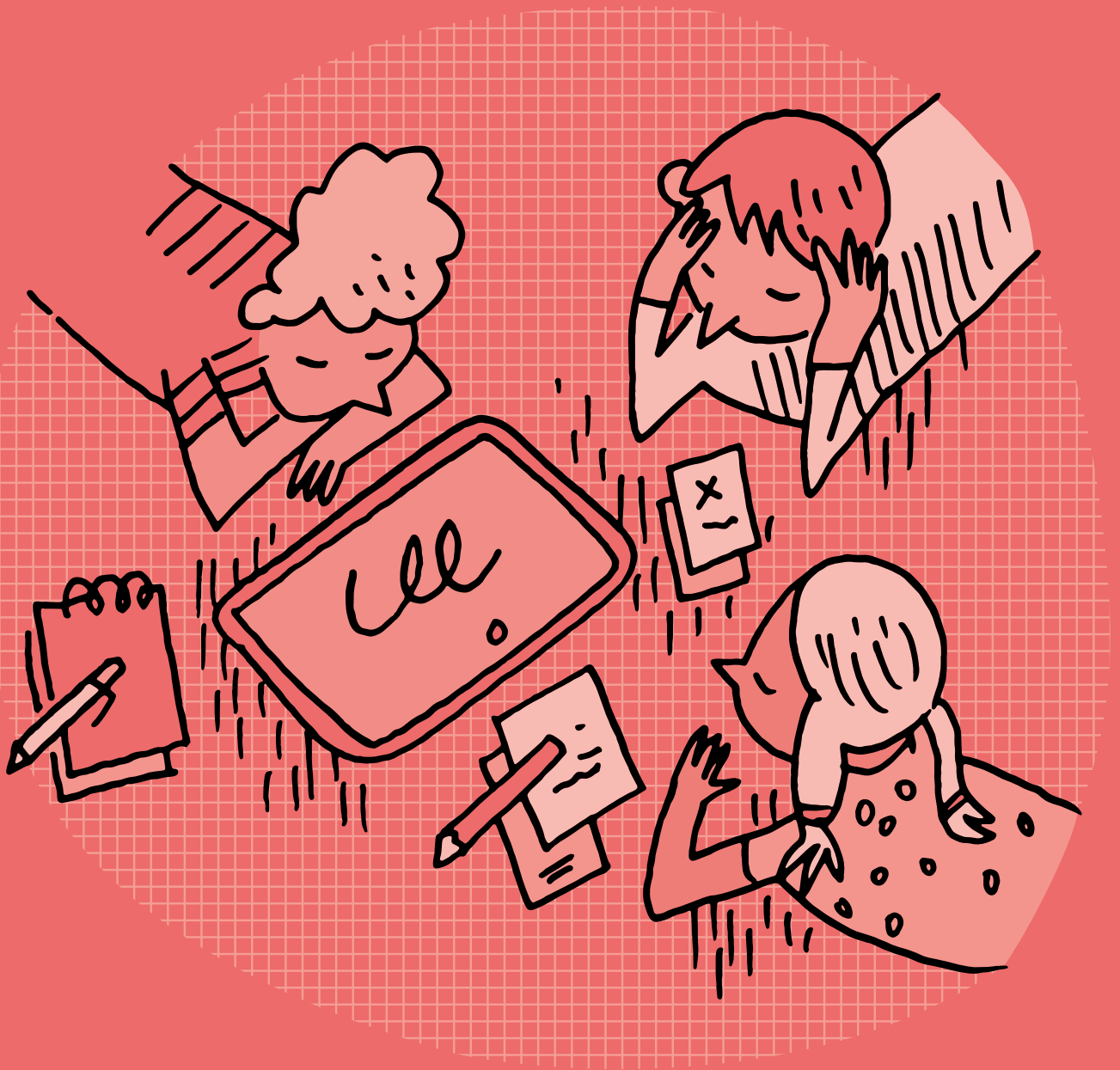


Enquête 2 • **SI** • 5<sup>e</sup>

# Pourquoi ce programme ne fonctionne-t-il pas ?



SI • 5<sup>e</sup>

## Pourquoi ce programme ne fonctionne-t-il pas ?

### 🎯 Objectifs du Plan d'études romand (PER):

**EN 22 – S'appropriier les concepts de base de la science informatique...**

**4** ... en créant, en exécutant, en comparant et en corrigeant des programmes

#### Algorithmes et programmation

- Création et comparaison de programmes avec des séquences, des tests conditionnels et des boucles à l'aide d'un langage de programmation visuel pour résoudre des problèmes simples

#### Liens disciplinaires:

- MSN 25 – Modélisation
- SHS 21 – Relation Homme-Espace; SHS 23 – Outils et méthodes de recherche

### 💡 Intentions pédagogiques:

Cette enquête vise à faire comprendre aux élèves la notion de bug, et comment le corriger. Elle permet de comprendre et de concevoir le bug non pas comme un problème en soi, mais comme la cause d'un problème. Il arrive parfois qu'un ordinateur ne fasse pas ce qu'on lui demande. Souvent, on dit qu'il bugue. Il faut alors mener l'enquête pour savoir d'où provient l'erreur. Cette enquête s'intéresse en particulier aux bugs dans un programme (ici sous ScratchJr). Lorsqu'un programme ne réalise pas ce à quoi on s'attend, ce qui est pensé, il faut traquer l'erreur, trouver l'algorithme (succession d'étapes permettant de résoudre un problème et d'écrire les programmes informatiques) qui n'est pas correctement écrit et qui provoque une erreur dans le programme informatique. Comment faire pour mettre en place une recherche méthodique de l'erreur ?

🔍 Cette enquête s'appuie sur des programmes sous ScratchJr. Il est donc nécessaire qu'en amont du travail proposé ici, la classe ait abordé une activité ou un scénario de ScratchJr.

