Algorithmique et programmation, un levier pour développer des compétences mathématiques

Saison 2

mai 2017

Document rédigé par
Véronique Bluteau-Davy, IA-IPR de mathématiques dans l'académie de Nantes
Yannick Danard, professeur au collège Clément Janequin - Avrillé
Stéphane Percot, professeur au collège Haxo - La Roche-sur-Yon et chargé de mission à la DAN – Rectorat de Nantes.

Avec la collaboration de
Yann Bertrand, professeur au lycée Aristide Briand - Saint Nazaire
Fabrice Foucher, professeur au lycée Jacques Prévert - Savenay
Julie Binet, professeur au collège René Guy Cadou - Ancenis
Laëtitia Charruau, professeur au collège François Truffaut – Longué Jumelles
Emmanuel Malgras, professeur au collège Pierre et Marie Curie - Le Pellerin
Grégory Maupu, professeur au collège Milcendeau – Challans
Damien Rivière, professeur au collège Pierre Dubois – Laval

SOMMAIRE

1) RAPPELS DE LA SAISON 1

a) Notre travail de l'an passé

Un groupe de sept professeurs de collège et de lycée de l'académie de Nantes avait travaillé, dans le cadre des actions mutualisées au niveau national par la DNE, au cours de l'année scolaire 2015-2016, sur l'enseignement de l'algorithmique et de la programmation au cycle 4. De nombreuses activités et un document de synthèse de cette première année de recherche avait été mis en ligne.

Voir : travail de recherche autour de l'algorithmique en collège - saison 1

b) Des idées pour démarrer l'algorithmique

Nos productions de l'an passé intègrent des pistes pour démarrer l'enseignement de l'algorithmique avec, entre autres, des outils en ligne simples pour introduire de façon très ludique tous les éléments nécessaires à la réalisation d'algorithmes (déplacements simples, boucles « pour », boucles « jusqu'à », test « si alors sinon… »).

Elles comportent également des petits exemples de programmes élaborés avec les élèves pour réaliser des figures simples, des déplacements de lutins, des phénomènes aléatoires simples.

Voir : pour démarrer l'algorithmique

c) Des exemples d'activités simples (ou pas), de scénarios pédagogiques

De nombreux exemples d'activités pédagogiques à mener avec les élèves ont été testés en classe (de la 6e à la 3e) et mis en ligne sur une page du site académique. D'activités simples à des projets plus complexes, ces exemples proposent un petit échantillon de possibilités à faire vivre en cycle 4.

Voir : des activités pour la classe et des projets à mener avec les élèves

Pour aller plus loin, une page est dédiée à des projets plus ambitieux ou des idées d'activités originales à mener avec Scratch.

Elle demande parfois un niveau de maîtrise de l'algorithmique et de l'outil Scratch lui-même un peu plus avancé.

Voir : des idées pour aller plus loin

d) Nos questionnements en suspens

Lors des travaux menés par le groupe sur l'année 2015-2016, des perspectives et des questionnements avaient conclu provisoirement notre travail de recherche :

1) ***Quels moments de régulation ?***
Lors d'une activité de programmation, un groupe plus ou moins important d'élèves peut rencontrer des blocages.
Comment anticiper ces blocages ?
Comment intervenir individuellement ? Collectivement ?
De façon générale, quelles sont les pistes pour gérer ces moments de régulation tant sur le plan de l'organisation que des contenus ?

2) ***Quelle communication autour d'un programme***
Lors de la réalisation d'un script simple, une erreur est assez facilement détectable. Dans le cas d'un programme plus élaboré, l'élève est amené à devoir présenter son travail, son organisation.
Quelle(s) stratégie(s) d'organisation, de programmation sont mises en œuvre ?
Quels choix ont été faits pour l'utilisation et le nom des variables ? Quel est le rôle de ces variables ?
Comment les différents éléments du script s'organisent-ils ?
Quelle place peut-on donner au travail partagé : décomposition d'un programme en sous-programmes réalisés par différents élèves et réunis ensuite ?

3) ***Quelle trace écrite ?***
Dans les autres domaines d'étude des mathématiques, l'enseignant met parfois en œuvre un dispositif vers une trace écrite autour des notions étudiées (synthèse, prise de notes, cours, fiches d'exercices, fiches méthodes …)
Qu'en est-il pour l'apprentissage de l'algorithmique et de la programmation dont le support privilégié est un support informatique ?
En quoi une trace écrite peut-elle ici consister ?
Peut-elle se réduire à des exemples significatifs ?
À quel moment cette trace écrite peut-elle apparaitre ?

