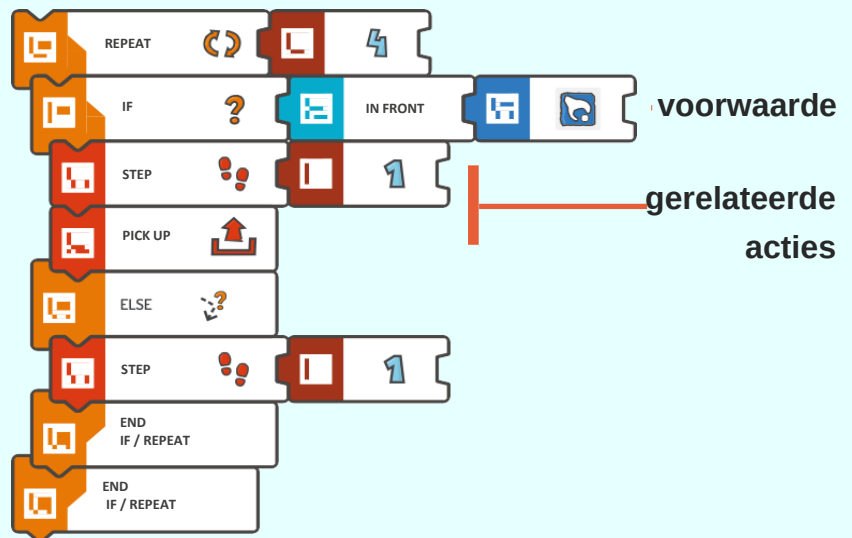


Met elke herhaling van de lus, zal Scottie twee stappen zetten, maar **ALLEEN** als hij niet op een verzamelobject staat.

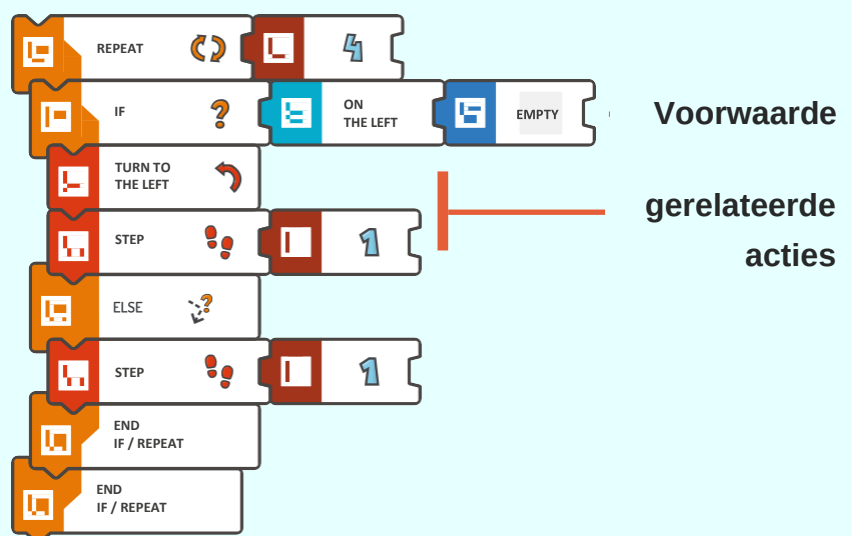
In een typische programmeertaal (**niet in Scottie Go!**) kun je ook het woord **NO** gebruiken om vergelijkbare voorwaardelijke commando's te maken.

Voorbeelden van het gebruik van het **ELSE** commando:

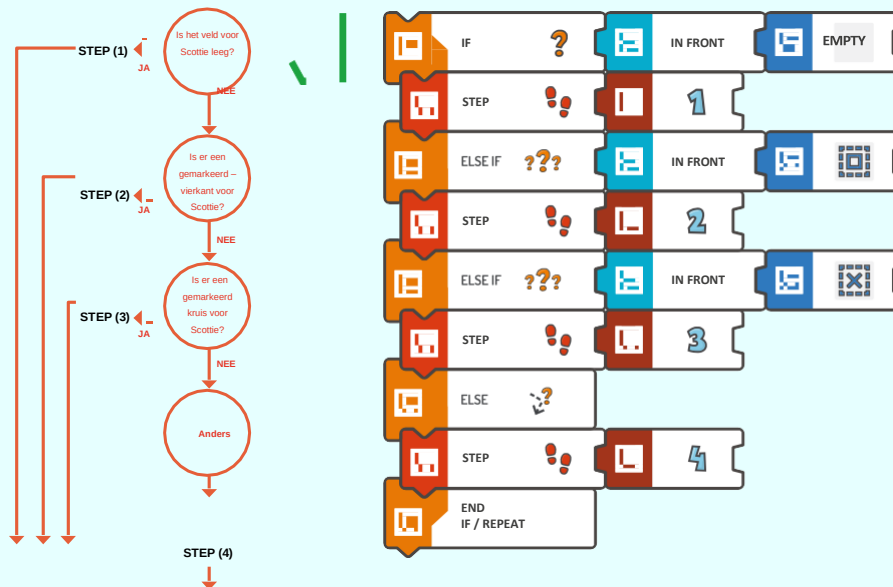
Laten we een programma schrijven waarin Scottie vooruit gaat, totdat hij iets tegenkomt waardoor hij een actie moet uitvoeren. Bijvoorbeeld, dat Scottie moet controleren of er een verzamelobject voor hem ligt, of dat hij naar links moet draaien als er links van hem een leeg veld is. Verder moet het programma makkelijk aan te passen zijn aan nieuwe wensen.



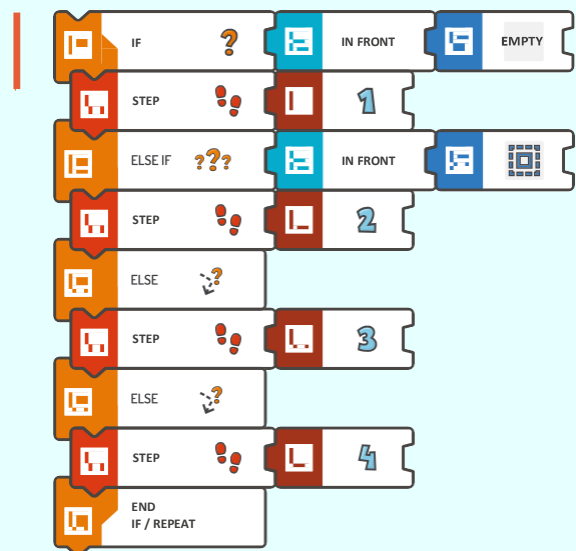
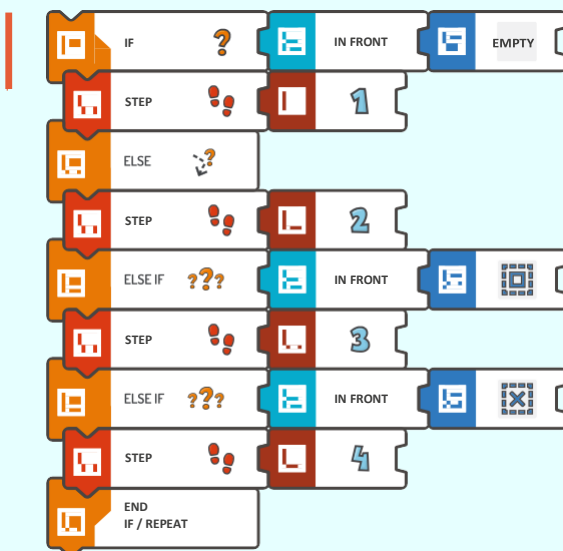
Je hoeft geen voorwaarde op te geven om Scottie vooruit te laten lopen. Het is voldoende om aan te geven in welke situatie Scottie moet reageren en wat zijn reactie moet zijn.



We kunnen het: **ELSE IF** en het **ELSE** commando beide gebruiken in een programma,



maar onthoud, dat je **ELSE** maar één keer kunt gebruiken en dat het aan het einde van het programma geplaatst moet worden. Onderstaande voorbeelden zijn **niet goed**.



# MODULE INTRODUCTIE (C): [optioneel] - REPEAT (FOREVER)

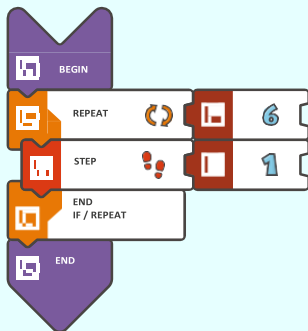
In programma's die voorwaardelijke commando's bevatten zonder een **REPEAT** lus, zal het voorwaardelijke commando slechts één keer gecontroleerd worden en daarna zal het programma stoppen. We willen echter dat de voorwaarde meerdere keren gecontroleerd wordt, net zo lang tot Scottie al zijn onderdelen en gereedschappen gevonden en verzameld heeft.

Hiervoor kunnen we de nieuwe tegel: **FOREVER** gebruiken.

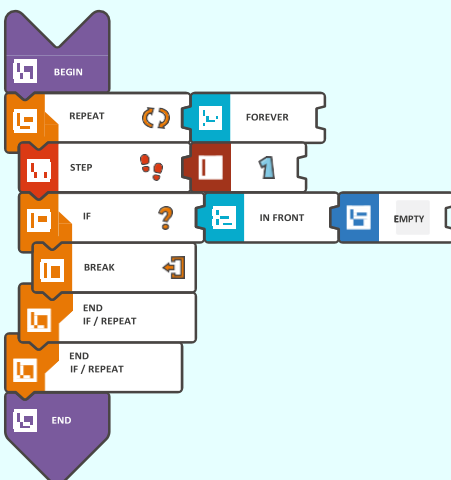


In de opdrachten van Module IX moet de **REPEAT** lus gebruikt worden, op een van de twee manieren zoals hieronder beschreven, afhankelijk van de leeftijd en ervaring van de leerlingen:

**1. REPEAT** (een bepaald aantal keer) – het commando dat de leerlingen al goed kennen van Module III. De uitdaging is, om aan te geven hoe vaak het commando in de lus uitgevoerd moet worden. In het voorbeeld hieronder, zal Scottie 6 keer 1 stap zetten, ongeacht wat er in zijn omgeving staat.

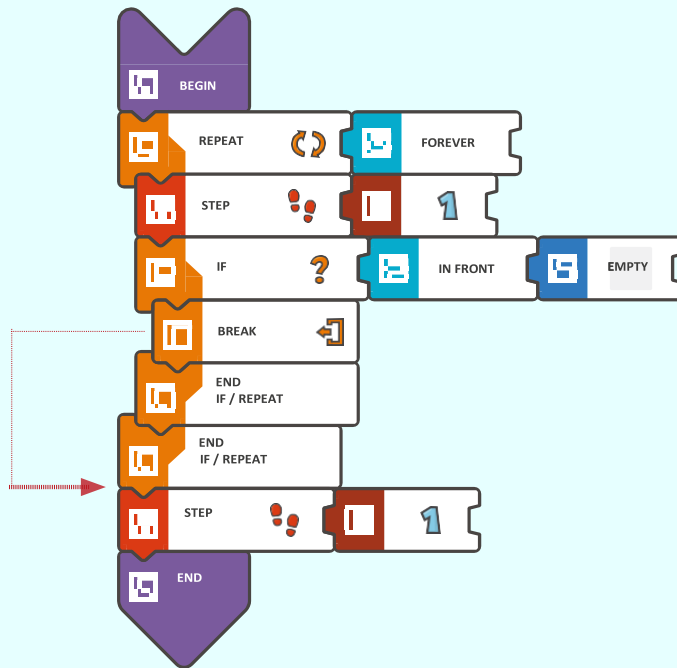


**2. REPEAT (FOREVER)** met het **BREAK** commando.



Het **REPEAT (FOREVER)** commando is niet in een les uitgelegd, maar het is interessanter en het is de meest gebruikte oplossing. Eenzelfde code zal in detail geanalyseerd worden op de volgende pagina.