La notion de bug peut être complexe à cerner. En effet, un bug peut avoir pour origine:


- l'erreur humaine (erreur dans l'écriture d'un programme par exemple - cas traité ici)
- un souci dans le traitement des données (programme inadapté, sans solution face à un problème donné, dépassé par le nombre de données à traiter...)
- un souci relevant de la machine (souci mécanique par exemple).

Il peut être également intéressant de prévoir un temps sur l'historique de la notion du bug. On trouvera à la fin de cette enquête un éclairage scientifique à ce propos.

Cette enquête peut être l'occasion d'aborder avec les élèves un aspect très important de la transmission de l'information, lié à la notion de bug évoquée ci-dessus.

En effet, le travail sur l'écriture d'un programme permet de découvrir la nécessité de l'exactitude des informations transmises à la machine. Il suffit d'un bug dans le programme pour produire un résultat non conforme. D'où la nécessité de mettre en place des procédures de vérification, permettant de s'assurer que le programme effectue bien la tâche demandée.






 Les programmes proposés dans le déroulé de l'enquête sont volontairement simples, de façon à aider les élèves à entrer dans cette activité de recherche de bugs.

Suivant le niveau de maîtrise, on pourra proposer des programmes plus complexes. La démarche, qui reste strictement identique à celle présentée ci-dessus, est proposée en fin d'enquête sous la rubrique « Compléments – Prolongements – Variantes ». Cette rubrique contient tous les éléments donnés permettant de réaliser l'enquête avec ces programmes plus complexes :

- la présentation de la situation
- les programmes « bugués »
- les programmes décomposés

La question de l'enquête :

### Pourquoi ce programme ne fonctionne-t-il pas ?

Étape	Résumé	Matériel
<p><b>1.</b> Pour comprendre la question</p> <p> <b>Durée:</b> 45 minutes</p>	<p><b>Début de l'investigation:</b></p> <ul style="list-style-type: none"> <li>• Observer en débranché des situations provenant de ScratchJr.</li> <li>• Émettre des hypothèses sur ce qui va se passer.</li> <li>• Donner les programmes associés.</li> <li>• Valider/invalidier les premières hypothèses.</li> <li>• Tester les programmes sur les tablettes et observer ce qu'il se passe.</li> <li>• Mesurer l'écart entre les effets attendus et ce qui se réalise vraiment.</li> <li>• Appréhender/comprendre et rechercher des bugs.</li> </ul>	<ul style="list-style-type: none"> <li>• fiche 1 (à projeter)</li> <li>• fiche 2 (1 par élève)</li> <li>• fiche 3 (1 étiquette de chaque programme par élève)</li> <li>• tablette/application ScratchJr (1 par groupe)</li> <li>• affichage numérique</li> </ul>
<p><b>2.</b> Pour répondre à la question</p> <p> <b>Durée:</b> 35 minutes</p>	<p><b>Poursuite de l'investigation:</b></p> <ul style="list-style-type: none"> <li>• Étudier avec méthode les programmes donnés afin de trouver l'erreur.</li> <li>• Lister les types d'erreurs.</li> <li>• Débattre et discuter en binôme pour chercher des solutions afin de corriger les programmes.</li> <li>• Proposer aux élèves un outil permettant de corriger les programmes à partir de la catégorisation des blocs de ScratchJr.</li> </ul>	<ul style="list-style-type: none"> <li>• fiche 2 (distribuée lors de l'étape 1)</li> <li>• fiches 4 et 6 (1 par élève)</li> <li>• fiche 5 (à projeter)</li> <li>• tablette/application ScratchJr (1 par groupe)</li> <li>• affichage numérique</li> </ul>
<p><b>3.</b> Pour conclure</p> <p> <b>Durée:</b> 30 minutes</p>	<p><b>Conclusion:</b></p> <ul style="list-style-type: none"> <li>• Utiliser un outil qui chasse les bugs basé sur la méthode validée préalablement pour vérifier des programmes donnés.</li> <li>• Rédiger une trace écrite (affiche-outil pour la classe).</li> </ul>	<ul style="list-style-type: none"> <li>• fiche 4 (distribuée lors de l'étape 2)</li> <li>• fiche 6 (1 par élève)</li> <li>• fiche 7 (1 par groupe ou par élève)</li> </ul>

# Étape 1

## Éléments pour comprendre la question

### Résumé:

- Observer en débranché des situations provenant de ScratchJr.
- Émettre des hypothèses sur ce qui va se passer.
- Donner les programmes associés.
- Valider/invalidier les premières hypothèses.
- Tester les programmes sur les tablettes et observer ce qu'il se passe.
- Mesurer l'écart entre les effets attendus et ce qui se réalise vraiment.
- Appréhender/comprendre et rechercher des bugs.

### Matériel:

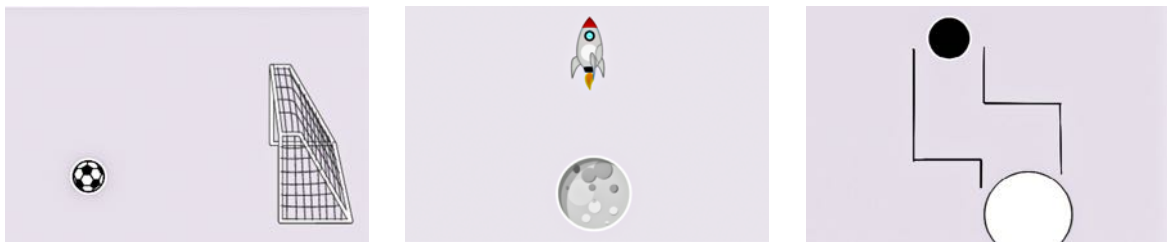
- fiche 1 (à projeter)
- fiche 2 (1 par élève)
- fiche 3 (1 étiquette de chaque programme par élève)
- tablette/application ScratchJr (1 par groupe)
- affichage numérique

## Temps 1.1: Observation des situations, lecture des programmes, émission d'hypothèses

Modalités de travail: en binômes

 Durée: 15 minutes

On projette la fiche 1 contenant les différentes situations. L'idée est de créer un « horizon d'attente » qui sera ensuite confronté à la réalité d'un programme.