4) ***L'algorithmique et la programmation : en EPI ? en AP ?***
Les heures utilisées en AP ou en EPI peuvent-elles être valorisées par des entrées algorithmiques ?
Dans quelle mesure l'algorithmique peut-elle être un support à l'interdisciplinarité ?
Quelle peut-être la place d'une pédagogie de projet avec l'apport de l'algorithmique ?

5) ***Quelle évaluation ?***
Comment mettre en œuvre une évaluation de l'algorithmique ? Quelle est la place des compétences dans les activités que l'on peut proposer ?

6) ***Quelle liaison collège-lycée ?***
Quelles passerelles entre l'enseignement de l'algorithmique en collège et celui du lycée ?
Quels outil(s) de programmation va t-on utiliser au lycée après le « Scratch » du collège ?
Quels éléments de l'enseignement d'exploration ICN en 2nde sont initiés dans le cadre des programmes de collège ?

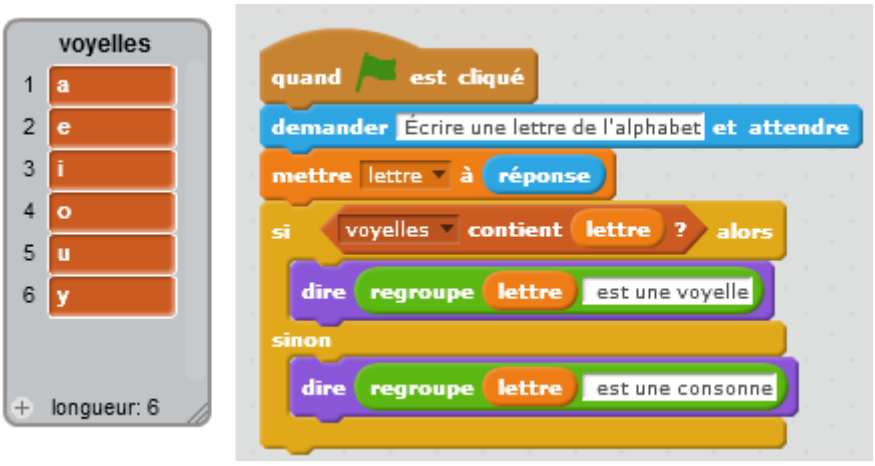
2) QUELQUES ELEMENTS POUR UNE PROGRESSION DE CYCLE 4.

La mise en œuvre des programmes de cycle et les retours des enseignants à l'occasion des journées de formation autour de l'algorithmique ont permis de faire émerger un autre questionnement essentiel : **Quelle progression de cycle 4 peut-on mettre en place pour l'algorithmique et la programmation qui respecte la démarche de projet attendue par le programme ?**

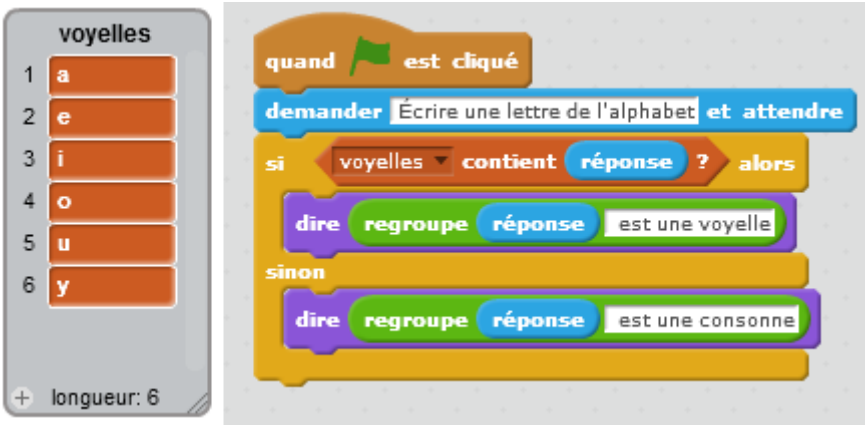
Axe 1 : avec les variables et les instructions conditionnelles dès la 5e.

« La difficulté n'est pas dans les instructions utilisées »

Un programme comme celui-ci :



qui peut d'ailleurs être réduit sans recours à une variable « lettre » :



est totalement accepté et compris dès la fin du cycle 3. Il a été testé auprès d'élèves de 6e (qui ont eu à le construire).
Ce programme contient pourtant un test conditionnel et une liste !

