GROUPE INFORMATIQUE DE L'ÉCOLE JUSQU'AU LYCÉE
IREMI DE GRENOBLE
INSTITUT DE RECHERCHE SUR L'ENSEIGNEMENT DES MATHÉMATIQUES ET DE L'INFORMATIQUE DE GRENOBLE

ACTIVITÉ INFORMATIQUE DEBRANCHEE

– CARGO-BOT –

Cycle 3 et 4

Objectifs de l'activité
- S'approprier les instructions d'un langage symbolique contraint
- Formaliser rigoureusement la solution d'un problème
- Utiliser des procédures et/ou la récursivité pour concevoir des programmes complexes
- Écrire, notamment à l'aide d'instructions conditionnelles, un programme qui répond à plusieurs problèmes d'une même classe de

Prérequis : Aucun

Durée de l'activité : une ou plusieurs séances de 45 min à 1h

Modalités de mise en œuvre de l'activité :

Classe entière ou demi-classe

Travail en groupes de 3 ou 4 élèves

Mise en commun et temps de discussion

Trace écrite – Institutionnalisation

Matériel :

Planches de travail à imprimer + étiquettes à découper
Fiches problèmes à imprimer
Gobelets de couleur
Facultatif : vidéoprojecteur

COMPETENCES INFORMATIQUES

Anticiper
- Concevoir un algorithme répondant à un problème de déplacements
- Utiliser les structures de contrôle d'un langage de programmation pour définir le déroulement d'un programme
- Corriger les erreurs dans un programme ne respectant pas la spécification demandée

Décomposer
- Écrire des procédures
- Utiliser des procédures pour construire un programme en plusieurs parties

Évaluer
- Tester un programme sur des exemples simples
- Comparer deux algorithmes répondant à un problème selon certains critères : domaine de validité, efficacité, lisibilité.

Généraliser
- Utiliser un algorithme unique pour résoudre plusieurs problèmes partageant les mêmes caractéristiques.

PHASE 1 : PROGRAMMER AVEC EXÉCUTION DIFFÉRÉE

20 min	Groupe classe puis trinômes	Premiers programmes	Une planche et un jeu d'étiquettes par groupe Quelques gobelets de même couleur par groupe Fiches problème DÉMO, TRANSPORTEUR, EMPILEMENT Cahier ou papier libre

OBJECTIFS POUR L'ÉLÈVE

Prévoir l'effet de chacune des instructions basiques du langage (, ,) utilisées dans différentes situations.
Construire un modèle mental d'exécution des programmes séquentiels.
Mettre une solution à l'épreuve du test.

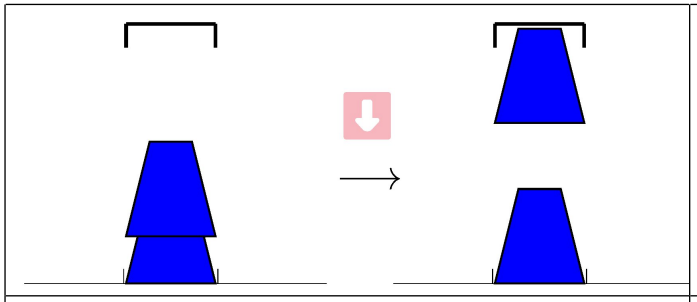
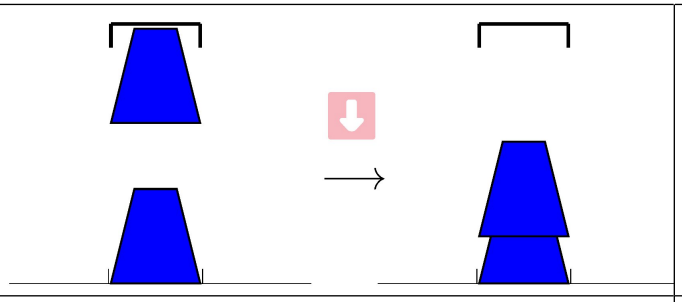
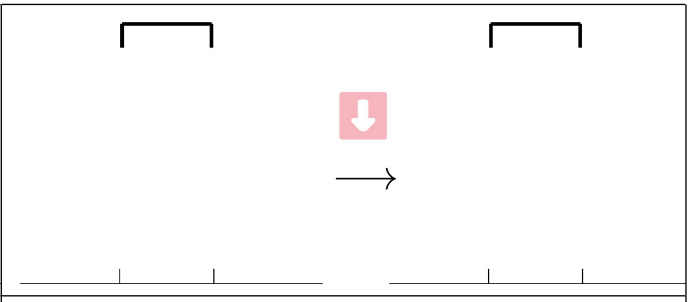
DÉROULEMENT

GROUPE CLASSE

L'enseignant commence par **présenter le matériel et son utilisation** : les gobelets représentent des objets à déplacer, et les déplacements seront effectués au moyen d'une « grue mécanique » dont le rôle sera joué par un élève.