Les situations sont choisies et préparées de manière à être les plus explicites possible. Lorsque l'on voit un ballon de football et des buts, on s'attend à ce que le ballon rentre dans les buts. De la même façon, la fusée doit se poser sur la Lune et la balle aller au bout de son parcours.

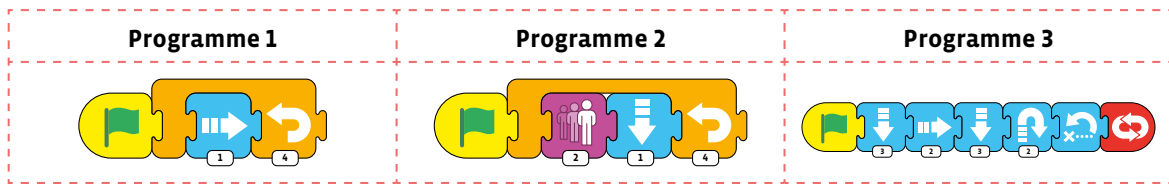
Les élèves consignent, en binômes, leurs hypothèses par écrit (voir fiche 2 - partie 1).

On attend des élèves ici des choses simples comme: **le ballon va dans le but, la fusée va sur la Lune, le rond noir va vers le rond blanc**. Si besoin, on peut aider les élèves qui rencontreraient des difficultés en épelant des mots, en proposant des étiquettes vocabulaires (ballon, but, fusée, Lune...).

Puis on distribue les étiquettes (préalablement découpées) de la fiche 3 à coller sur la fiche 2 avec les 3 programmes associés à chaque situation. On est toujours en **débranché**, les élèves n'ont donc pas la possibilité de vérifier. Cela se fera plus tard.

Pour travailler sur une palette la plus large possible, les programmes mettront en avant :

- programmes 1 et 3 : bug dans les déplacements (trop longs, trop courts...)
- programmes 2 et 3 : bug dans l'utilisation des commandes (blocs)




Les élèves essaient de comprendre ce que le programme va faire et émettent des hypothèses, qui sont consignées dans la fiche 2 - partie 2. Elles serviront de base de travail pour l'étape suivante.

Ce qui est intéressant ici, c'est la confrontation entre élèves. Certaines et certains connaîtront peut-être des blocs de ScratchJr et verront déjà que les programmes donnés ne permettent pas de réaliser la tâche pensée au départ. D'autres ne remettront pas en cause ce qui est écrit et verront dans les programmes la confirmation de l'hypothèse émise au départ. Ce temps est vraiment propice aux échanges et prépare bien à l'étape suivante.

## Temps 1.2: Réaliser les programmes (à l'aide d'une tablette)

Modalités de travail : en binômes

 **Durée: 15 minutes**

 Le programme 3 nécessite la création de la balle, du rond d'arrivée et des murs. On trouvera sur l'application ScratchJr une rubrique d'aide intitulée **Manuel de l'éditeur graphique** qui donne de nombreuses explications.

Dans ce second temps, les élèves disposent des tablettes, sur lesquelles ils doivent refaire les programmes et les lancer pour observer leurs effets. On leur demande de réaliser les programmes des étiquettes collées sur la fiche 2 et d'observer ce qui se passe. Cette fiche est à conserver et servira de trace écrite.

Pour chaque programme, il y a une erreur :

- programme 1 : le ballon n'arrive pas jusqu'au but
- programme 2 : la fusée grossit énormément et ne se pose pas sur la Lune
- programme 3 : la balle n'atteint pas le cercle et a un comportement inattendu.

Lors de ce moment, on insiste sur le fait qu'il ne faut pas modifier les programmes donnés. En effet, les élèves seront peut-être tentés de faire en sorte que le programme corresponde à l'hypothèse émise lors du temps précédent, soit en rapprochant le but ou le ballon, soit en modifiant les murs du labyrinthe dans l'arrière-plan par exemple. Il faudra donc être vigilant et expliquer aux élèves qu'avant de corriger les programmes, on va chercher les causes du problème.

Les élèves mettent ensuite en parallèle les résultats obtenus avec les hypothèses formulées lors de l'étape précédente. Elles et ils constatent des divergences et cherchent à comprendre d'où elles peuvent provenir.

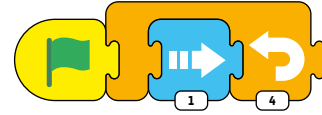
- programme 1 : pourquoi le ballon s'arrête-t-il avant le but ?
- programme 2 : pourquoi la fusée devient-elle si grande et ne se pose-t-elle pas sur la Lune ?
- programme 3 : d'où provient le comportement erratique de la balle ? Pourquoi n'atteint-elle pas le cercle ?

## Temps 1.3: Introduction de la notion de bug

Modalités de travail: en collectif

 **Durée:** 15 minutes

Ce temps a pour objectif de mettre en commun ce que les élèves auront trouvé et leurs éventuelles remarques quant aux écarts trouvés avec leurs hypothèses. En s'appuyant sur la fiche 2 (hypothèses) et sur les tablettes, on met en avant les différences observées. Le débat est lancé: pourquoi n'obtient-on pas ce que l'on attendait?