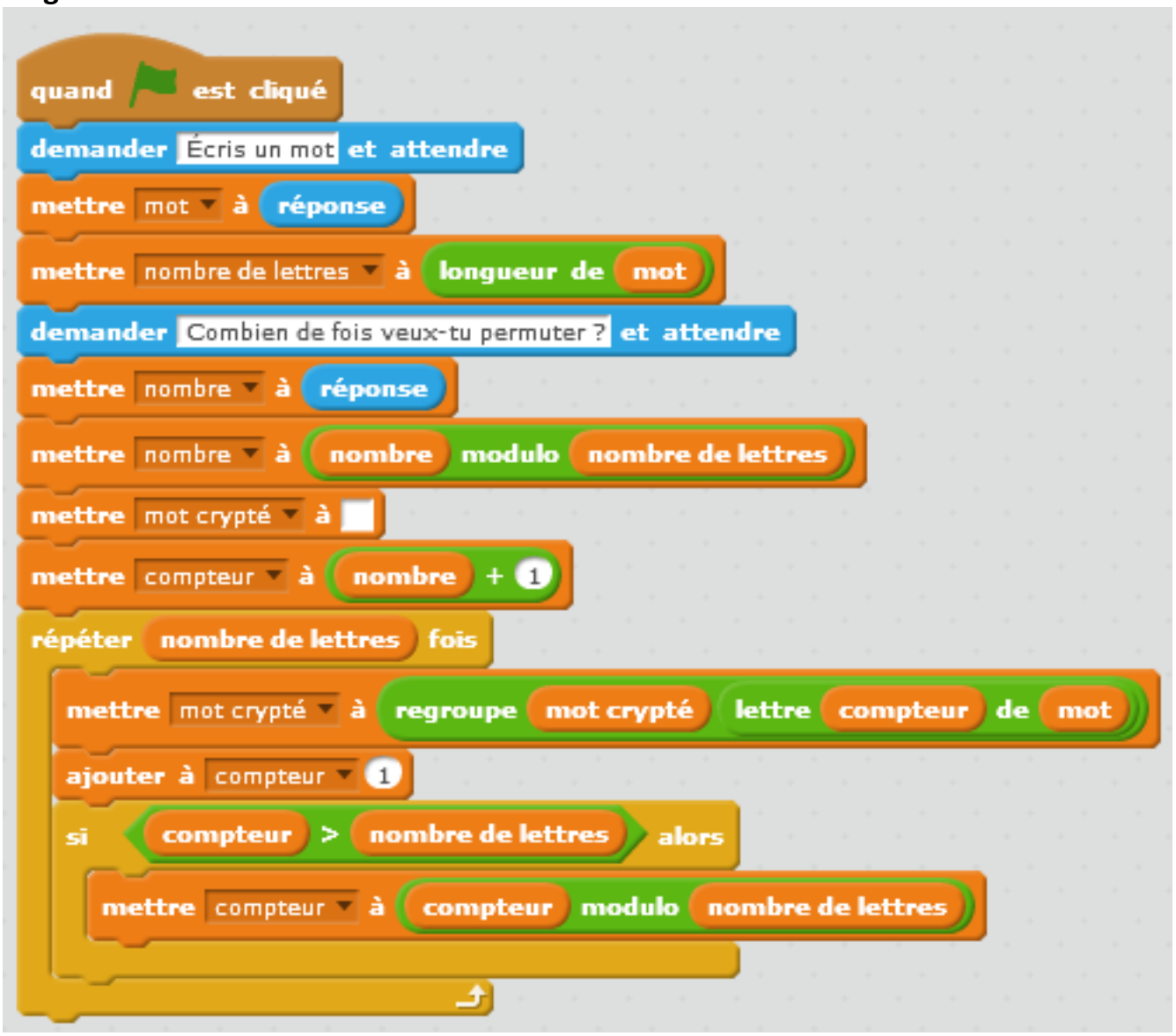
Nous sommes convaincus qu'une progression de cycle construite sur un principe du style « la boucle pas avant la 5e », « le test conditionnel et la variable pas avant la 4e » et « la liste et le bloc pas avant la 3e » n'a pas de sens !

Ce qui complexifie un programme n'est pas l'instruction en elle-même mais l'usage qui en est fait !