## REPEAT (FOREVER) met het **BREAK** commando.



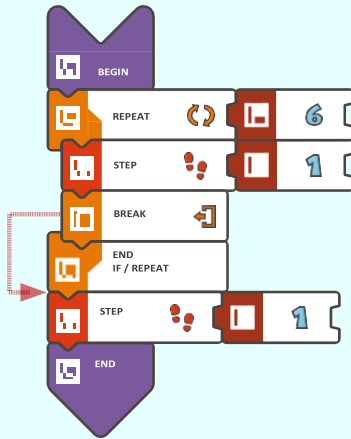
In bovenstaande code zal het **STEP 1** commando herhaald worden, tot het **BREAK** commando aangeroepen wordt (dat we normaal gesproken in een voorwaardelijk commando plaatsen, bijvoorbeeld gerelateerd aan een object op het spelbord). In ons voorbeeld zal Scottie stoppen met het uitvoeren van de lus, als er een leeg veld voor hem is.

Het **BREAK** commando, op een strategische locatie geplaatst, is van essentieel belang om de code goed te laten functioneren. De commando's die binnen de lus, maar na het **BREAK** commando staan, zullen niet uitgevoerd worden.

In deze module kun je de volgende dingen tegenkomen: **GEMARKEERD VIERKANT**, **GEMARKEERD KRUIS**, **VERZAMELOBJECT**, **ACTIEVELD** en een **OBSTAKEL**. Deze kunnen gebruikt worden als referentiepunt voor het maken van goede voorwaardelijke commando's.

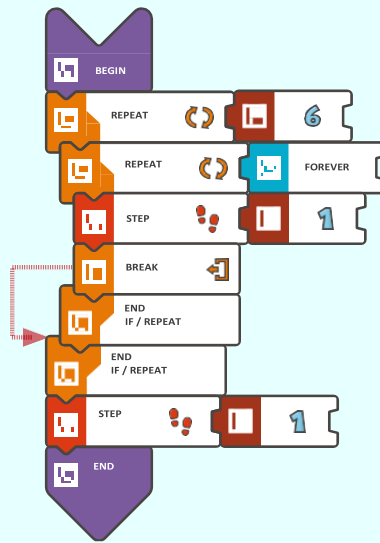


Wanneer het **BREAK** commando uitgevoerd wordt, zal het programma uit de lus springen naar het **BREAK** commando in staat.



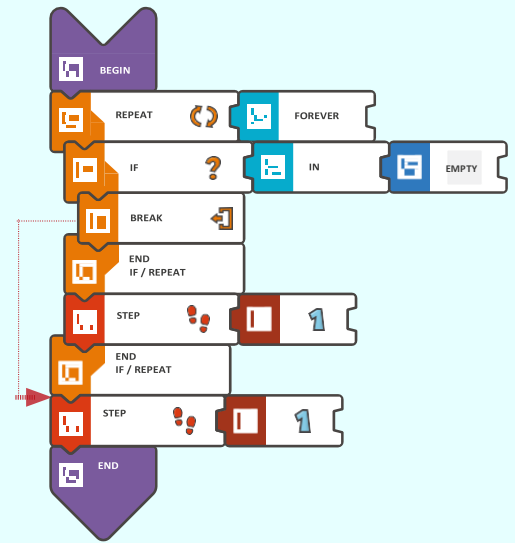
**Scottie zet 2 stappen.**

(1 in de lus;  
1 buiten de lus)



**Scottie zet 7 stappen.**

(1 in de lus voor  
elke herhaling van  
de **FOREVER** lus  
- in totaal 6  
herhalingen -;  
1 buiten de lus)



**Scottie blijft  
voortgaan tot  
hij stopt bij een  
leeg veld.**

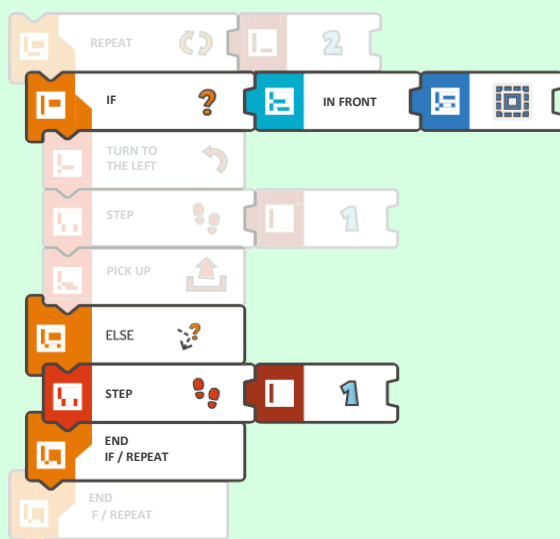
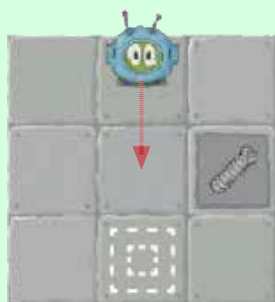
(Als het veld voor  
Scottie niet leeg  
is, zal hij altijd 1  
stap zetten;  
Wanneer Scottie  
een leeg veld  
tegenkomt, gaat  
het programma uit  
de lus en zet  
Scottie nog één  
stap voor het **END**  
commando)

# ELSE

De opdrachten in Module IX vereisen dat je een **REPEAT** commando in combinatie met voorwaardelijke commando's gebruikt. In deze opdracht moet je het **ELSE** commando gebruiken.



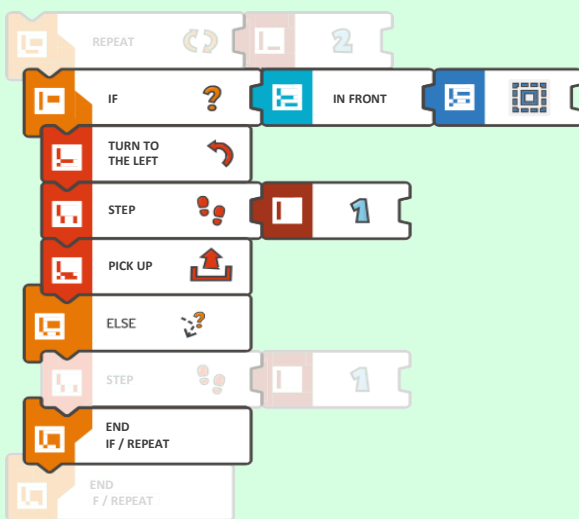
Met de **ELSE** tegel kun je commando's maken, die Scottie alleen uitvoert als niet aan de voorgaande voorwaardes voldaan wordt. Laten we een voorbeeld van een programma bekijken, waarin Scottie acties uitvoert, gebaseerd op de relatie tussen waar hij staat en waar **GEMARKEERDE VIERKANTEN** op het spelbord staan.



## 1<sup>e</sup> HERHALING VAN DE LUS:

Er is geen **GEMARKEERD VIERKANT** voor Scottie ->

Scottie voert de commando's tussen de **ELSE** en de **END IF / REPEAT** tegel uit.



## 2<sup>e</sup> HERHALING VAN DE LUS:

Er is een **GEMARKEERD VIERKANT** voor Scottie ->

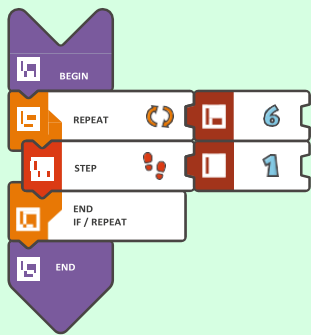
Scottie voert de commando's tussen de **IF** en de **ELSE** tegel uit.

**LET OP!!**

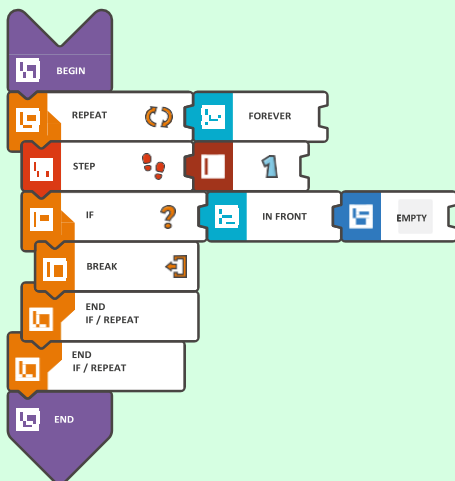
In deze en de hier opvolgende opdrachten kun je de **REPEAT (FOREVER)** lus gebruiken.

In de opdrachten van Module IX moet de **REPEAT** lus gebruikt worden, op een van de twee manieren zoals hieronder beschreven, afhankelijk van de leeftijd en ervaring van de leerlingen:

**1. REPEAT** (een bepaald aantal keer) – het commando dat de leerlingen al goed kennen van Module III. De uitdaging is, om aan te geven hoe vaak het commando in de lus uitgevoerd moet worden. In het voorbeeld hieronder, zal Scottie 6 keer 1 stap zetten, ongeacht wat er in zijn omgeving staat.



**2. REPEAT (FOREVER)** met het **BREAK** commando.

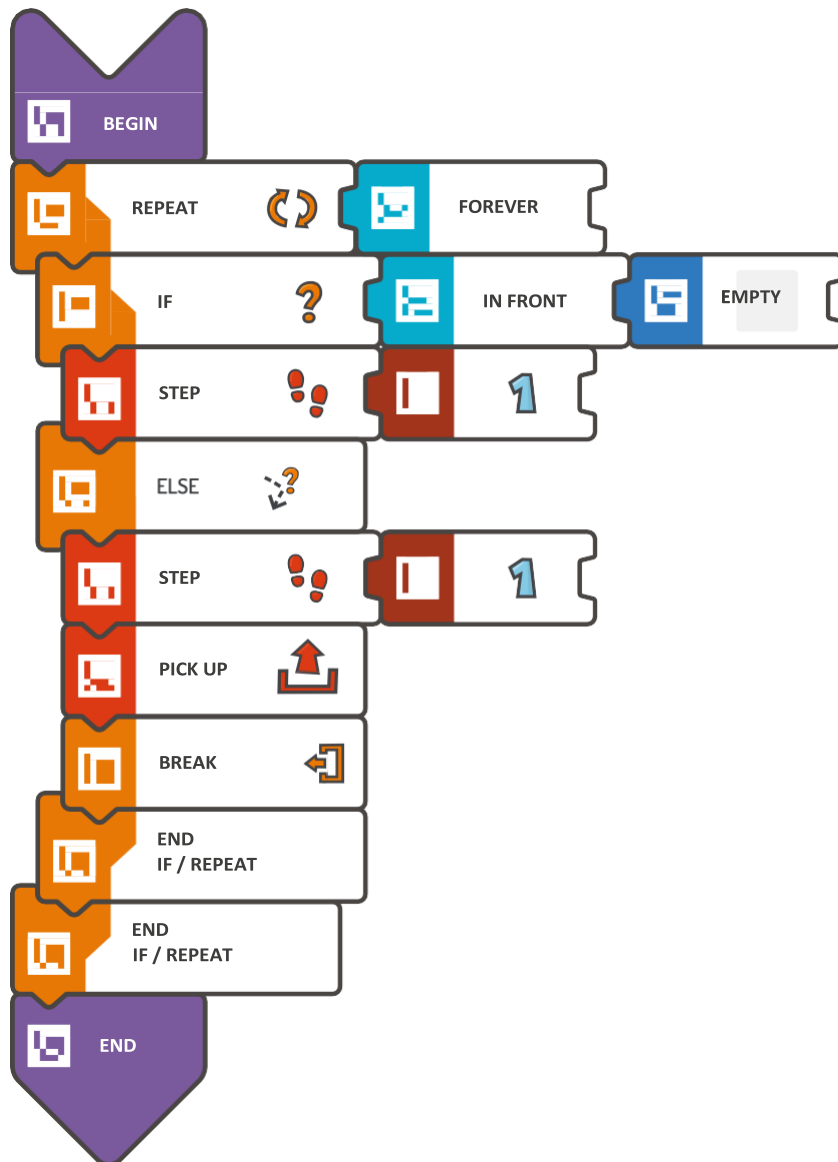


Het **REPEAT (FOREVER)** commando is niet in een les uitgelegd, maar het is interessanter en het is de meest gebruikte oplossing. In de Module introductie (C) kun je meer lezen over het **REPEAT (FOREVER)** commando.

Opdracht: (gebruik de ELSE tegel in deze opdracht):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

```
BEGIN
  REPEAT (FOREVER) {
    IF (IN FRONT < EMPTY) {
      STEP (1)
    } ELSE {
      STEP (1)
      PICK UP
      BREAK
    }
  }
END
```

# ELSE IF en MARKERINGEN

De opdrachten in Module IX vereisen dat je een **REPEAT** lus gebruikt in combinatie met voorwaardelijke commando's. In deze opdracht moet je het **ELSE IF** commando gebruiken.