Quelques règles s'appliquent : la grue ne peut transporter qu'un gobelet à la fois, et déplacer la grue hors du terrain de jeu constitue une erreur.

Attention, la même instruction est utilisée pour ramasser ou poser un gobelet :

		
Si la grue est vide et au-dessus d'un gobelet, elle le ramasse et remonte.	Si la grue contient un gobelet, elle le dépose et remonte à vide.	Si la grue est vide et qu'il n'y a rien en dessous, rien ne se passe.

Important : les instructions ne sont pas données au fur et à mesure à la grue. On écrit un programme en positionnant des instructions dans la grille, qui sera exécuté dans un second temps sur le terrain d'expérimentation.

Chaque fiche problème propose **une situation de départ** (emplacement des gobelets et de la grue) et la **position finale** attendue ; l'exécution du programme s'arrête dès l'instant où la position finale est atteinte.

L'exemple **DÉMO** est soumis à la classe entière (éventuellement au vidéoprojecteur), pour en mettre au point ensemble la solution et discuter des propositions incorrectes (compréhension erronée des consignes, instructions oubliées...).

Les élèves peuvent noter les instructions utilisées de façon à pouvoir comparer ou ré-exécuter les programmes plus tard.

EN GROUPE

L'enseignant distribue à chaque groupe des gobelets, d'une planche et d'un jeu d'instructions. **Une moitié des groupes** reçoit la fiche problème **TRANSPORTEUR** et l'autre la fiche **EMPILEMENT.**

Chaque groupe d'élèves cherche à écrire un programme qui résout le problème proposé. Une fois qu'ils pensent le problème résolu, ils désignent un « élève-grue » pour tester leur programme parmi un des groupes qui a cherché l'autre problème. Si un problème apparaît en cours d'exécution, il est possible de faire stopper la grue, lui demander de s'éloigner à nouveau pour corriger l'erreur, puis le faire revenir et tester à nouveau.

Quelles sont les conditions qui permettent la vérification effective d'un programme ?

La personne qui exécute le programme doit le faire en respectant scrupuleusement ce qui est écrit, sans ajouter ni supprimer d'instructions implicitement. Elle doit procéder comme si elle ne connaissait pas l'objectif à atteindre.

Conseils pour la mise en œuvre

Il est important que la grue n'ait pas connaissance de la situation d'arrivée du groupe dont il teste le programme, ni pendant la phase de recherche ni pendant le test proprement dit (le reste du groupe ne lui donne pas d'autre instruction que celles formant le programme écrit).

PHASE 2 : ÉCRITURE DE PROGRAMMES SÉQUENTIELS, DÉCOUVERTE DES PROCÉDURES

15-25 min	Trinômes	Programmes produits par les élèves Procédures	Étiquettes F1, F2 et F3 Fiches problème SANDWICH, ÉCHANGE, ÉCHANGE VERTICAL, MISE À PLAT, MULTITRANSPORTEUR

OBJECTIFS POUR L'ÉLÈVE

Utiliser des procédures pour construire un programme en plusieurs parties
Écrire des programmes structurés

DÉROULEMENT

On passe maintenant à une phase de résolution de problèmes par groupe et en autonomie. Chaque groupe prend maintenant en charge la vérification de ses programmes.

À vous de jouer ! Voici plusieurs problèmes à résoudre, de difficulté croissante. Vérifiez que vos programmes sont corrects et notez-les sur papier libre avant de passer au suivant.

À noter : Il n'est pas possible de résoudre le problème SANDWICH avec un programme qui tienne sur la ligne F0, en effet plus de 12 instructions sont nécessaires.

Quand on a besoin de plus de cases ou d'étiquettes que disponibles pour écrire un programme, comment peut-on procéder ?
Y a-t-il des séquences d'instructions qui se répètent dans vos programmes ?