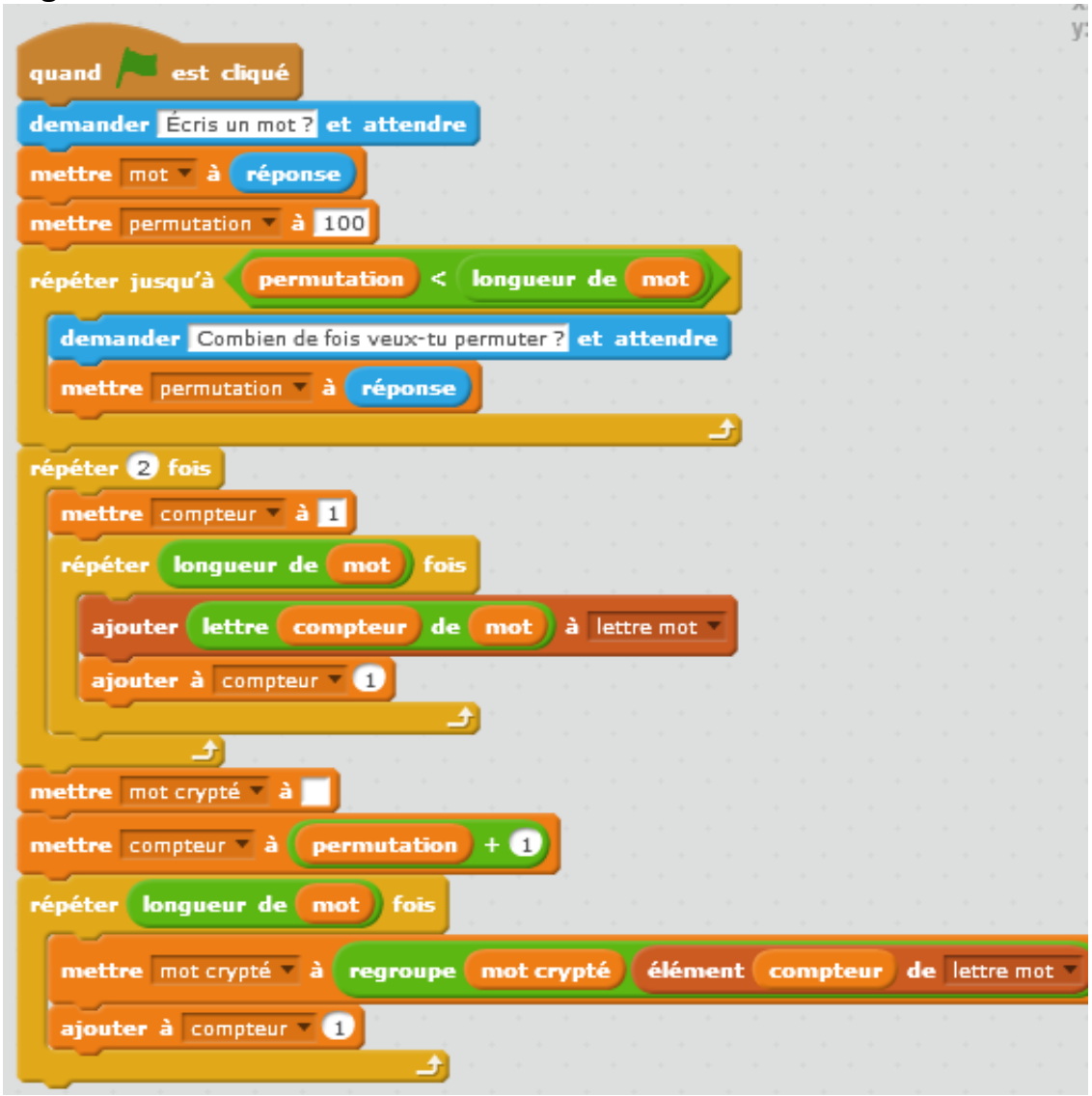
Ce n'est pas non plus le nombre d'instructions : un programme court peut être plus simple qu'un programme long mais il peut aussi être beaucoup plus délicat et difficile.

Prenons un autre exemple, proposé en formation à des enseignants et qui n'a pas vocation à être proposé actuellement en classe, celui d'un programme qui ferait une permutation circulaire sur les lettres d'un mot : le mot *maison* peut ainsi devenir si on permute deux lettres *onmais* ou *isonma* selon le sens choisi.
Les deux programmes suivants respectent cette consigne.
Le plus délicat à comprendre et à interpréter est sans nul doute le programme n°1 qui est aussi plus court !