Met de **ELSE IF** tegel kun je **aanvullende voorwaarden** bepalen die Scottie zal controleren als er niet aan de eerste voorwaarde voldaan wordt. Scottie zal de voorwaarden een voor een controleren totdat er aan een van de voorwaarden voldaan wordt.

Bijvoorbeeld:



**1<sup>e</sup> HERHALING VAN DE LUS:**

Er is een **GEMARKEERD VIERKANT** voor Scottie ->

Scottie voert uit:

**STEP (1)**

**TURN TO THE RIGHT**



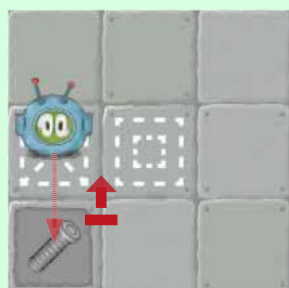
**2<sup>e</sup> HERHALING VAN DE LUS:**

Er is een **GEMARKEERD KRUIS** voor Scottie ->

Scottie voert uit:

**STEP (1)**

**TURN TO THE LEFT**



**3<sup>e</sup> HERHALING VAN DE LUS:**

Er is geen **GEMARKEERD VIERKANT** voor Scottie ->

Scottie voert uit:

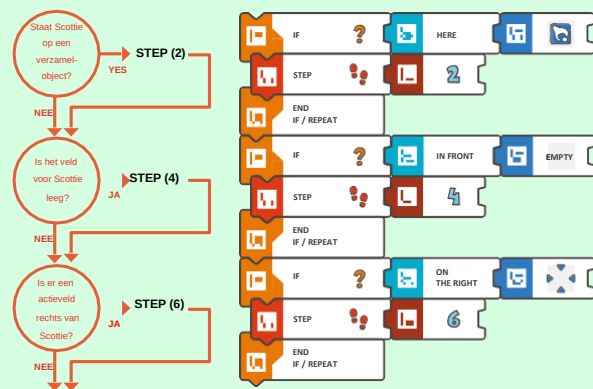
**STEP (1)**

**PICK UP**

Met de **ELSE IF** tegel kun je verschillende commandosets maken, waarvan er slechts één wordt uitgevoerd.

### Zonder

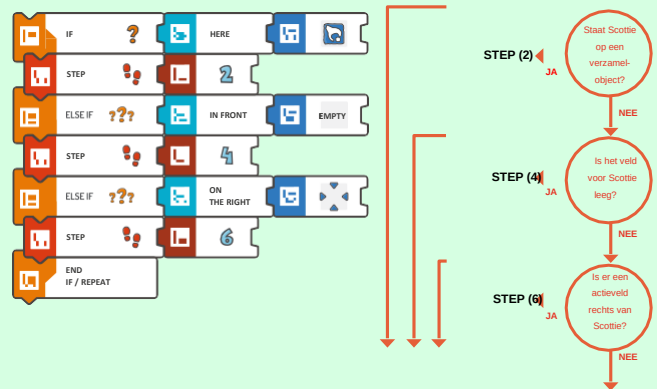
### ELSE IF



Scottie controleert de voorwaardes een voor een. Zo kan Scottie wel  $2+4+6=12$  stappen moeten zetten!

### Met

### ELSE IF



Scottie controleert de voorwaardes een voor een. Wanneer aan een van de voorwaardes voldaan wordt, worden de andere voorwaardes niet meer doorlopen. Zo zal Scottie slechts 2, 4, of 6 stappen hoeven zetten.

### ONTHOUD

In deze en volgende opdrachten van deze module kun je de **REPEAT (FOREVER)** tegel gebruiken.

Het **REPEAT (FOREVER)** commando is nog niet uitgelegd, maar het is een interessantere en universele oplossing.

Naast de reeds bekende objecten zoals verzamelobjecten, actievelden en obstakels, krijgt Scottie in deze module te maken met **GEMARKEERDE VIERKANTEN** en **GEMARKEERDE KRUISEN** op de vloer van de werkplaats.



De medewerkers van de werkplaats gebruiken de gemarkeerde velden om de werkplaats georganiseerd te houden. Je kunt ze gebruiken als referentiepunten in de code voor Scottie, met de voorwaardelijke commando's **IF** en **ELSE IF**.

Gebruik de tegels in de voorwaardelijke commando's van deze opdracht.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

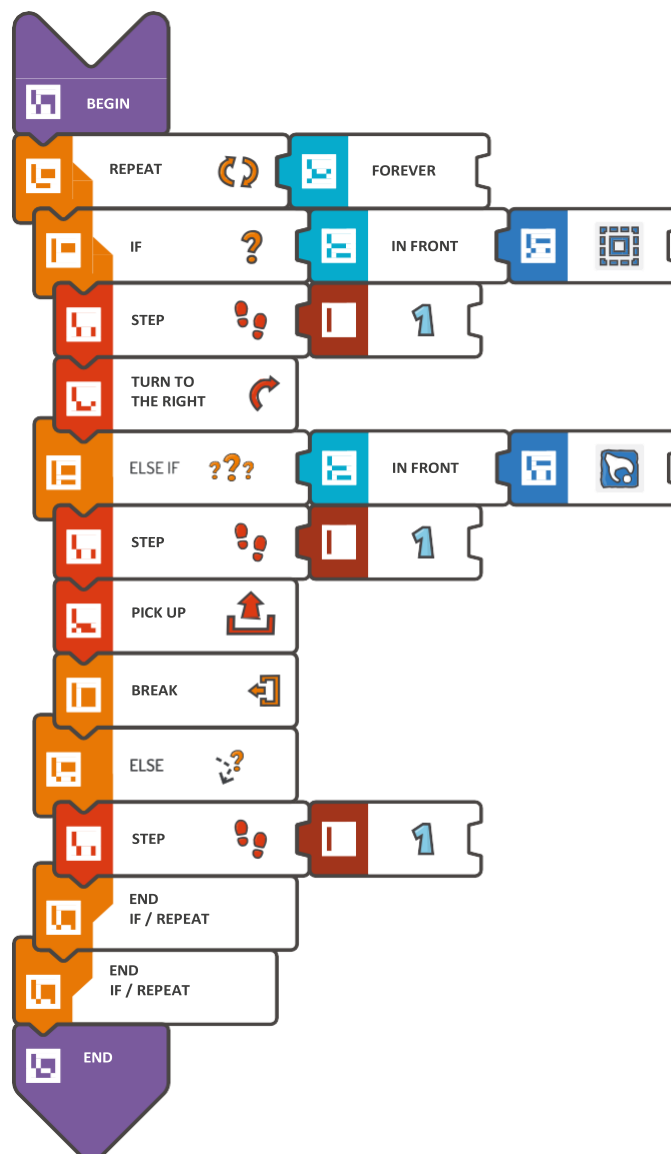
```
REPEAT (FOREVER) {  
  IF (IN FRONT < EMPTY) {  
    STEP (1)  
  } ELSE IF (IN FRONT < MARK - SQUARE) {  
    STEP (1)  
    TURN TO THE LEFT  
  } ELSE {  
    STEP (1)  
    PICK UP  
    BREAK  
  }  
}
```

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

# BEGIN

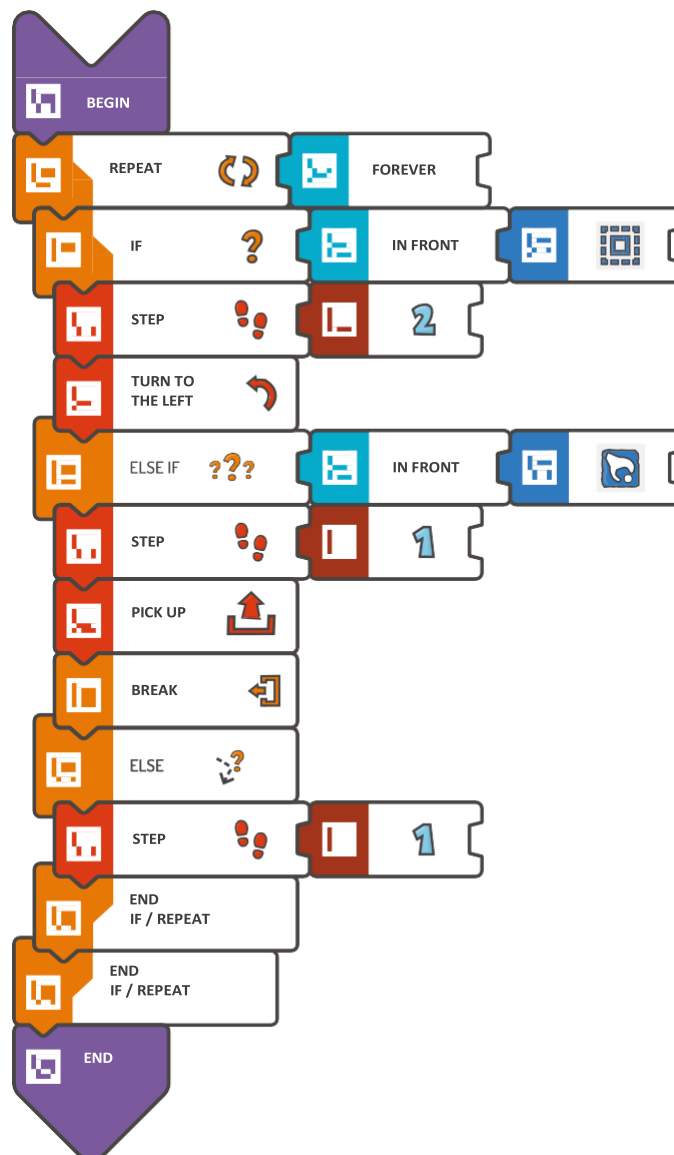
```
REPEAT (FOREVER) {
  IF (IN FRONT < MARK - SQUARE) {
    STEP (1)
    TURN TO THE RIGHT
  } ELSE IF (IN FRONT < COLLECTABLE OBJECT) {
    STEP (1)
    PICK UP
    BREAK
  } ELSE {
    STEP (1)
  }
}
```