Si l'on écarte rapidement la volonté délibérée de la programmeuse ou du programmeur de ne pas faire rentrer le ballon dans le but (programme 1), il faut chercher ailleurs... Qu'est-ce qui guide le ballon? C'est le programme qui lui est associé. Donc, si le ballon ne va pas jusqu'au but, c'est qu'il n'en a pas reçu l'ordre. Il faut donc observer le programme de près et voir ce qui pose problème.

On demande aux élèves s'ils savent comment on appelle une erreur qui empêche l'exécution correcte d'un programme informatique. On appelle ça un bug (ou bogue).

Si on le souhaite et sans rentrer dans les détails, il est possible de faire un peu d'histoire de l'informatique avec les élèves en révélant les origines du mot bug (voir rubrique **Compléments – Prolongements – Variantes** en fin de séquence).

À ce niveau, les élèves auront remarqué que le problème venait de l'écriture du programme, qui contenait des erreurs: mauvais blocs utilisés, mauvaises instructions données. Il est important de rappeler que l'ordinateur ne fait qu'exécuter sans réfléchir les instructions qu'il reçoit. Un lien intéressant peut être fait avec d'autres situations connues des élèves (le jeu du robot par exemple).

## Étape 2

# Éléments pour répondre

### Résumé:

- Étudier avec méthode les programmes donnés afin de trouver l'erreur.
- Lister les types d'erreurs.
- Débattre et discuter en binôme pour chercher des solutions afin de corriger les programmes.
- Proposer aux élèves un outil permettant de corriger les programmes à partir de la catégorisation des blocs de ScratchJr.

### Matériel:

- fiche 2 (distribuée lors de l'étape 1)
- fiches 4 et 6 (1 par élève)
- fiche 5 (à projeter)
- tablette/application ScratchJr (1 par groupe)
- affichage numérique

## Temps 2.1: Repérer les erreurs ou les causes d'erreurs des programmes

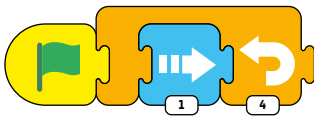
Modalités de travail: en binômes


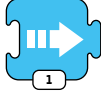

 Durée: 20 minutes

Les élèves reprennent la fiche 2 et cherchent d'où pourrait venir l'erreur dans chaque programme. Pour cela, elles et ils décomposent chaque programme bloc par bloc en les verbalisant et en s'aidant du référentiel (voir fiche 4). Les élèves comprennent ainsi ce que chaque instruction signifie. Pour ce travail, elles et ils s'appuient également sur la fiche 2 et sur les programmes réalisés sur les tablettes.

Une correction collective sous forme d'une mise en commun peut être proposée au fur et à mesure des programmes traités. Pour soutenir cette correction, les programmes que les élèves ont découpés et collés peuvent être projetés (voir fiche 5).

### Programme 1



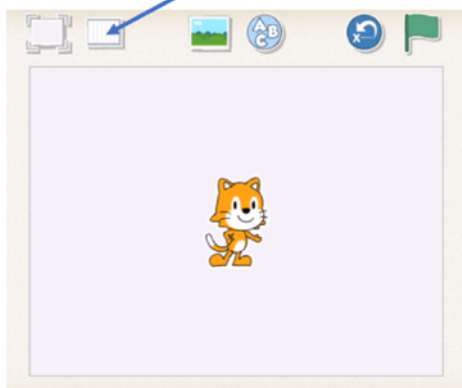
		
Ce bloc lance le programme.	L'objet avance de 1 vers la droite.	L'opération avancer de 1 vers la droite est répétée 4 fois.

Cette étape permet de visualiser où se situe le bug. Si le ballon n'arrive pas dans le but, c'est que la distance qu'on lui demande de parcourir n'est pas suffisante. Deux solutions:

- au lieu de le déplacer de 1 vers la droite, on augmente la valeur attribuée au déplacement.
- au lieu de répéter l'opération 4 fois, on la répète plus de fois, et on teste.

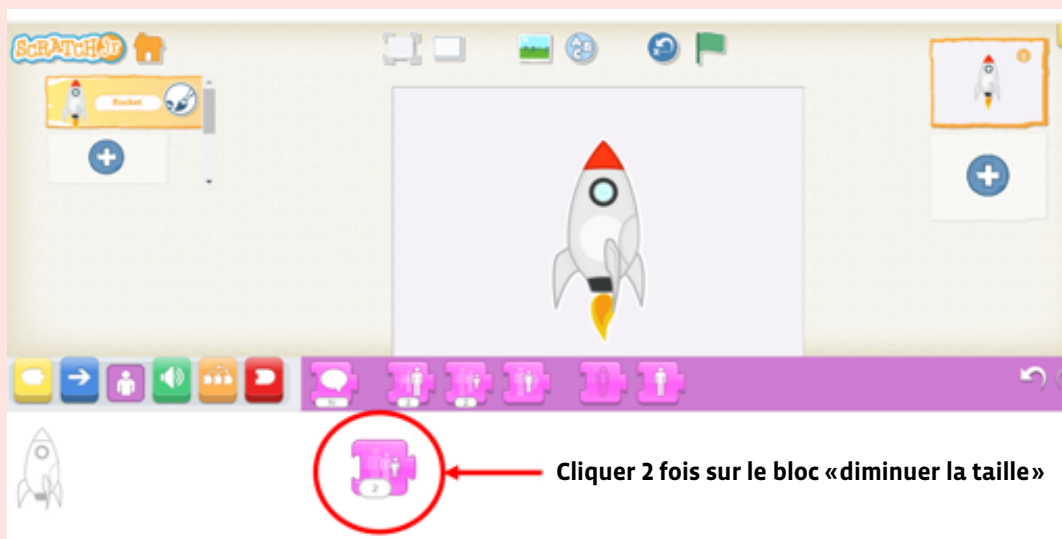
Cette notion de test est également importante et souvent liée à la notion de bug. On teste toujours un programme afin de vérifier que tout va bien. Dans le cas présent, il faut adapter le déplacement à la distance. Les élèves peuvent bien sûr procéder par tests successifs, essais-erreurs, mais il existe également une autre solution. Faire apparaître la grille permet de savoir directement de combien de **cases** il va falloir faire avancer le ballon, sans passer par des essais-erreurs.