Programme 1



Programme 2

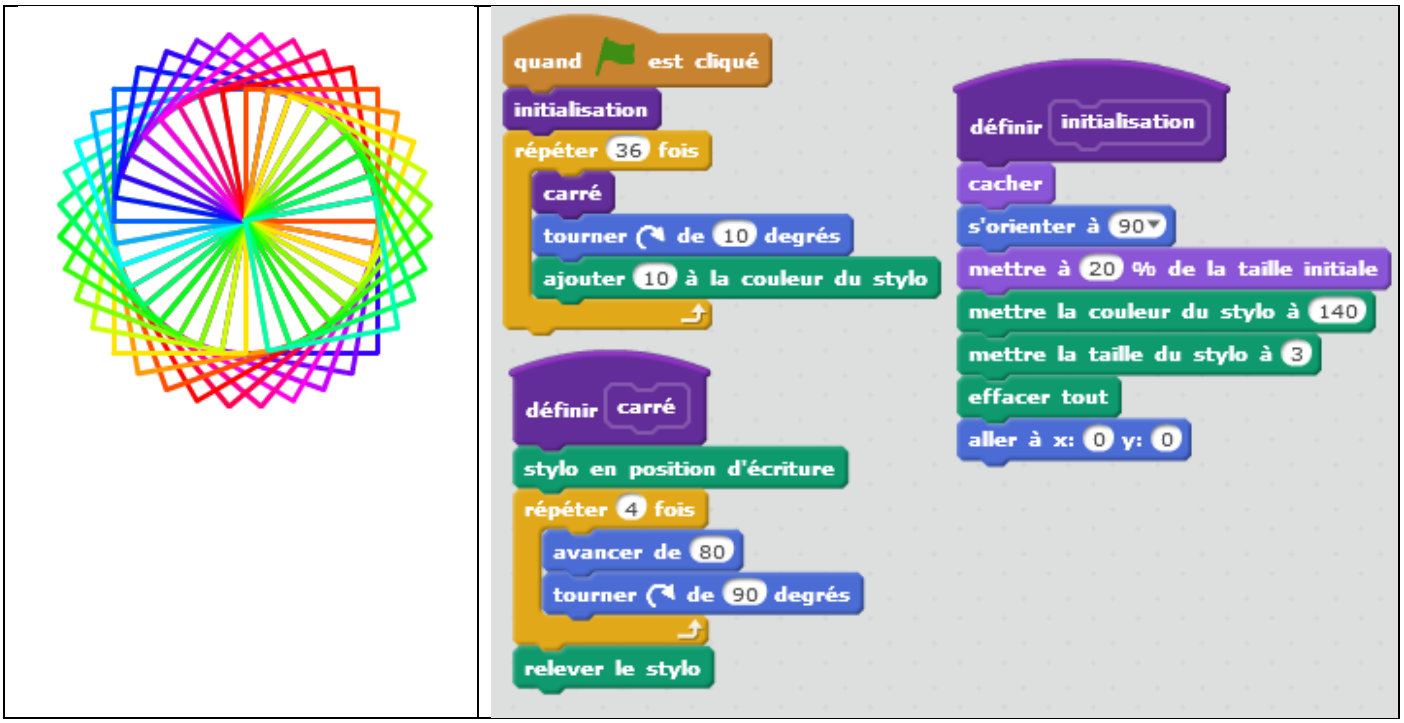


Ce programme construit une liste avec les lettres du mot répétées 2 fois : il suffit alors de recomposer le mot en décalant.

Les élèves vont souvent être amenés à travailler dans une stratégie de type essais/erreurs. Il s'agit bien évidemment de leur faire analyser leurs erreurs pour orienter les améliorations. À la façon de Samuel Becket, on pourrait dire : « essaie, rate, essaie encore, rate encore, rate mieux ! »

Axe 2 : des activités progressives et des jeux à programmer dès la 5e.