**END**

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (FOREVER) {**

**IF (IN FRONT < MARK - SQUARE) {**

**STEP (2)**

**TURN TO THE LEFT**

**} ELSE IF (IN FRONT < COLLECTABLE OBJECT) {**

**STEP (1)**

**PICK UP**

**BREAK**

**} ELSE {**

**STEP (1)**

**}**

**}**

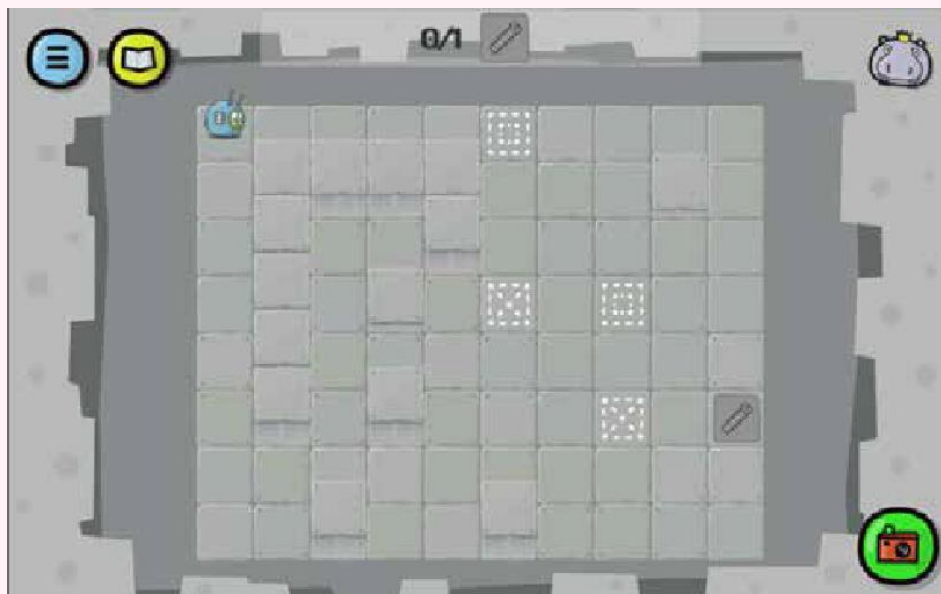
**END**

## HINT

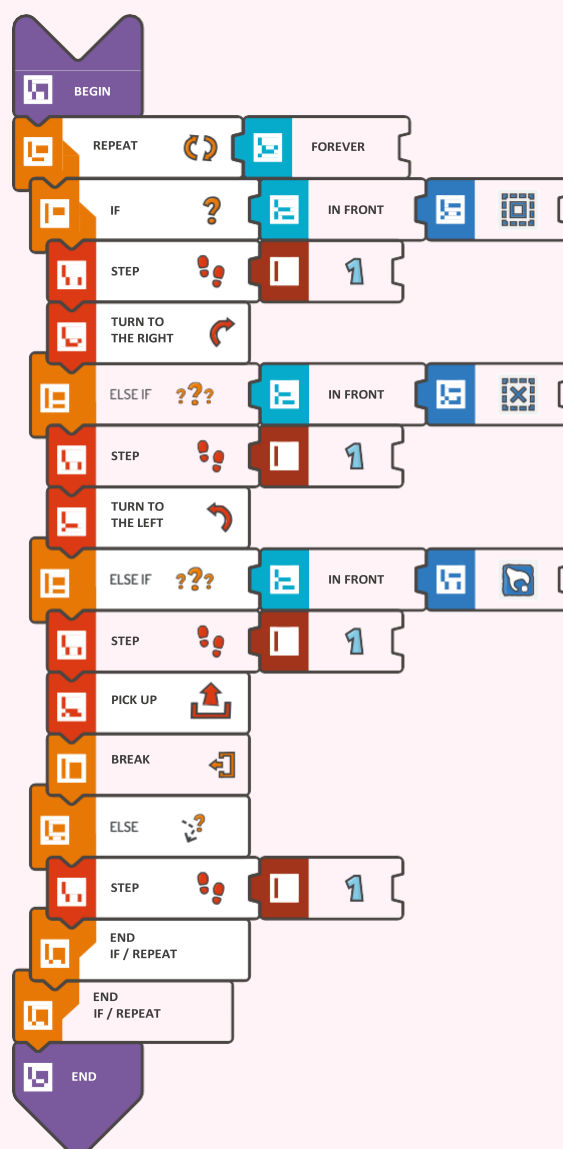
Na de eerste **IF** in je code kun je meerdere **ELSE IF** tegels gebruiken.

## Module IX - Opdracht 5

Opdracht (gebruik een voorwaardelijk commando in deze opdracht):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (FOREVER) {**

**IF (IN FRONT < MARK - SQUARE) {**

**STEP (1)**

**TURN TO THE RIGHT**

**} ELSE IF (IN FRONT < MARK - CROSS) {**

**STEP (1)**

**TURN TO THE LEFT**

**} ELSE IF (IN FRONT < COLLECTABLE OBJECT) {**

**STEP (1)**

**PICK UP**

**BREAK**

**} ELSE {**

**STEP (1)**

**}**

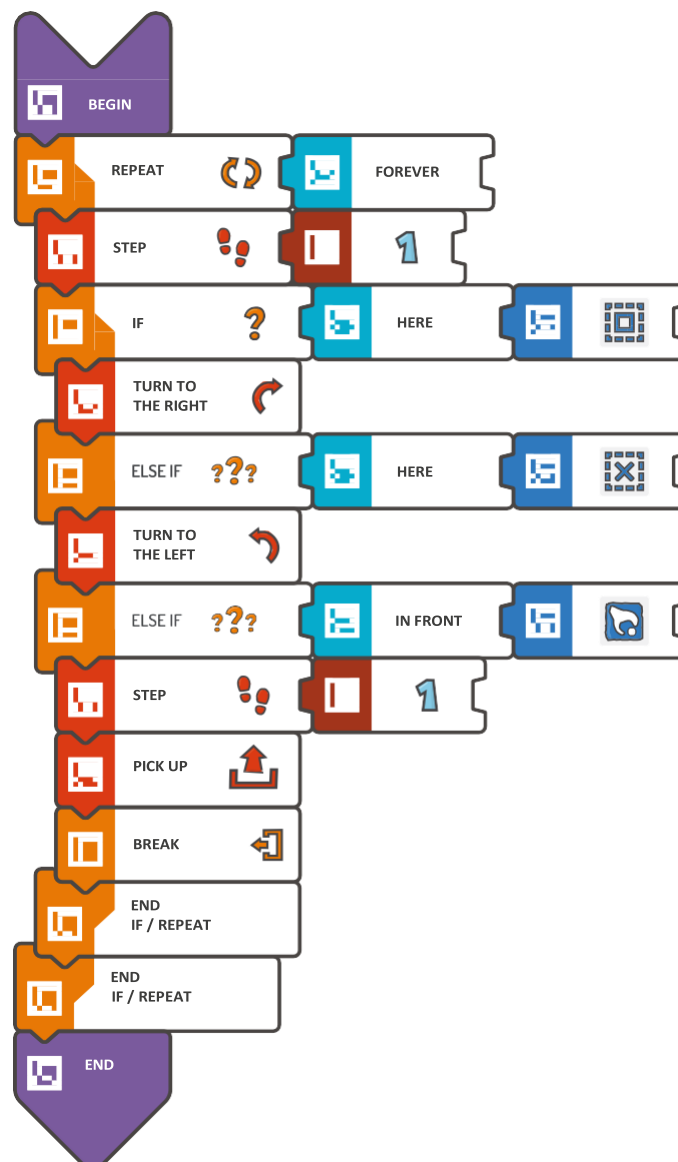
**}**

**END**

Opdracht (gebruik voorwaardelijke commando's in deze opdracht):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (FOREVER) {**

**STEP (1)**

**IF (HERE < MARK - SQUARE) {**

**TURN TO THE RIGHT**

**} ELSE IF (HERE < MARK - CROSS) {**

**TURN TO THE LEFT**

**} ELSE IF (IN FRONT < COLLECTABLE OBJECT) {**

**STEP(1)**

**PICK UP**

**BREAK**

**}**

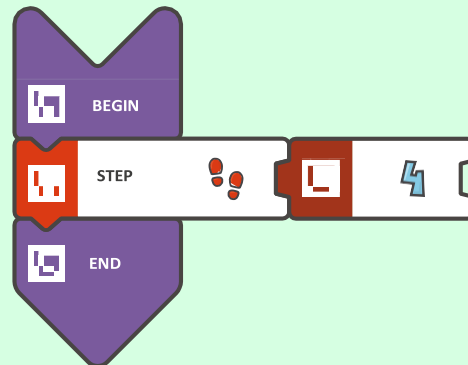
**}**

**END**

## Lopende banden

In de werkplaats staan **lopende banden**. Door deze te gebruiken, hoeft er geen complexe code geschreven te worden om Scottie over de werkplaatsvloer te laten bewegen.

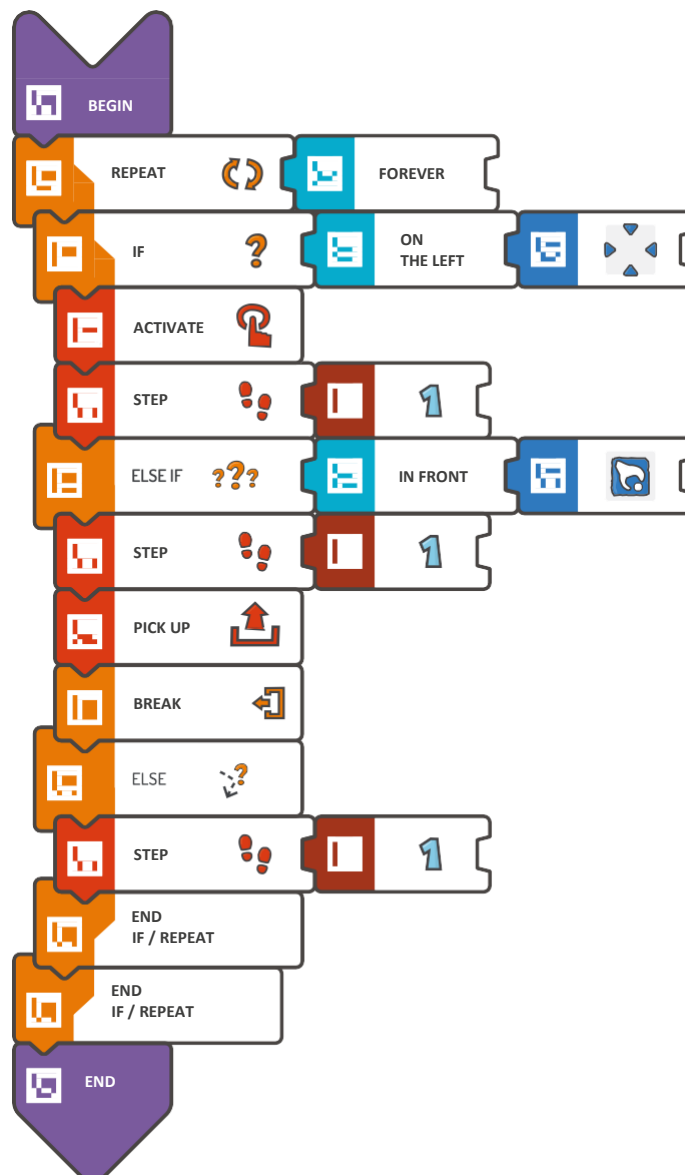
Wanneer Scottie op de lopende band stapt, zal de uitvoering van de code gepauzeerd worden, totdat Scottie op zijn bestemming is gearriveerd.



Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

**REPEAT (FOREVER) {**

**IF (ON THE LEFT < ACTION SQUARE) {**

**ACTIVATE**

**STEP (1)**

**} ELSE IF (IN FRONT < COLLECTABLE OBJECT) {**

**STEP (1)**

**PICK UP**

**BREAK**

**} ELSE {**

**STEP (1)**

**}**

**}**

**END**

# Schakelaars

Sommige lopende banden staan uit. Zoek een schakelaar in de werkplaats, ga daar naartoe en gebruik de **ACTIVATE** tegel om hem aan te zetten.



**Onthoud!**

Om het **ACTIVATE** commando te laten werken, mag **SCOTTIE** niet met zijn rug naar de schakelaar staan.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

```

REPEAT (FOREVER) {
  IF (ON THE RIGHT < ACTION SQUARE) {
    ACTIVATE
    STEP (1)
  } ELSE IF (IN FRONT < COLLECTABLE OBJECT) {
    STEP (1)
    PICK UP
  } ELSE IF (IN FRONT < OBSTACLE) {
    BREAK
  } ELSE {
    STEP (1)
  }
}

```

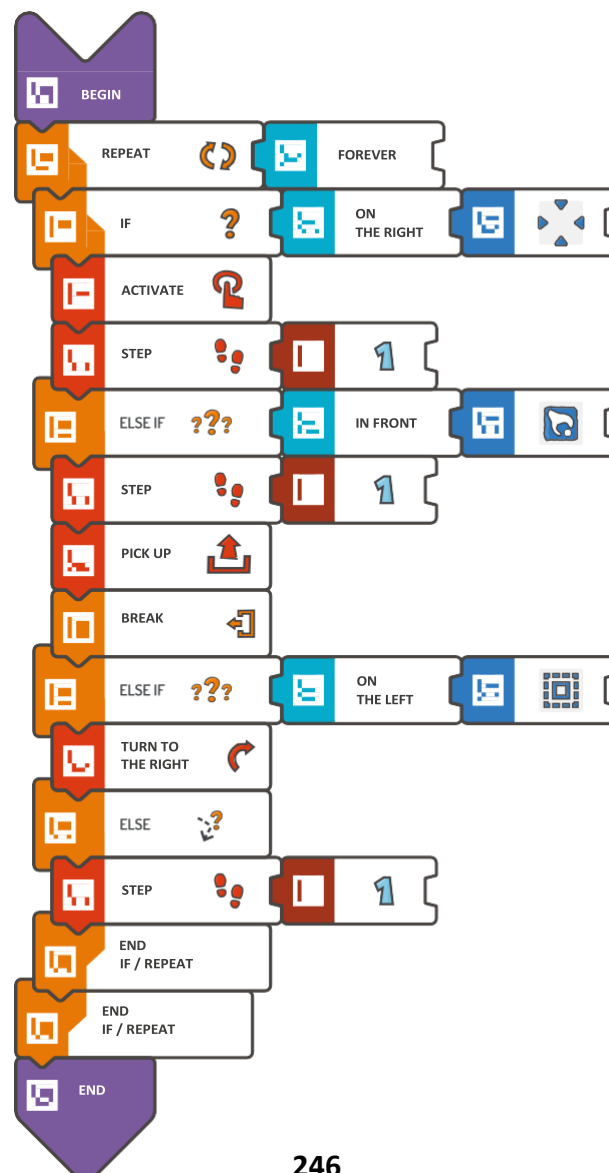
**END**



Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

# BEGIN

```
REPEAT (FOREVER) {
  IF (ON THE RIGHT < ACTION SQUARE) {
    ACTIVATE
    STEP (1)
  } ELSE IF (IN FRONT < COLLECTABLE OBJECT) {
    STEP (1)
    PICK UP
    BREAK
  } ELSE IF (ON THE LEFT < MARK - SQUARE) {
    TURN TO THE RIGHT
  } ELSE {
    STEP (1)
  }
}
```

# Module X

## Afrika



**Functies maken.**

Scottie wil de reparatie van zijn ruimteschip voltooien en de taken die hij zijn dierenvrienden beloofd heeft afronden. Plant de ontbrekende boompjes in het park en teken lijnen om aan te geven waar een pad moet komen. Vergeet niet om bij de uitvoer hiervan functies te gebruiken.