En cliquant ici, on active la grille



## Programme 2

🔍 Pour préparer ce programme 2, on doit diminuer la taille de l'objet fusée. La démarche est la suivante:



Ce bloc lance le programme.	Ce bloc augmente la taille de l'objet (ou du personnage).	Ce bloc fait descendre l'objet de 1.	L'opération descendre l'objet de 1 vers la droite est répétée 4 fois.



En passant en revue le programme, les élèves se rendent compte qu'un bloc n'a pas sa place dans le programme: celle qui augmente la taille de la fusée.

Les corrections à apporter sont donc les suivantes:

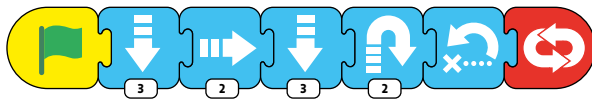
- supprimer ce bloc



- il faut ensuite vérifier si le fait de répéter 4 fois la descente de 1 permet de se poser sur la Lune. Sinon, il faut adapter la distance.

Dans ce programme, les élèves se rendent compte que le bug provient de l'utilisation erronée d'un bloc, qui ne correspond pas à ce que l'on attend. Et encore une fois, il est nécessaire de tester pour vérifier et adapter le programme.

### Programme 3



Ce bloc lance le programme.	Ces blocs codent les déplacements: 3 vers le bas, 2 vers la droite, 3 vers le bas.	Ce bloc fait sauter l'objet 2 fois.	Ce bloc fait revenir l'objet à sa position initiale.

Dans ce programme, il y a plusieurs soucis:

- il faut adapter les déplacements afin qu'ils correspondent au décor donné.
- des blocs ne correspondent pas à ce qui est demandé.

Les élèves suppriment les blocs inutiles (saut et retour au départ) et testent les distances afin que la balle se déplace à l'intérieur du décor et termine sa course dans le rond blanc.

En fonction du travail des élèves, prévoir une mise en commun: 3 groupes présentent la version corrigée des programmes et verbalisent les bugs et correction(s) réalisées par exemple.

## Temps 2.2: Mise en forme d'un outil de traque des erreurs

Modalités de travail: en collectif

Durée: 15 minutes

Les élèves sont maintenant en mesure de voir ce qu'il faut faire lorsqu'on est confronté à un programme qui bugue. Il faut observer méthodiquement chaque bloc, comprendre quelle est l'instruction derrière en verbalisant l'algorithme puis en vérifiant qu'elle correspond bien à ce qu'on attend ou ce qui est demandé. Dès lors, pour être sûr de ne rien oublier, il faut établir une «check-list», à la manière des pilotes avant le décollage. Il faut également insister sur le sens de lecture du programme: partir de la gauche pour éviter un repérage global qui omet la construction même du programme proposé (programmation séquentielle). Pour construire cette liste, on va s'appuyer sur le référentiel des blocs de ScratchJr (voir fiche 4).

Chaque «**moment**» d'un programme est pris en compte:

- lancement du programme
- déplacements des objets
- contrôle (répétitions)
- fin d'un programme



Afin de ne pas alourdir le travail sur le bug dans cette enquête, ne sont traités ici avec les élèves que les cas relevant des situations proposées (déplacement, blocs non conformes). On laisse cependant la porte ouverte aux autres situations:

- le déclenchement des actions
- l'apparence des objets
- les sons

Les élèves ont donc à disposition la fiche 4 (référentiel des blocs ScratchJr) et la fiche 6 (outil de traque des bugs dans un programme). On suggérera aux élèves d'annoter cette fiche 6 à l'aide d'un crayon afin de pouvoir la réutiliser avec d'autres programmes.

## Étape 3

### Éléments pour conclure (validation, mise en forme)

#### Résumé:

- Utiliser un outil qui chasse les bugs basé sur la méthode validée préalablement pour vérifier des programmes donnés.
- Rédiger une trace écrite (affiche-outil pour la classe).

#### Matériel:

- fiche 4 (distribuée lors de l'étape 2)
- fiche 6 (1 par élève)
- fiche 7 (1 par groupe ou par élève)
















### Temps 3.1: Validation des outils










Modalités de travail: en collectif, puis en groupes (2-3 élèves)

 Durée: 15 minutes

Ce temps commence par un moment en **débranché**. Les tablettes ne sont pas disponibles. La fiche 7 est distribuée aux élèves. Elle contient un programme à vérifier. Les élèves appliquent la méthodologie vue jusqu'ici. Le titre du programme (**Départ de la Lune**) crée un horizon d'attente. On s'attend à ce que le personnage monte dans la fusée et décolle. Pour s'assurer que tous les élèves de la classe s'accordent sur ce qui est attendu, on fera le point avec eux avant de les lancer à la recherche des erreurs.