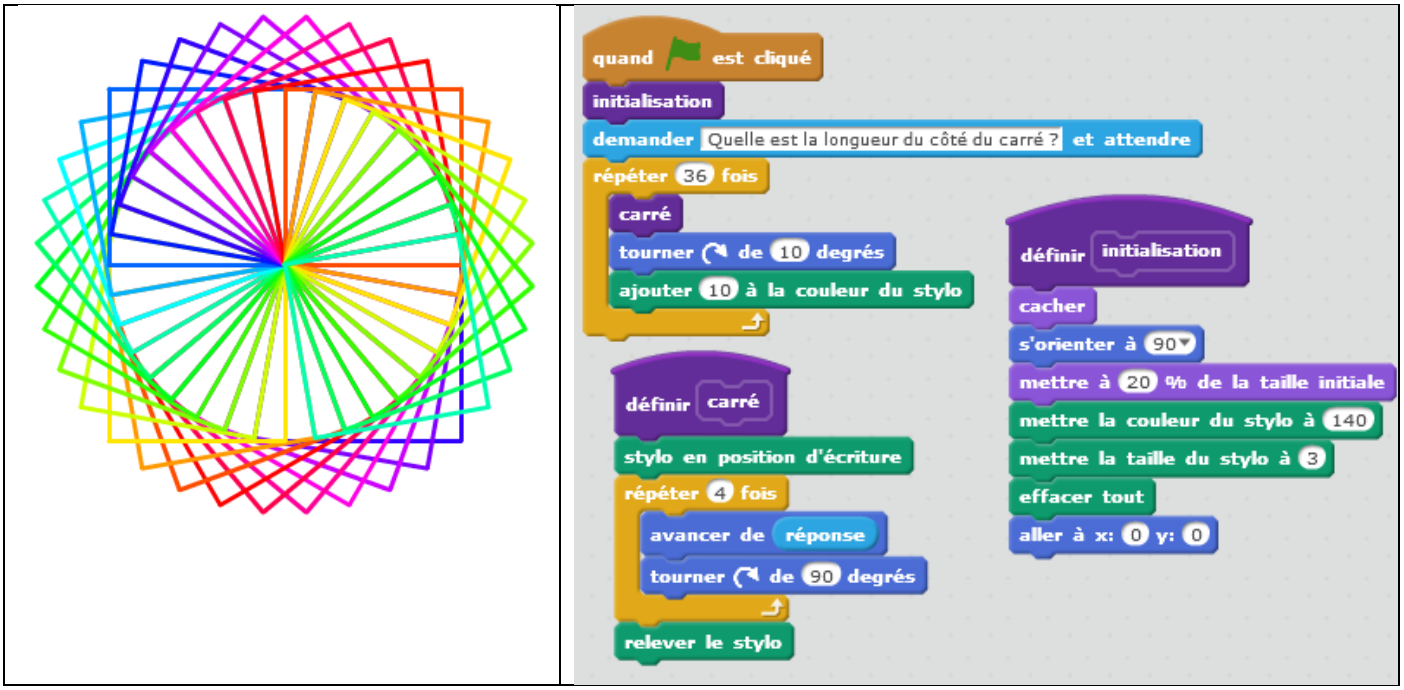
Dès la classe de 5e, après quelques activités pour débuter (revoir « des idées pour démarrer »), on peut proposer aux élèves de construire des figures surprenantes avec pourtant des programmes abordables. Ce faisant, on illustre la démarche de création préconisée avec cet enseignement en mettant en évidence la facilité de sa mise en œuvre :



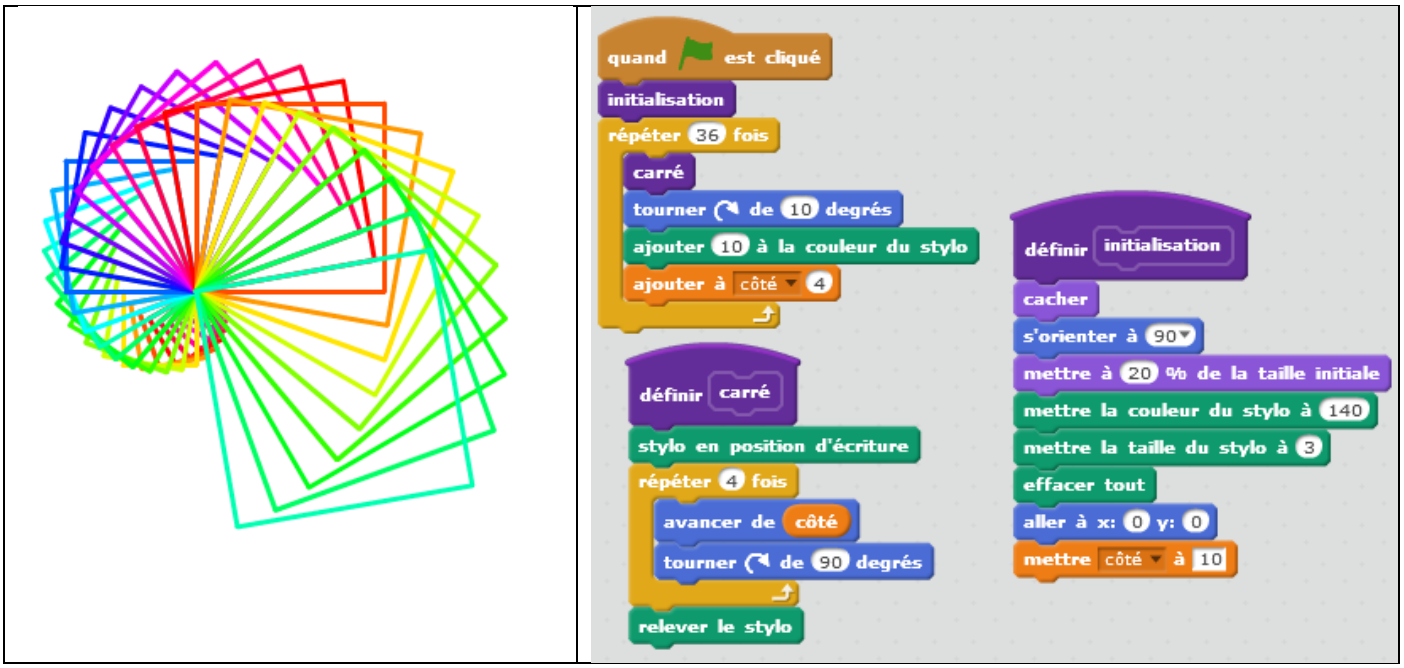
Dans la phase de conception du programme, le lutin est apparent et de taille réduite pour ne pas gêner la visualisation du tracé (« mettre à 20% la taille du lutin »).
Lorsque le programme est finalisé, le lutin peut être caché (instruction « caché » rajoutée).
Ces 2 étapes sont visibles dans le bloc initialisation.