# MODULE INTRODUCTIE: functies

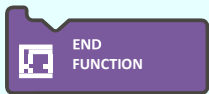
Het komt vaak voor dat je tijdens het schrijven van code dezelfde commandosets keer op keer gebruikt. Dit is echter zonde van je tijd en maakt de code onnodig lang. Het is beter om een oplossing te zoeken, die lijkt op het gebruik van variabelen. Je kunt een set commando's in een speciale verzamelbak plaatsen en deze gebruiken wanneer je hem nodig hebt. Dit maakt je code eenvoudiger, korter en duidelijker.

Deze verzamelbakken noemen we **functies** (subprogramma's) en om ze te kunnen gebruiken, moet je ze eerst aanmaken. Hiervoor gebruik je een aantal specifieke tegels.

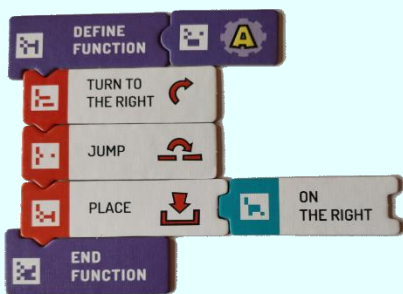
Voor het maken van een functie gebruik je de **DEFINE FUNCTION** tegel en met de: **A, B of C** tegel geef je de functie een naam. In Scottie Go! kunnen we maximaal drie functies tegelijk maken):



We hebben ook de **END FUNCTION** tegel nodig.



Tussen de **DEFINE FUNCTION** en **END FUNCTION** tegels plaats je de set van commando's die je regelmatig in je code nodig hebt bijvoorbeeld:



Wanneer de functie gescand is, kan de functie gebruikt worden. Hiervoor gebruik je de **CALL FUNCTION** tegel in combinatie met de tegel met de naam van de functie (**A, B or C**).

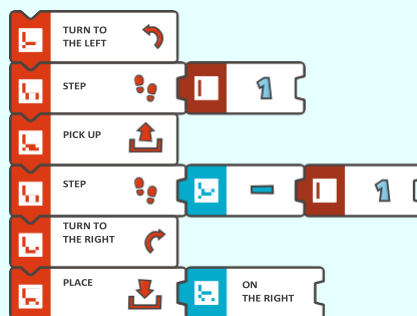
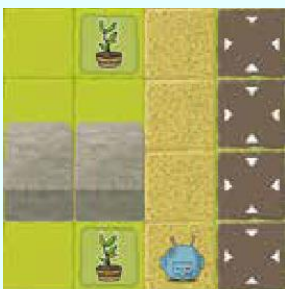
In dit geval:



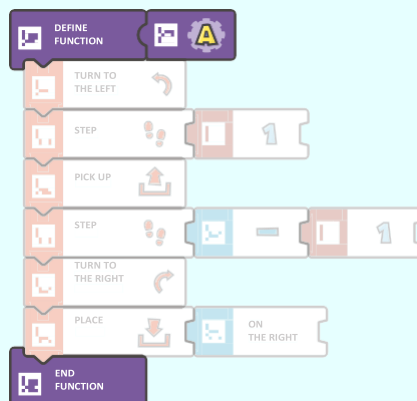
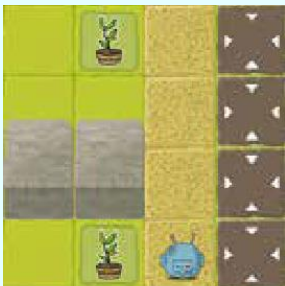
## Voorbeeld:

In onderstaande opdracht, moet Scottie boompjes oppakken en planten op de aangegeven plaatsen (de actievelden). Zoals je ziet, moeten beide taken twee keer uitgevoerd worden. Hiervoor ga je functies maken en gebruiken.

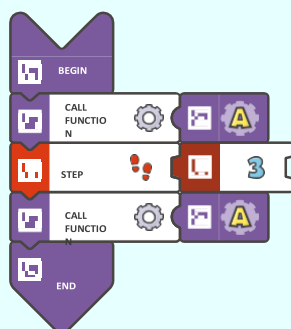
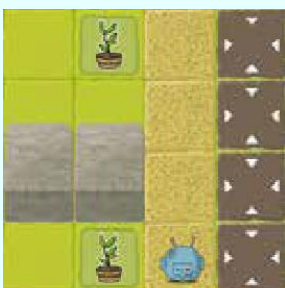
1. Schrijf eerst de code voor het verzamelen en planten van een boompje op de aangegeven plaats.



2. Gebruik deze code om een functie van de maken door er de **DEFINE FUNCTION** en **END FUNCTION** tegel omheen te zetten. Noem de functie **A**, **B** or **C**.



3. Schrijf de code voor het gebruiken van de functie wanneer je hem nodig hebt.



# Functies [extra]

**Je kunt functies in andere functies plaatsen.**

Door functies te gebruiken kun je complexe code schrijven waarbij je:

1. Minder tegels hoeft te gebruiken (door de functies zal je code korter en makkelijker te onderhouden zijn).
2. Beter kunt focussen op de taken die Scottie moet uitvoeren. De introductie van functies, verhoogt het abstractieniveau.

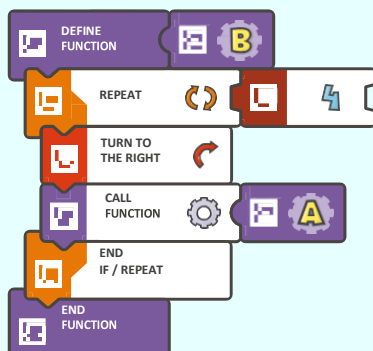
Als je bijvoorbeeld kijkt naar hoe je „veters leert strikken”, dan zie je dit als één taak, terwijl het eigenlijk een reeks complexe bewegingen is, waarbij de schoenveters in een specifieke volgorde bewogen moeten worden.

**Welk pad zal Scottie bewandelen wanneer je:**

**Function A uitvoert?**



**Functie B uitvoert?**



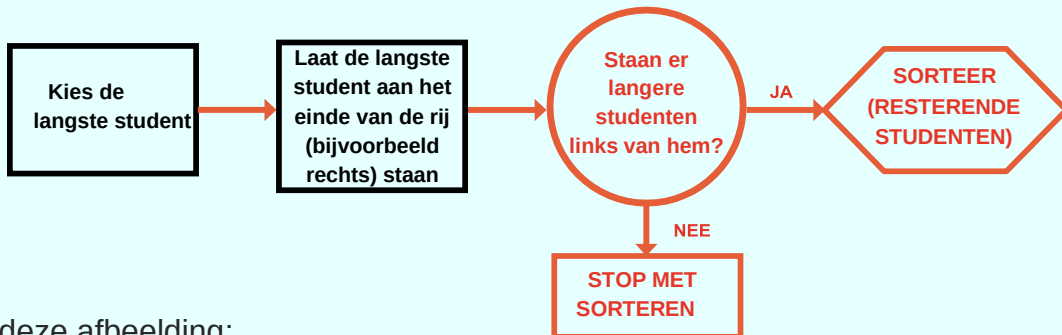
**Functie C uitvoert?**



## Een functie kan zichzelf aanroepen (gebruiken)!

Dit noem je een **recursieve** (zichzelf herhalende) functie. Recursieve functies worden bijvoorbeeld gebruikt bij sorteer algoritmen, bijvoorbeeld algoritmen die worden gebruikt om waarden op volgorde van grootte te rangschikken, of boeken op titel.

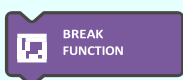
Bekijk het volgende algoritme dat gebruikt kan worden om studenten op volgorde van lengte te sorteren:



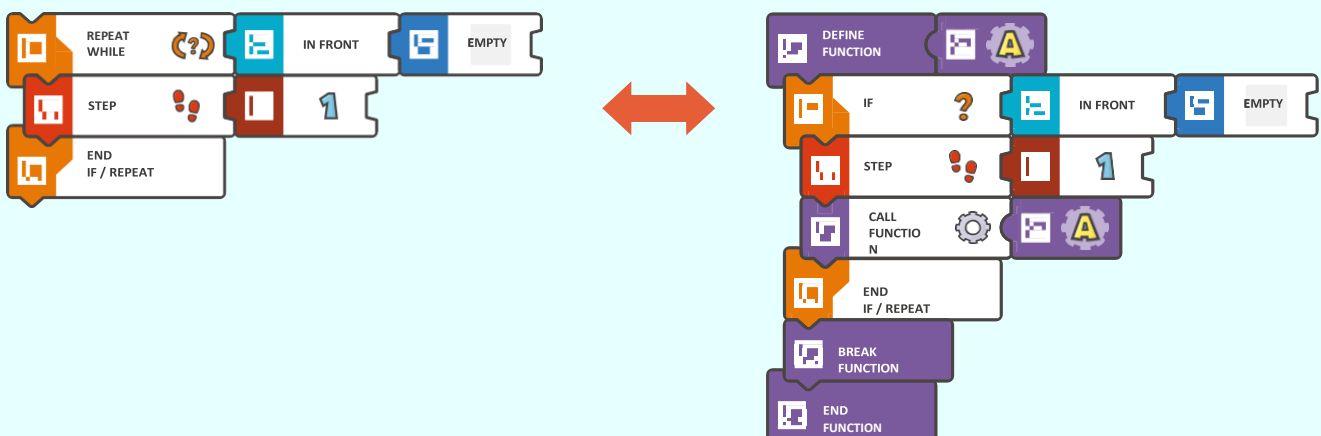
of bekijk deze afbeelding:



Het sleutelement in recursieve functies is het voorwaardelijke commando dat de functie onderbreekt als er aan de voorwaarde voldaan wordt. Zonder dit voorwaardelijke commando, zal de functie nooit stoppen en een fout genereren (of de computer bezig houden tot het geheugen vol is). Om de functie te laten stoppen, hebben we de **BREAK FUNCTION** tegel nodig.



Als we een recursieve functie gebruiken, kunnen we onze eigen voorwaardelijke lus maken:



# Eerste functie, HEUVELS

In deze opdracht moet je functie A gebruiken door de **CALL FUNCTION** tegel in combinatie met de **A** tegel te gebruiken (A is de naam van de functie).



Je moet echter eerst de functie maken door de code te schrijven, die uitgevoerd moet worden als de functie gebruik wordt. Gebruik de **DEFINE FUNCTION** tegel samen met de **A** tegel (waarbij de A weer voor de naam van de functie staat) en aan het einde van de code de **END FUNCTION** tegel.