À l'aide des fiches 4 et 6, les élèves vérifient méthodiquement le programme. On explique que ces fiches permettent de passer en revue les blocs utilisés dans le programme et de vérifier qu'ils correspondent à ce que l'on a prévu de faire. Le résultat est le suivant:

Lancement du programme	Déplacement des objets	Contrôle
 Drapeau vert <input checked="" type="checkbox"/>	 Droite <input checked="" type="checkbox"/>	 Attendre <input checked="" type="checkbox"/>
 Toucher <input type="checkbox"/>	 Gauche <input type="checkbox"/>	 Arrêt <input type="checkbox"/>
 Contact <input checked="" type="checkbox"/>	 Haut <input checked="" type="checkbox"/>	 Vitesse <input type="checkbox"/>
 Message reçu <input type="checkbox"/>	 Bas <input type="checkbox"/>	 Répéter <input type="checkbox"/>
 Message envoyé <input type="checkbox"/>	 Tourner vers la droite <input type="checkbox"/>	
	 Tourner vers la gauche <input type="checkbox"/>	

Fin		Apparence		Sons	
	Fin	<input type="checkbox"/>		Droite	<input type="checkbox"/>
	Toujours répéter	<input type="checkbox"/>		Gauche	<input type="checkbox"/>
	Aller à la page	<input type="checkbox"/>		Haut	<input type="checkbox"/>
				Bas	<input type="checkbox"/>
				Tourner vers la droite	<input checked="" type="checkbox"/>
				Tourner vers la gauche	<input type="checkbox"/>

Le problème est le suivant: le personnage ne «monte» pas dans la fusée. Autrement dit, il ne disparaît pas de l'écran. Le bug se situe au niveau de l'oubli d'un bloc. Les élèves peuvent facilement corriger le programme, le «débuguer» pour qu'il corresponde à ce qui est attendu. À cette fin, on leur distribue les tablettes. Les élèves passent ainsi en mode «test» et procèdent alors à différents essais pour que le programme fonctionne correctement.

## Temps 3.2: Institutionnalisation

Modalités de travail: en groupes (2-3 élèves)

 **Durée: 15 minutes**

Afin d'aider les élèves dans leur traque des bugs, on pourra afficher dans la classe le document de «traque», qui reprend tous les blocs de ScratchJr, et qui permet de réaliser une «check list» pour tous les programmes.

Pour prolonger ce travail, on propose aux élèves, par groupes de 2 ou 3 élèves, de créer un programme, d'y introduire un bug, et de demander à un autre groupe de le débbuger.

## Compléments – Prolongements – Variantes

Afin de confronter les élèves à d'autres bugs de programmes, on pourra traiter les 2 situations ci-dessous de la manière suivante :

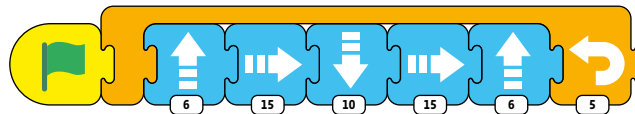
- On commence par projeter uniquement le point de départ et on demande aux élèves de formuler des hypothèses quant à ce qu'il va se passer.
- On affiche ensuite le programme et les élèves tentent de voir s'il correspond à l'hypothèse retenue.
- On teste effectivement le programme donné (il faut le reproduire sur la tablette). On constate alors que ça ne fonctionne pas.
- On procède à l'analyse pas à pas pour chercher ce qui bloque.
- On corrige le programme (la correction est donnée sur la fiche).

### Situation A



**Hypothèses des élèves:** Scratch doit courir autour du terrain, traverser le terrain, faire un parcours...




**Programme proposé aux élèves:**



**Hypothèses à la lecture du programme:** Scratch fait le tour du terrain plusieurs fois.

**Constat lors de la réalisation effective du programme:** Scratch ne court pas complètement autour du terrain. Il n'en fait pas le tour. Où est le bug ?

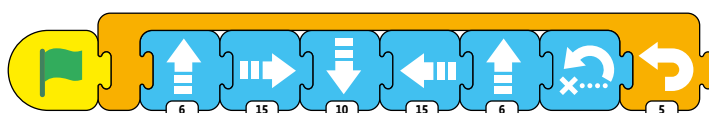
Programme décomposé :

		
<p>Ce bloc lance le programme.</p>	<p>Ces blocs codent les déplacements : 6 vers le haut, 15 vers la droite, 10 vers le bas, 15 vers la droite, 6 vers le haut.</p>	<p>Le déplacement précédent est répété 5 fois.</p>

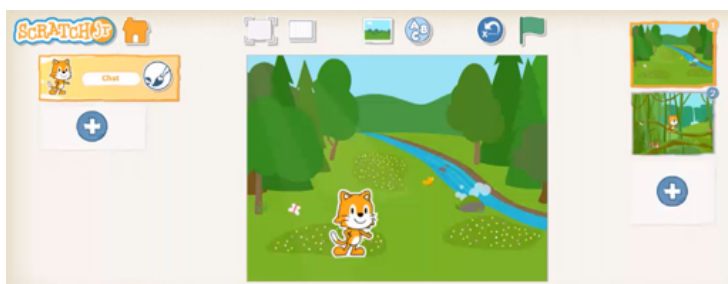
**Les origines du bug:**

- Le déplacement n'est pas correct (2 déplacements vers la droite). Il faut remplacer le 2<sup>e</sup> bloc **15 vers la droite** par un **15 vers la gauche**.
- Mais même là, les élèves, en testant, vont se rendre compte que le programme ne fonctionne pas. Scratch ne se déplace pas autour du terrain. Il manque un bloc : **retour à la position de départ**.

Cela donne le programme corrigé que l'on retrouve ci-dessous :



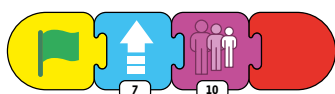
## Situation B



### Hypothèses des élèves:

Scratch va se promener, se jeter à l'eau, aller dans le nouveau paysage (présence de 2 décors)...

### Programme proposé aux élèves:



**Hypothèses à la lecture du programme:** Scratch se dirige vers le haut, sa taille diminue et il change de décor.

**Constat lors de la réalisation effective du programme:** Scratch se dirige effectivement vers le haut, comme s'il s'éloignait. Mais sa taille diminue d'un coup, et il ne bascule pas dans le nouveau décor.