Des évolutions possibles :

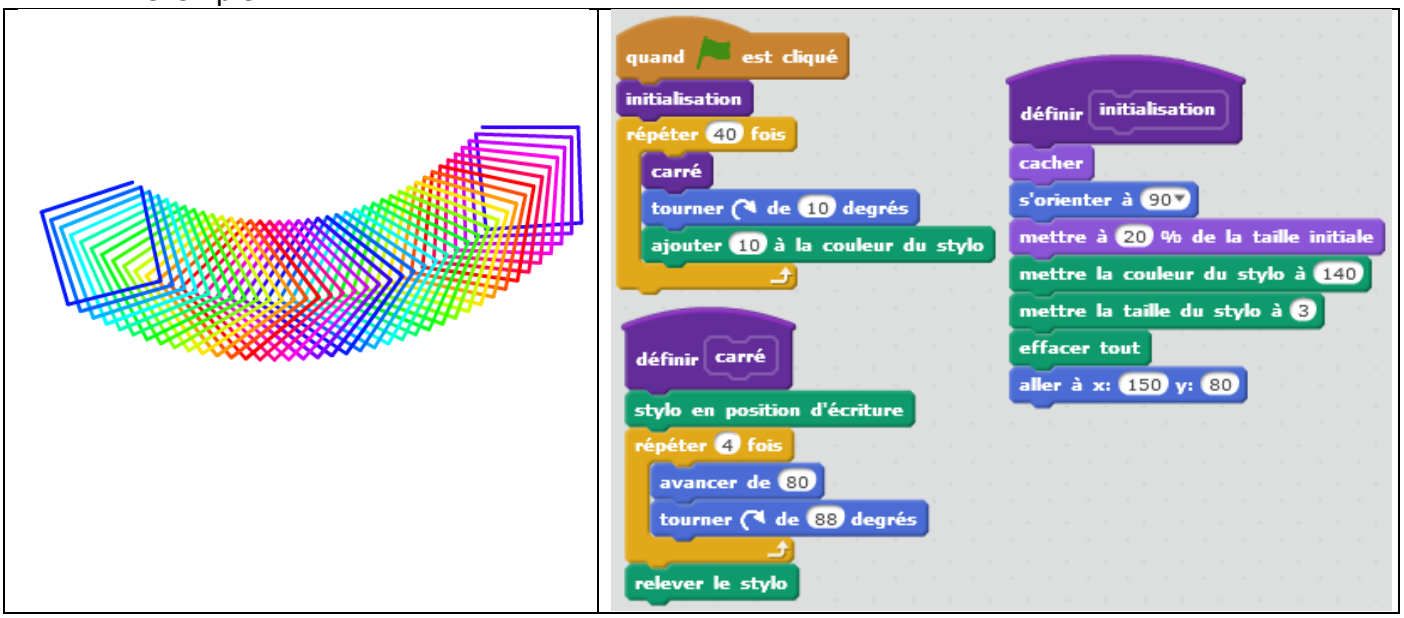
- Demander à l'utilisateur la taille du carré