## Bijvoorbeeld:



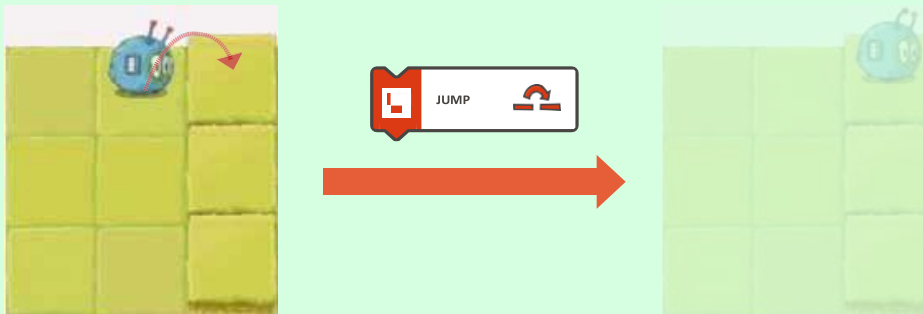
## In tekstformaat:

```
DEFINE FUNCTION A {  
    STEP (2)  
}
```

### ONTHOUD!

- Stap 1** – MAAK DE FUNCTIES (maximaal drie functies: **A, B, C**)
- Stap 2** - SCAN DE FUNCTIES (maximaal drie functies: **A, B, C**)
- Stap 3** – SCHRIJF DE CODE WAARIN JE DE FUNCTIES GEBRUIKT
- Stap 4** - SCAN HET PROGRAMMA EN LAAT HET UITVOEREN

Bij sommige opdrachten in deze module kan het zijn dat Scottie op een heuvel moet springen om de opdracht met zo min mogelijk regels de code op te lossen.



De enige manier om op een heuvel te komen is het gebruik van de **JUMP** tegel. Om van de heuvel af te gaan, kun je de **JUMP** of de **STEP** tegel gebruiken.

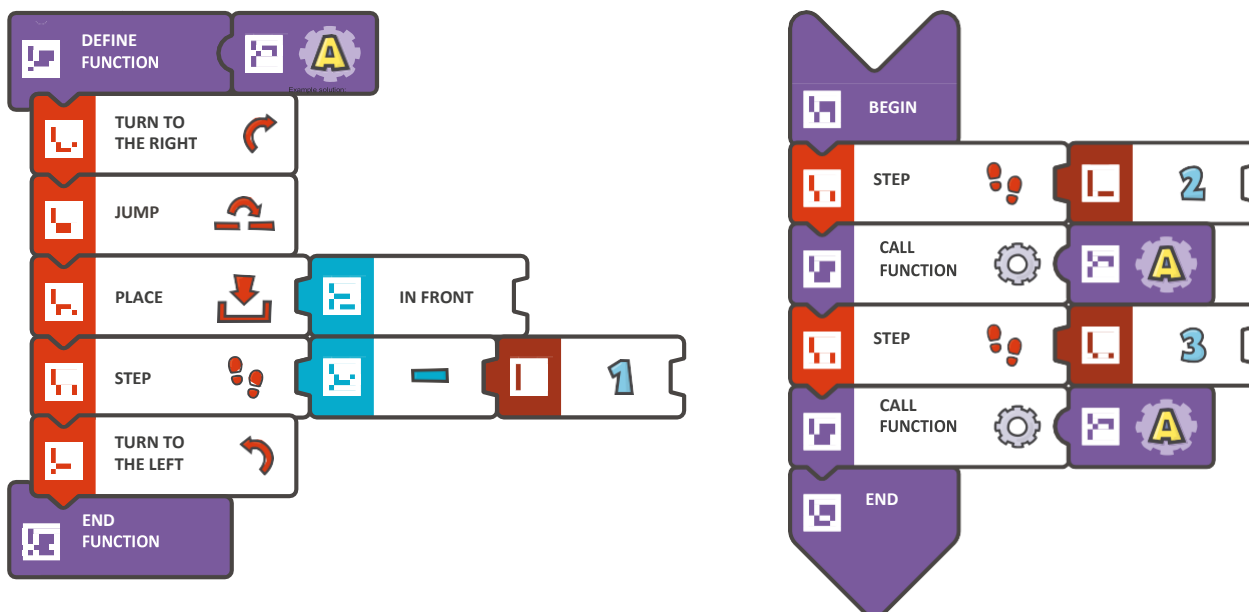
In je code kun je ook naar een heuvel verwijzen door de **HEUVEL** tegel te gebruiken.



Opdracht: (maak en gebruik je zelf gemaakte functie, plant boompjes op de actievelden):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

```

DEFINE FUNCTION A {
  TURN TO THE RIGHT
  JUMP
  PLACE (IN FRONT)
  STEP (-1)
  TURN TO THE LEFT
}

```

```

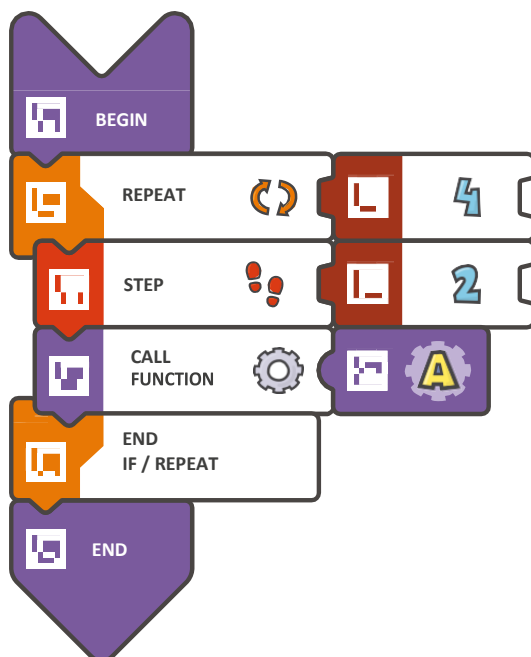
BEGIN
  STEP (2)
  CALL FUNCTION A
  STEP (3)
  CALL FUNCTION A
END

```

Opdracht: (gebruik je zelf gemaakte functie vier keer, gebruik hiervoor de REPEAT tile):



Voorbeeldoplossing:



Opdracht: (gebruik je zelf gemaakte functie vier keer, gebruik hiervoor de REPEAT lus):



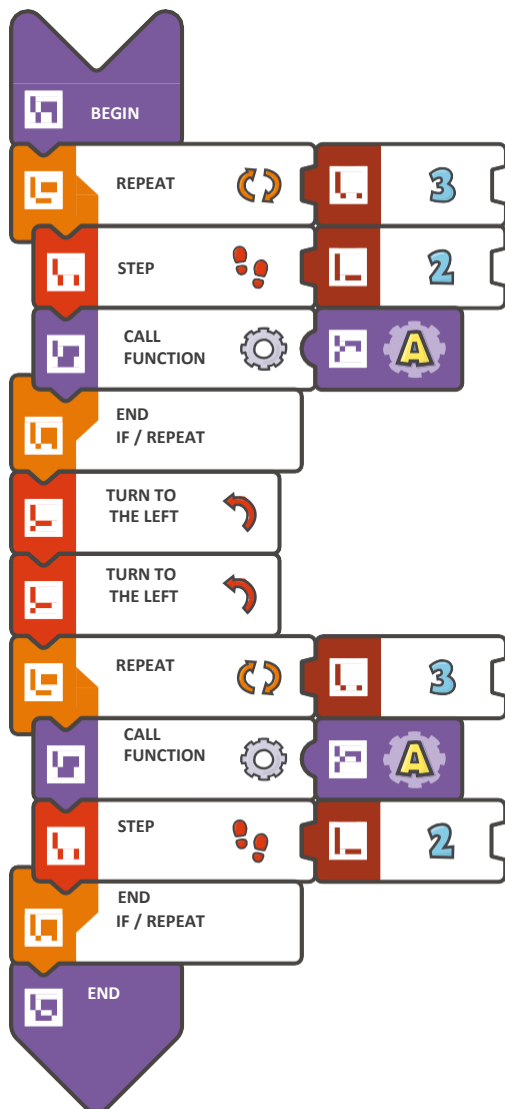
Voorbeeldoplossing (in tekstformaat):

```
BEGIN
  REPEAT (4) {
    STEP (2)
    CALL FUNCTION A
  }
END
```

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

**BEGIN**

```
REPEAT (3) {  
  STEP (2)  
  CALL FUNCTION A  
}
```

TURN TO THE LEFT

TURN TO THE LEFT

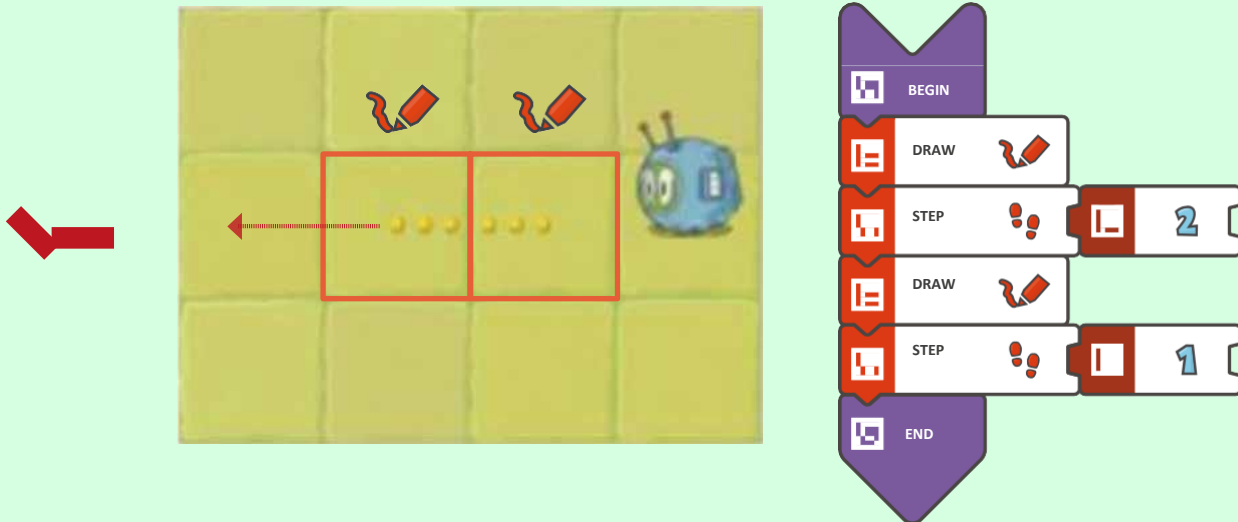
```
REPEAT (3) {  
  CALL FUNCTION A  
  STEP (2)  
}
```

**END**

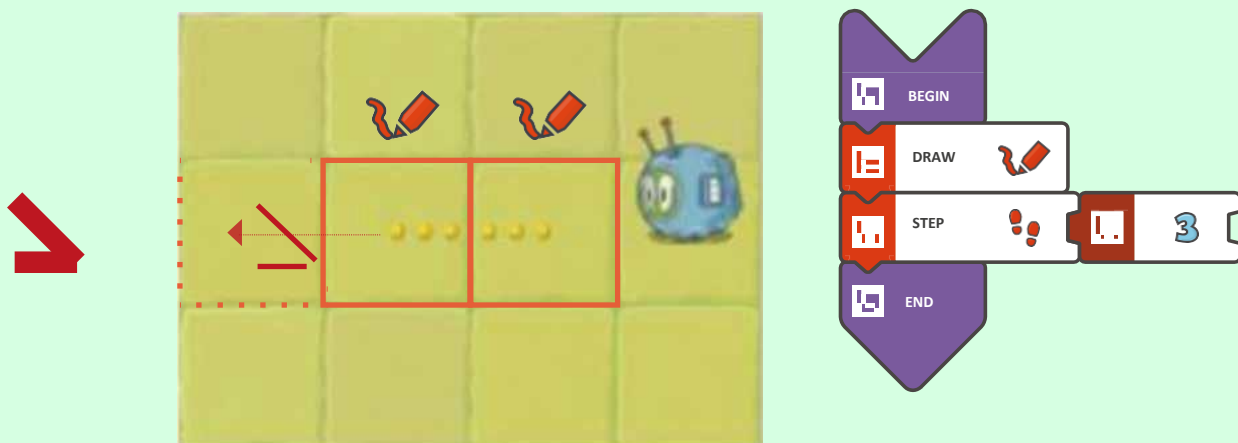
# Tekenen

Scottie heeft de taak gekregen om lijnen te tekenen, om de routes te markeren, waar paden gebouwd moeten worden in het park.

Om Scottie deze taak uit te laten voeren, gebruik je de **DRAW** tegel. Gebruik de tegel zowel voor het starten als voor het beëindigen van het tekenen. In de tekenmodus zal Scottie een lijn achterlaten op elk veld waar hij op gaat staan of overheen loopt.