### Programme décomposé:

Ce bloc lance le programme.	Ce bloc code le déplacement: 7 vers le haut.	Ce bloc code une diminution de taille de 10.	Ce bloc code la fin du programme.

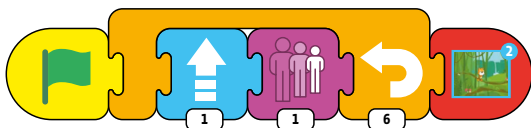
### Les origines du bug:

- Les élèves se rendent compte que la diminution de taille ne peut pas se faire d'un coup à la fin du déplacement. Ils doivent chercher, tester, essayer pour trouver une solution qui donne l'impression que Scratch devient de plus en plus petit au fur et à mesure qu'il s'éloigne. Pour comprendre le monde numérique qui nous entoure, l'élève doit d'abord identifier les phénomènes observés (ici le rapetissement). Ensuite, l'élève modélise comment reproduire numériquement ce phénomène. Enfin, il traite cela en créant le programme qui «donne l'impression» de reproduire ce phénomène.
- Une solution consiste à utiliser le bloc **répéter** de la manière suivante:



- Pour la fin du programme, il faut utiliser le bloc **aller à la plage 2**.

Cela donne le programme suivant:



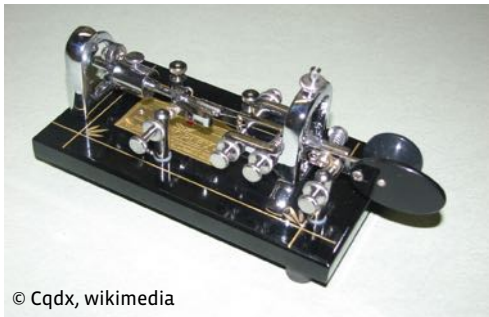
## Petit historique du bug en informatique

La définition du mot bug (ou bogue) est la suivante: défaut de conception ou de réalisation d'un programme informatique, qui se manifeste par des anomalies de fonctionnement de l'ordinateur. Le mot bug a largement dépassé les frontières de l'informatique, puisqu'on l'emploie aujourd'hui couramment comme synonyme de problème.

Mais pourquoi ce mot qui signifie *insecte* en anglais en est-il venu à être synonyme de problème informatique? Il existe deux anecdotes différentes à ce sujet. La première remonte bien avant l'ère informatique, au XIX<sup>e</sup> siècle pour trouver trace du mot, et plus particulièrement au début du télégraphe.

## Bug et télégraphie

Le mot serait né en lien avec des problèmes techniques sur des circuits de télégraphes, provoquant des faux signaux sur les circuits des premiers appareils. Ce terme faisait à ce point partie de la vie des télégraphistes qu'un inventeur et fabricant, Horace G.



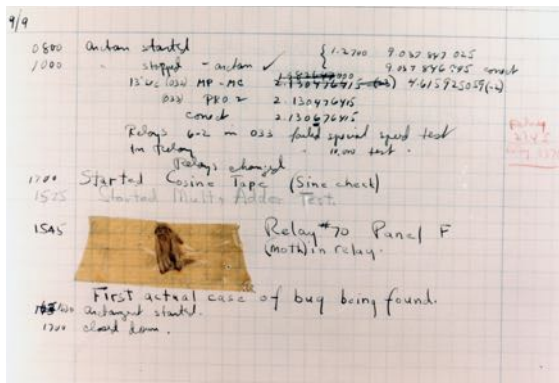
© Cqdx, wikimedia

Martin, va même utiliser l'image d'un insecte (bug en anglais), le scarabée, sur une de ses inventions, qu'il nommera même le **Vibroplex le bug**.

Il s'agit d'un télégraphe amélioré qui révolutionne en 1903 la manière dont le code Morse est transmis. Ces inventions ont affecté les carrières des milieux des télégraphes et

de la radio. Le terme **bug** sera popularisé par Edison, qui appelait **bug** les problèmes techniques qu'il rencontrait lors du développement du télégraphe quadruplex (appareil pouvant envoyer et recevoir jusqu'à quatre télégrammes séparés à la fois).

## Grace Hopper



Une autre origine de l'utilisation du mot bug en informatique viendrait de l'anecdote suivante (ce n'est peut-être qu'une légende). En 1947, un papillon de nuit se bloque dans le calculateur Mark II de l'université Harvard aux Etats-Unis, qu'utilisait Grace Hopper, engendrant des problèmes de fonctionnement. Cette pionnière américaine de l'informatique, mobilisée comme auxiliaire dans la marine américaine, fut affectée aux travaux de programmation et d'exploitation de l'ENIAC (Electronic Numerical Integrator And Computer:



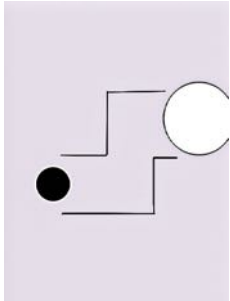
premier ordinateur entièrement électronique). Elle est également la principale créatrice du COBOL (Common Business Oriented Language: langage de programmation créé en 1959). Quand elle trouve cet insecte, elle le scotche dans son carnet de bord.

Le carnet de bord de l'ordinateur, qui se trouve actuellement au Musée national d'histoire américaine, garde la trace de ce petit incident. À la date du 9 septembre 1947, on peut voir, collé sur la page, le cadavre d'un insecte qui avait volé dans un commutateur et s'était retrouvé coincé, a relaté Grace Hopper, citée dans l'ouvrage d'Isabelle Collet. Sur la page, sous l'insecte, on distingue une note rédigée à la main, mentionnant ainsi le premier cas de bug recensé.