Une première option est simplement de revenir à la ligne mais elle est vite limitée, et en principe interdite par les règles.

On introduit l'**appel** de procédures avec les étiquettes associées : F1, F2, F3.

Il devient ainsi possible de nommer une séquence d'instructions pour l'appeler à l'intérieur du programme principal F0.

*Une **procédure** est une séquence d'instructions que l'on isole et à laquelle on donne un **nom**. Il est possible de **faire appel** à la procédure (l'exécuter) depuis un autre programme en remplaçant la séquence d'instructions par le nom.*

Dans l'activité cargo bot, on appelle une procédure en plaçant l'étiquette qui lui correspond (par exemple F1).

Existe-t-il différentes solutions pour un même problème ?
Sont-elles toutes équivalentes, ou certaines sont-elles « meilleures » que d'autres ?
Et sur quels critères ? le plus court à écrire, le plus court à exécuter, le plus lisible...

Quand on écrit un programme on essaye d'isoler les parties redondantes dans des procédures, mais pour autant un « bon » programme n'est pas forcément celui qui utilise le moins d'instructions, il vaut mieux que le service rendu par chaque procédure (sa fonctionnalité) soit facilement identifiable.

Conseils pour la mise en œuvre

Les fiches problèmes peuvent être distribuées au fur et à mesure de l'avancement de chaque groupe, ou bien on peut en donner plusieurs dès le début pour que chaque groupe avance à son rythme. Si l'enseignant constate qu'un groupe valide ses programmes malgré des erreurs, il peut jouer ponctuellement le rôle de la grue pour leur rappeler l'importance d'exécuter les programmes littéralement.

Le problème Multitransporteur permet d'aborder la notion de boucle imbriquée.

PHASE 3 : RÉCURSIVITÉ

15-25 min	Trinôme	Programmes produits par les élèves	Étiquettes F0 Fiches problème TRANSPORTEUR LOURD, DICHOTOMIE, INVERSION

OBJECTIFS POUR L'ÉLÈVE

Utiliser la récursivité pour programmer des répétitions
Écrire des solutions génériques, résolvant une classe de problèmes de même nature

DÉROULEMENT

Pour les prochains problèmes, il faut trouver une solution générale, qui s'adapte à plusieurs configurations de départ.
Par exemple, pour TRANSPORTEUR LOURD ou INVERSION, la pile de gobelets pourrait être de n'importe quelle taille.
N'oubliez pas que dès que les gobelets sont dans la situation finale voulue, la grue s'arrête !

Les élèves savent déjà répéter « manuellement » une procédure en l'appelant plusieurs fois. Le cap à franchir ici est de « boucler » sur un même groupe d'instructions, ce qui est possible à condition d'appeler une procédure à l'intérieur d'elle-même. On peut distribuer les étiquettes F0 soit après ce constat pour écrire certains programmes en une seule ligne, soir pour donner l'idée de cet appel d'une procédure par elle-même.

On dit qu'une procédure est récursive si on appelle la procédure à l'intérieur d'elle-même. Cela permet de répéter indéfiniment le contenu de cette procédure.
Par contre, attention ! Une telle procédure va s'exécuter indéfiniment si on n'a pas un moyen de l'arrêter.

Conseils pour la mise en œuvre

Le problème MULTITRANSPORTEUR peut aussi donner lieu à une solution récursive mais il ne sera pas possible d'être complètement générique, on doit connaître la taille des piles pour savoir quand passer à la suivante, on pourra seulement s'adapter à un nombre de piles variable. Il peut aussi servir de motivation pour la prochaine phase : l'introduction de conditions pour savoir quand sortir d'une procédure.

Dans le cas où des élèves cherchent à écrire systématiquement les programmes les plus courts possibles, on peut revenir sur le premier problème proposé (TRANSPORTEUR) et faire remarquer qu'il existe une solution récursive plus courte (↓ → ↓ F0) que la solution « évidente » que les élèves ont pu trouver auparavant. Cependant, cette solution est beaucoup plus longue à s'exécuter car la grue effectue de nombreux mouvements supplémentaires. On peut alors prolonger la discussion de la phase 2 à propos des différentes mesures possibles de la qualité d'un programme : nombre d'instructions utilisées, temps d'exécution, lisibilité, généricité.