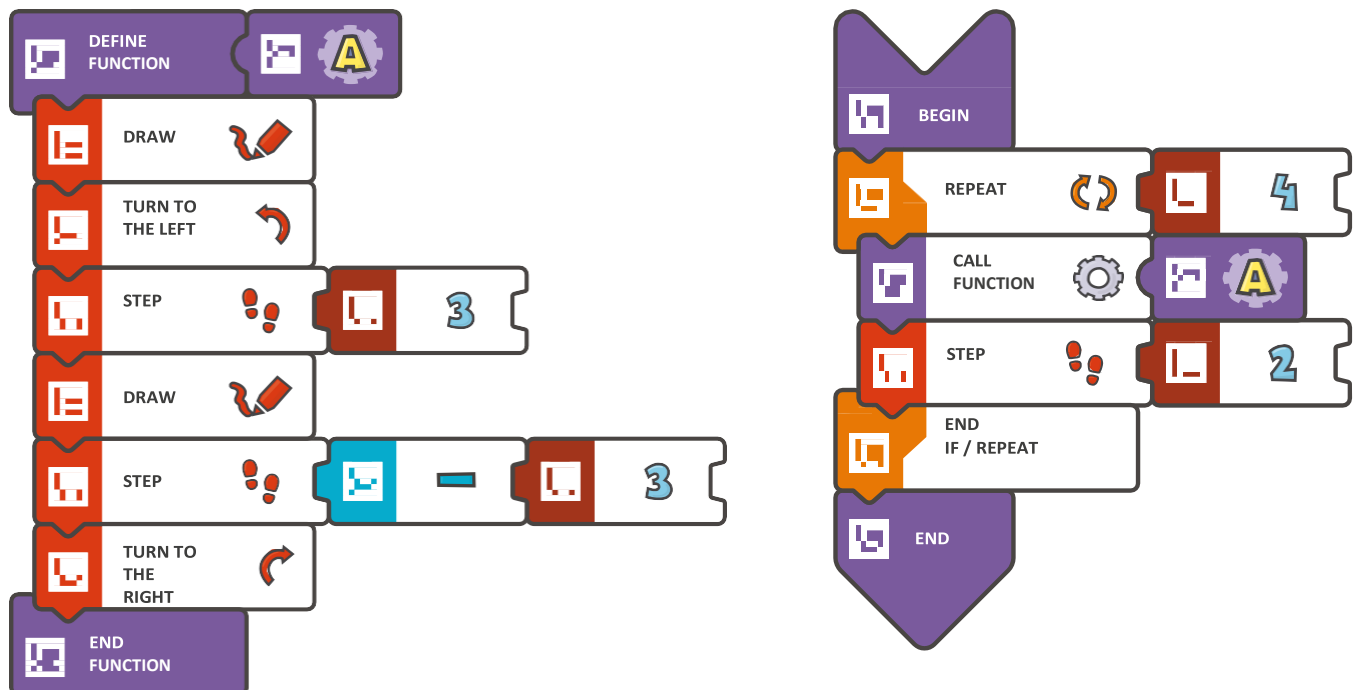
Tekenen op velden zonder stippellijn geeft een foutmelding.



Opdracht (maak je eigen functie A en gebruik de DRAW tegel in de functie):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

```

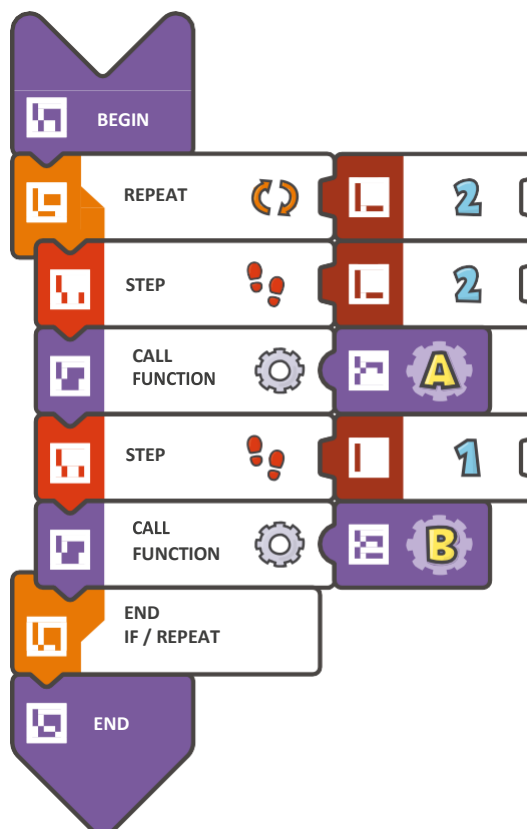
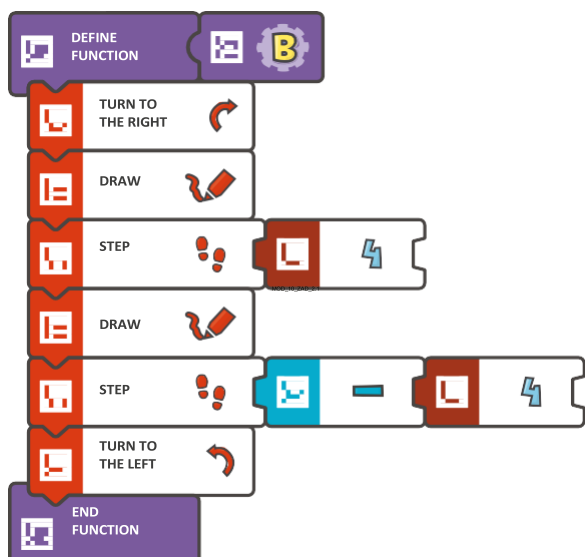
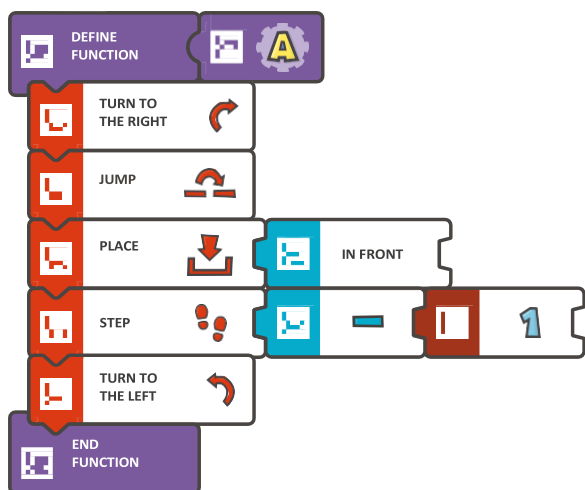
DEFINE FUNCTION A {
  DRAW
  TURN TO THE LEFT
  STEP (3)
  DRAW
  STEP (-3)
  TURN TO THE RIGHT
}

BEGIN
  REPEAT (4) {
    CALL FUNCTION A
    STEP (2)
  }
END
  
```

Opdracht (maak twee eigen functies A en B en gebruik ze in de code):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

```

DEFINE FUNCTION A {
  TURN TO THE RIGHT
  JUMP
  PLACE (IN FRONT)
  STEP (-1)
  TURN TO THE LEFT
}

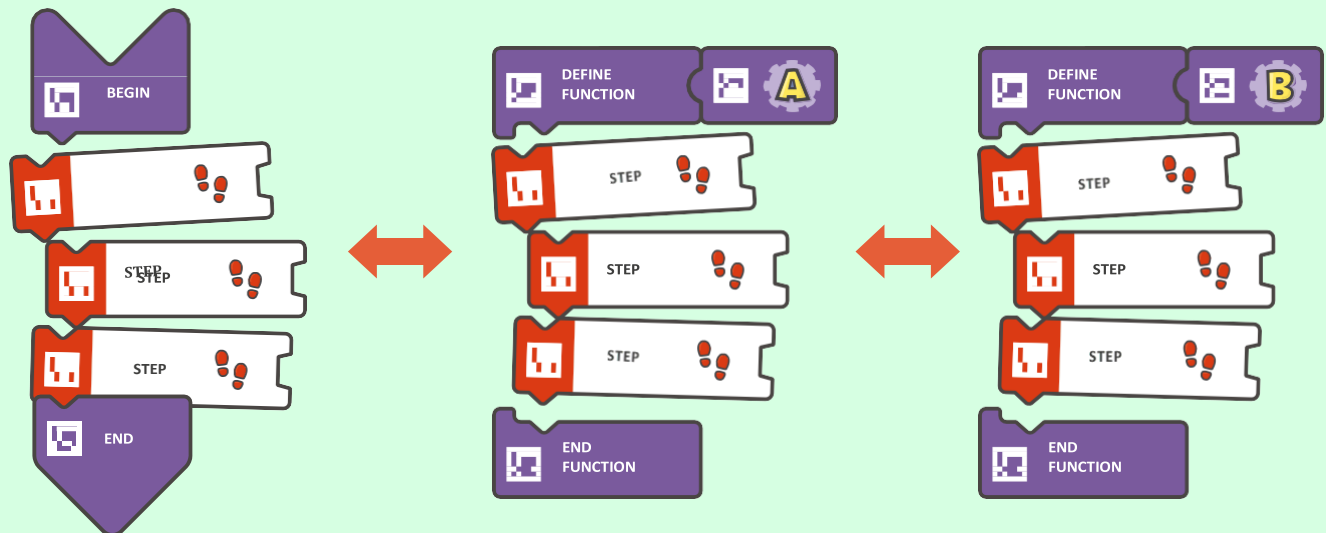
BEGIN
  REPEAT (2) {
    STEP (2)
    CALL FUNCTION A
    STEP (1)
    CALL FUNCTION B
  }

END

DEFINE FUNCTION B {
  TURN TO THE RIGHT
  DRAW
  STEP (4)
  DRAW
  STEP (-4)
  TURN TO THE LEFT
}
  
```

## Een tegel meerdere keren gebruiken

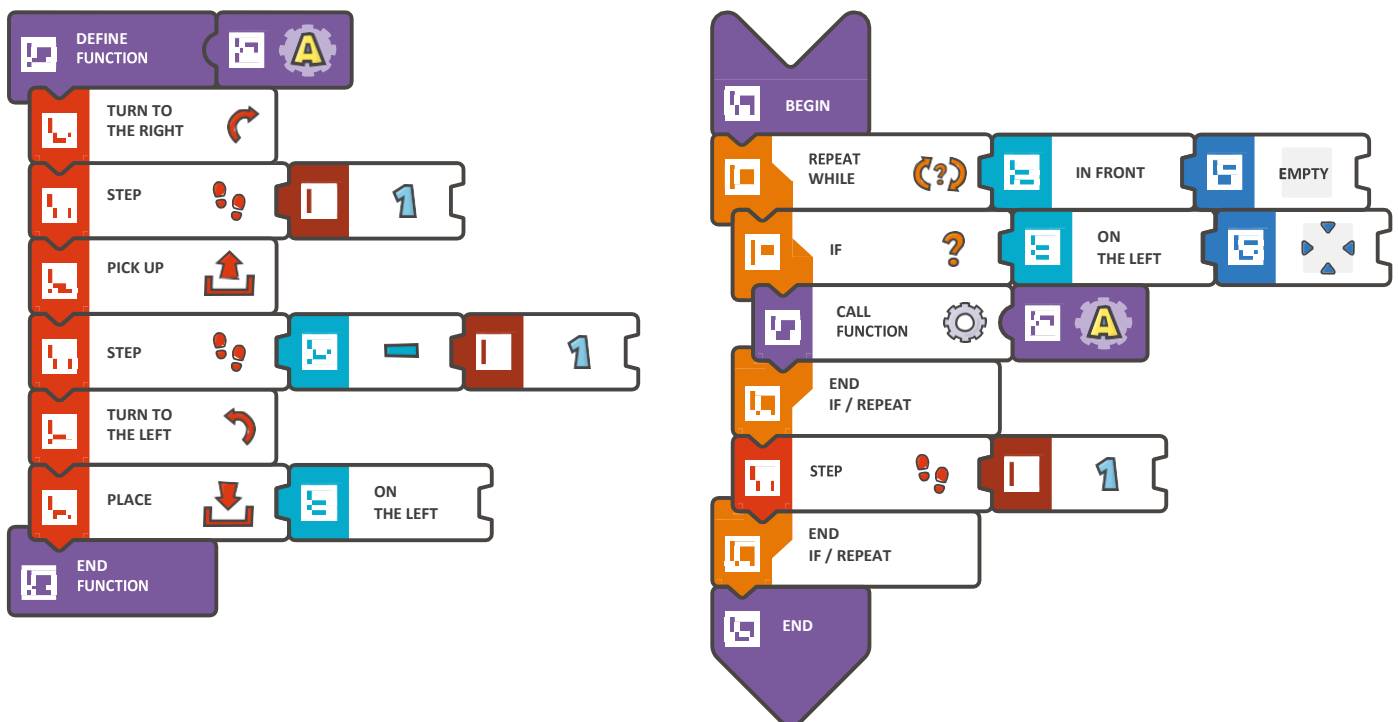
Het aantal tegels in de doos is beperkt, maar je kunt dezelfde tegels gebruiken voor het maken van verschillende functies en voor het schrijven van de code zelf. Je kunt de tegels bijvoorbeeld onderling uitwisselen tussen de functies en de code wanneer je gaat scannen.