## Trois situations

Situations	Partie 1 : A ton avis, que va-t-il se passer ?	Partie 2 : Après avoir lu les programmes, que va-t-il se passer ?
	<p>↑</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p>	<p>Coller programme 1</p> <p>↑</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p>
	<p>↑</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p>	<p>Coller programme 2</p> <p>↑</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p>
	<p>↑</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p>	<p>Coller programme 3</p> <p>↑</p> <p>_____</p> <p>_____</p> <p>_____</p> <p>_____</p>

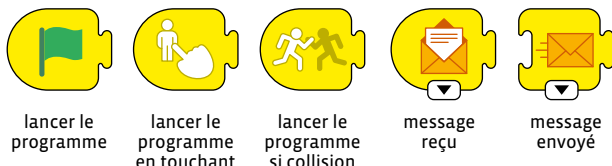
Fiche 3

## Programmes à découper pour la fiche 2

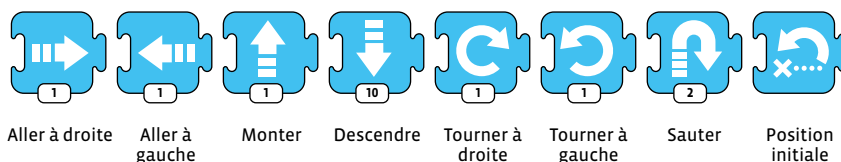
The image displays three sets of interlocking puzzle pieces, each set labeled 'Programme 1', 'Programme 2', and 'Programme 3'. The pieces are arranged in a 3x3 grid. Each set consists of four pieces: a yellow piece with a green flag (1), a blue piece with a white arrow (1), a purple piece with a white arrow (2), and a yellow piece with a white arrow (4). The pieces are numbered 1, 2, 3, and 4. A red dashed border surrounds the pieces, and a red scissors icon is in the top right corner.

## Les blocs de ScratchJr

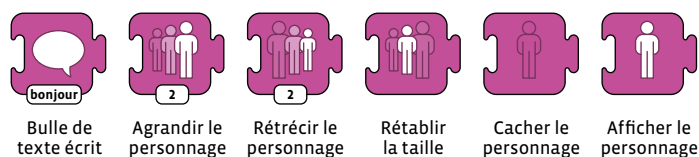
### Les briques de déclenchement des programmes



### Les briques de mouvement des personnages



### Les briques d'apparence des personnages



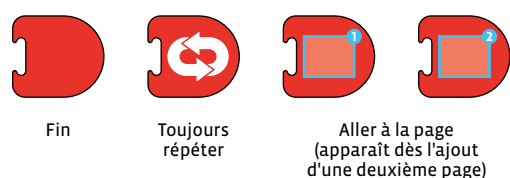
### Les briques pour les sons



### Les briques pour le contrôle



### Les briques de fin de programmes

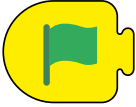
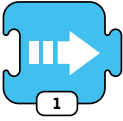
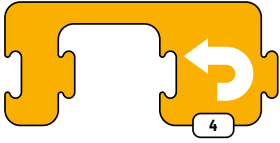


# Fiche 5



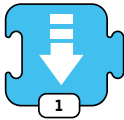
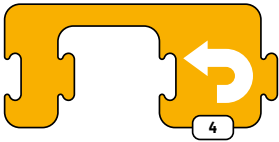
## À projeter

### Programmes décomposés

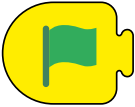
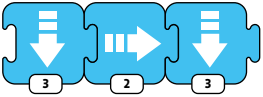


#### Programme 1

		
Ce bloc lance le programme.	L'objet avance de 1 vers la droite.	L'opération avancer de 1 vers la droite est répétée 4 fois.

#### Programme 2

			
Ce bloc lance le programme.	Ce bloc augmente la taille de l'objet.	Ce bloc fait descendre l'objet de 1.	L'opération descendre de 1 est répétée 4 fois.


#### Programme 3

			
Ce bloc lance le programme.	Ces blocs codent les déplacements: 3 vers le bas, 2 vers la droite, 3 vers le bas.	Ce bloc fait sauter le personnage 2 fois.	Ce bloc fait revenir le personnage à sa position initiale.

**Fiche 6**

Prénom : .....

## Outil de traque des bugs dans un programme

<p><b>Lancement du programme</b></p> <p> Drapeau vert <input type="checkbox"/></p> <p> Toucher <input type="checkbox"/></p> <p> Contact <input type="checkbox"/></p> <p> Message reçu <input type="checkbox"/></p> <p> Message envoyé <input type="checkbox"/></p>	<p><b>Déplacement des objets</b></p> <p> Droite <input type="checkbox"/></p> <p> Gauche <input type="checkbox"/></p> <p> Haut <input type="checkbox"/></p> <p> Bas <input type="checkbox"/></p> <p> Tourner vers la droite <input type="checkbox"/></p> <p> Tourner vers la gauche <input type="checkbox"/></p>	<p><b>Contrôle</b></p> <p> Attendre <input type="checkbox"/></p> <p> Arrêt <input type="checkbox"/></p> <p> Vitesse <input type="checkbox"/></p> <p> Répéter <input type="checkbox"/></p>
<p><b>Fin</b></p> <p> Fin <input type="checkbox"/></p> <p> Toujours répéter <input type="checkbox"/></p> <p> Aller à la page <input type="checkbox"/></p>	<p><b>Apparence</b></p> <p> Droite <input type="checkbox"/></p> <p> Gauche <input type="checkbox"/></p> <p> Haut <input type="checkbox"/></p> <p> Bas <input type="checkbox"/></p> <p> Tourner vers la droite <input type="checkbox"/></p> <p> Tourner vers la gauche <input type="checkbox"/></p>	<p><b>Sons</b></p> <p> Droite <input type="checkbox"/></p> <p> Gauche <input type="checkbox"/></p>

Fiche 7

Prénom: .....

Programme à tester : départ de la Lune