PHASE 4 : PROGRAMMATION AVEC DES CONDITIONS

10-30 min	Trinômes	Programmes produits par les élèves	Étiquettes conditionnelles Fiches problème INTRUS, EXTRACTION, CHROMATOMIE, TRANSLATION, RAMASSEUR, À BAS LES BLEUS

OBJECTIFS POUR L'ÉLÈVE

Utiliser les conditions pour écrire des programmes qui s'adaptent à la configuration initiale du terrain
Renforcer encore la généricité des solutions écrites, prévoir un cas d'arrêt propre pour les programmes écrits.

DÉROULEMENT

On a vu dans les problèmes précédents que la grue pouvait exécuter des instructions à l'infini avec la récursivité.
On ajoute un dernier type d'instructions dans nos programmes pour décider si on exécute ou non une instruction.

Les étiquettes conditionnelles  ,  ,  , etc. sont distribuées. Elles se placent au-dessus d'une autre instruction. Pour savoir si l'instruction s'exécute ou non, on regarde si *au moment d'exécuter cette instruction* le contenu de la grue correspond : respectivement si elle est vide, si elle est pleine, si elle contient un gobelet rouge, etc.

La fiche INTRUS est distribuée ou vidéoprojetée pour donner un exemple et montrer que le programme réagira différemment selon la disposition initiale des gobelets.

Les problèmes **EXTRACTION** et **CHROMATOMIE** utilisent les étiquettes de couleur, et les suivants (**TRANSLATION**, **RAMASSEUR**, **À BAS LES BLEUS**) utilisent les étiquettes « grue pleine » ou « grue vide » pour détecter quand on a fini de déplacer une pile.

Les instructions conditionnelles permettent de choisir en cours d'exécution si une partie du programme sera effectuée ou non. Mais la machine n'a toujours pas besoin de prendre de décision, on doit lui indiquer précisément quoi faire dans quel cas. On peut ainsi s'adapter à plusieurs situations similaires.

Pouvez-vous exploiter les instructions conditionnelles pour faire en sorte que les programmes précédents (TRANSPORTEUR LOURD ou INVERSION) s'arrêtent tous seuls lorsque le travail est fini ?

Il faut ici placer une condition sur l'appel récursif de procédure, de sorte qu'on s'arrête lorsqu'il n'y a plus de gobelet à déplacer. Attention, cela demande souvent de « préparer le terrain » : on doit prendre un gobelet dans la grue avant d'entrer dans la procédure, et s'il y a encore des gobelets à déplacer la grue doit en contenir un au moment de faire l'appel récursif.

Quand on écrit un programme, en général ce n'est pas pour résoudre un problème particulier mais plutôt un ensemble de problèmes similaires (on parle de « classe de problèmes »). Plus le programme est générique, plus on pourra le réutiliser sans avoir besoin d'en écrire un autre.

Ce qui est remarquable, c'est qu'avec uniquement ces idées très simples (l'enchaînement d'instructions, l'appel de procédure et les conditions) on peut écrire des programmes pour pratiquement n'importe quoi ! On les retrouve d'ailleurs dans tous les langages de programmation, de Scratch aux langages utilisés professionnellement.

Conseils pour la mise en œuvre

- Penser à rappeler régulièrement aux élèves qu'on cherche des solutions aussi génériques que possible : qui s'adaptent à la fois au nombre de gobelets, à une disposition différente des couleurs, etc.

SOURCES ET AUTEURS

Cargo-Bot est un jeu de Rui Vianna ; adaptation en JavaScript par Joe Tessler.

Certains des problèmes proposés sont librement inspirés des travaux du groupe Informatique Sans Ordinateur de l'IREM de Clermont-Ferrand.

Ont participé à la rédaction de ce document : Maryline Althuser, Anne Rasse, Jean-Marc Vincent, Benjamin Wack, Gaëlle Walgenwitz.

RESSOURCES POUR ALLER PLUS LOIN

https://www-verimag.imag.fr/~wack/CargoBot/

https://pinguee.github.io/cargo-bot-tangible/

Tangente Éducation n° 42-43 https://tangente-education.com/TE.php#te42

Articles de recherche en anglais :

https://www.researchgate.net/publication/262362202_Using_Cargo-Bot_to_Provide_Contextualized_Learning_of_Recursion

https://www.researchgate.net/publication/265087747_A_Structured_Approach_to_Teaching_Recursion_Using_Cargo-Bot