Opdracht (gebruik de REPEAT WHILE en de IF tegel in deze opdracht):



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

```

DEFINE FUNCTION A {
  TURN TO THE RIGHT
  STEP (1)
  PICK UP
  STEP (-1)
  TURN TO THE LEFT
  PLACE (ON THE LEFT)
}

```

```

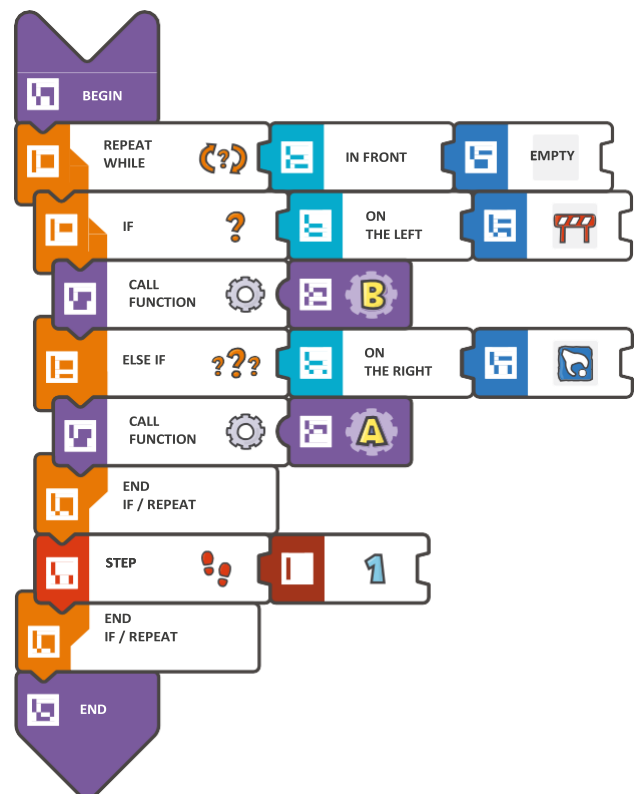
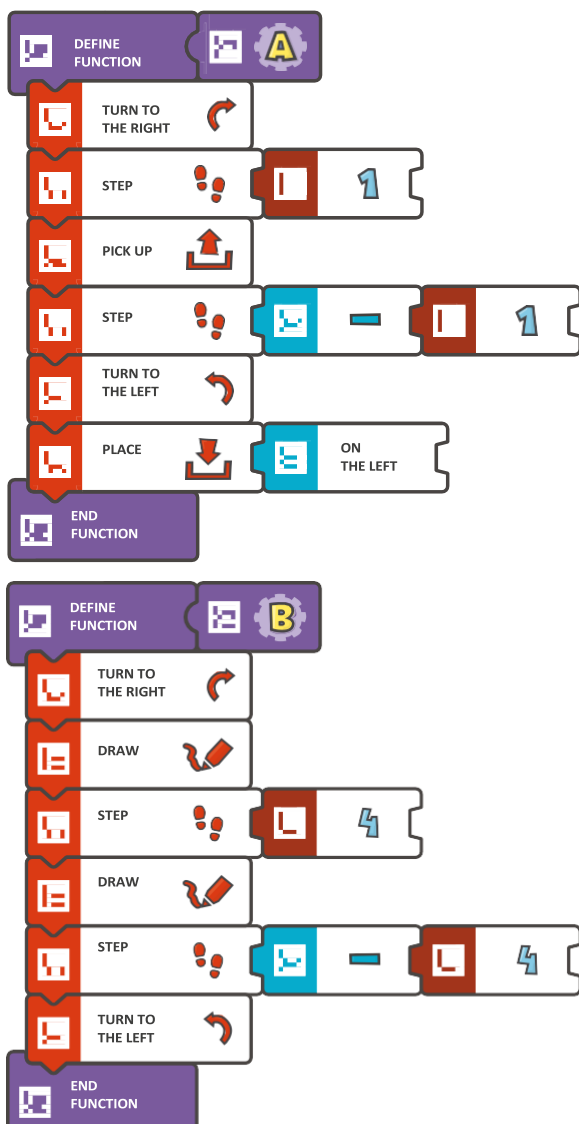
BEGIN
  REPEAT WHILE (IN FRONT < EMPTY) {
    IF (ON THE LEFT < ACTION SQUARE) {
      CALL FUNCTION A
    }
    STEP (1)
  }
END

```

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

```

DEFINE FUNCTION A {
  TURN TO THE RIGHT
  STEP (1)
  PICK UP
  STEP (-1)
  TURN TO THE LEFT
  PLACE (ON THE LEFT)
}

DEFINE FUNCTION B {
  TURN TO THE RIGHT
  DRAW
  STEP (4)
  DRAW
  STEP (-4)
  TURN TO THE LEFT
}

BEGIN
  REPEAT WHILE (IN FRONT < EMPTY) {
    IF (ON THE LEFT < OBSTACLE) {
      CALL FUNCTION B
    } ELSE IF
    (ON THE RIGHT < COLLECTABLE
    OBJECT) {
      CALL FUNCTION A
    }
    STEP (1)
  }
END
  
```

In Scottie Go! kunnen we maximaal drie functies maken (A, B, C) die we in één code kunnen gebruiken. Als je een functie gescand hebt, zal het icoon van die functie links in het scherm verschijnen.



Functies die nog niet gemaakt zijn, worden grijs weergegeven. Druk op de naam van de functie om de code van de functie op het scherm te bekijken.

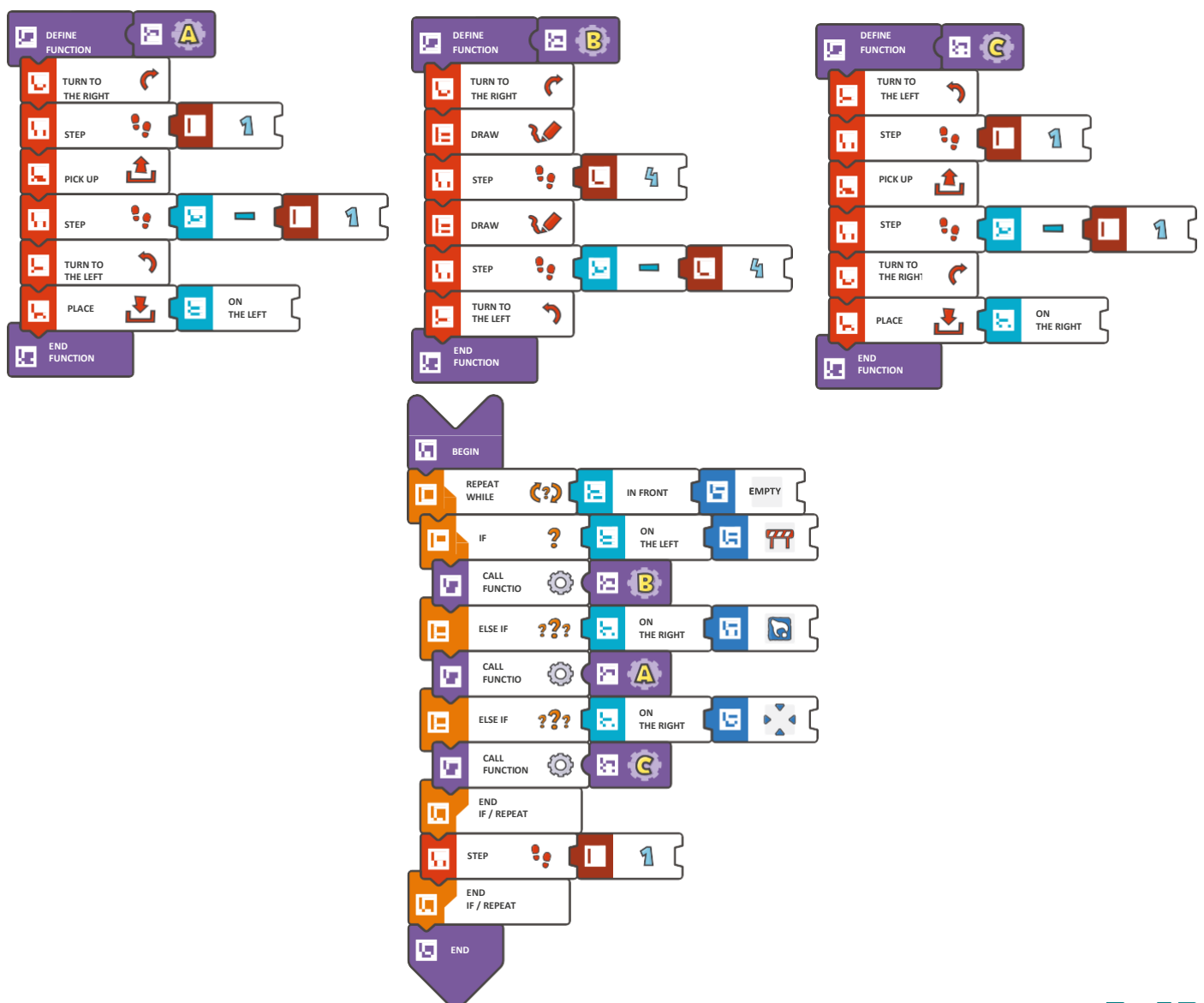


Op elk gewenst moment kun je een gemaakte functie uit het geheugen van je apparaat verwijderen. Als je een nieuwe code voor een functie scant en deze de naam geeft van een eerder gemaakte functie, dan zal de oude code van de functie vervangen worden door de nieuwe code.

Opdracht:



Voorbeeldoplossing:





Voorbeeldoplossing (in tekstformaat):

```

DEFINE FUNCTION A {
  TURN TO THE RIGHT
  STEP (1)
  PICK UP
  STEP (-1)
  TURN TO THE LEFT
  PLACE (ON THE LEFT)
}

```

```

DEFINE FUNCTION B {
  TURN TO THE RIGHT
  DRAW
  STEP (4)
  DRAW
  STEP (-4)
  TURN TO THE LEFT
}

```

```

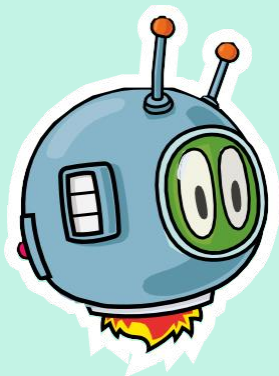
DEFINE FUNCTION C {
  TURN TO THE LEFT
  STEP (1)
  PICK UP
  STEP (-1)
  TURN TO THE RIGHT
  PLACE (ON THE RIGHT)
}

```

```

BEGIN
  REPEAT WHILE (IN FRONT < EMPTY) {
    IF (ON THE LEFT < OBSTACLE ) {
      CALL FUNCTION B
    } ELSE IF (ON THE RIGHT < COLLECTABLE
      OBJECT) {
      CALL FUNCTION A
    } ELSE IF (ON THE RIGHT < ACTION
      SQUARE) {
      CALL FUNCTION C
    }
    STEP (1)
  }
END

```



## **DANK JE WEL GENIET VAN SCOTTIE GO!**

Sommige van de voorbeeldoplossingen kunnen nog korter.

Dit is nog maar het begin van je plezier met Scottie Go! Onthoud dat je alle uitdrukkingen, commando's en functies in elke module kunt gebruiken.

Als je opmerkingen of vragen hebt over deze docentenhandleiding, e-mail ons via **info@scottiego.nl** en zet **"DOCENTENHANDLEIDING"** in het onderwerp van je e-mail.