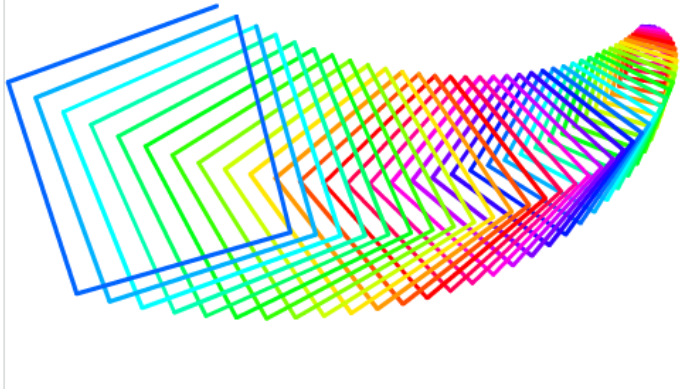
- Faire évoluer la taille du carré

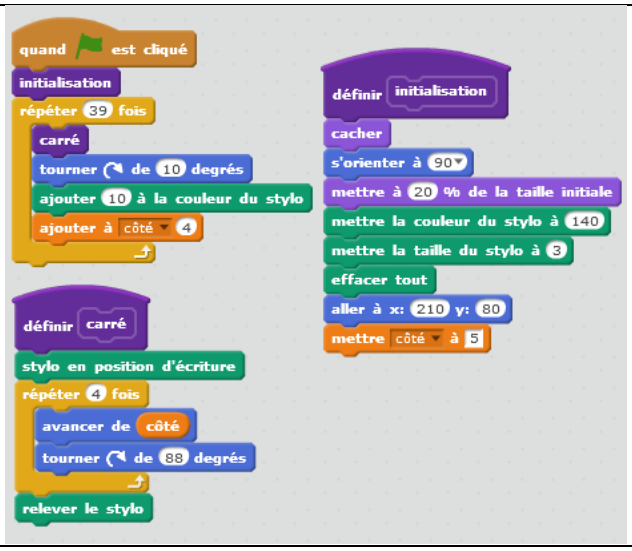


- Observer une « déformation » en modifiant l'angle de 90° dans la définition du carré : par exemple

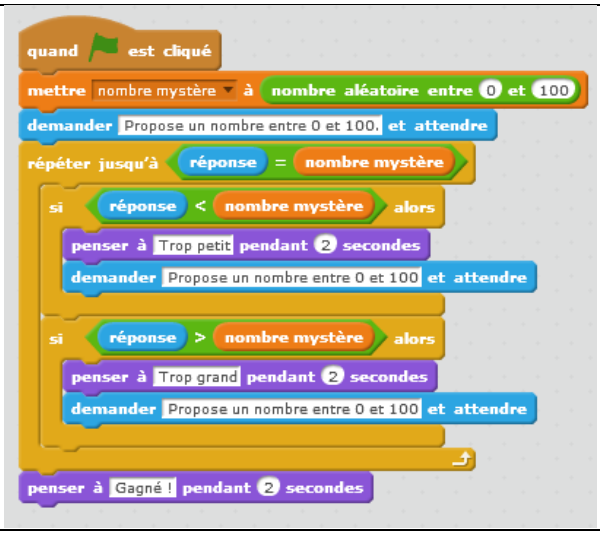


- Modifier le programme précédent en faisant évoluer la taille du « carré »



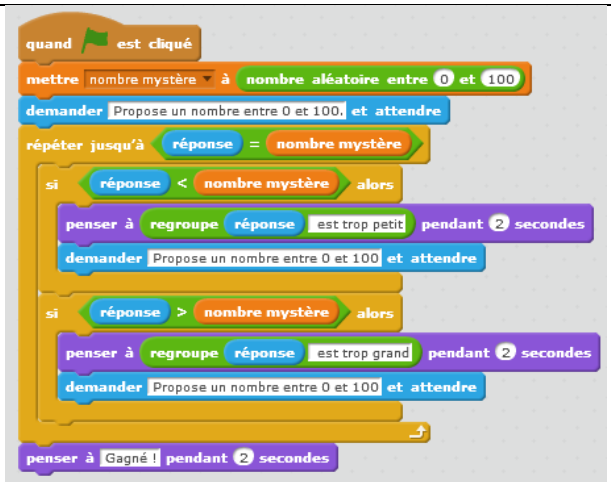


On peut aussi proposer de réaliser des petits jeux type quizz qu'on peut faire évoluer facilement pour s'adapter à la diversité des élèves :

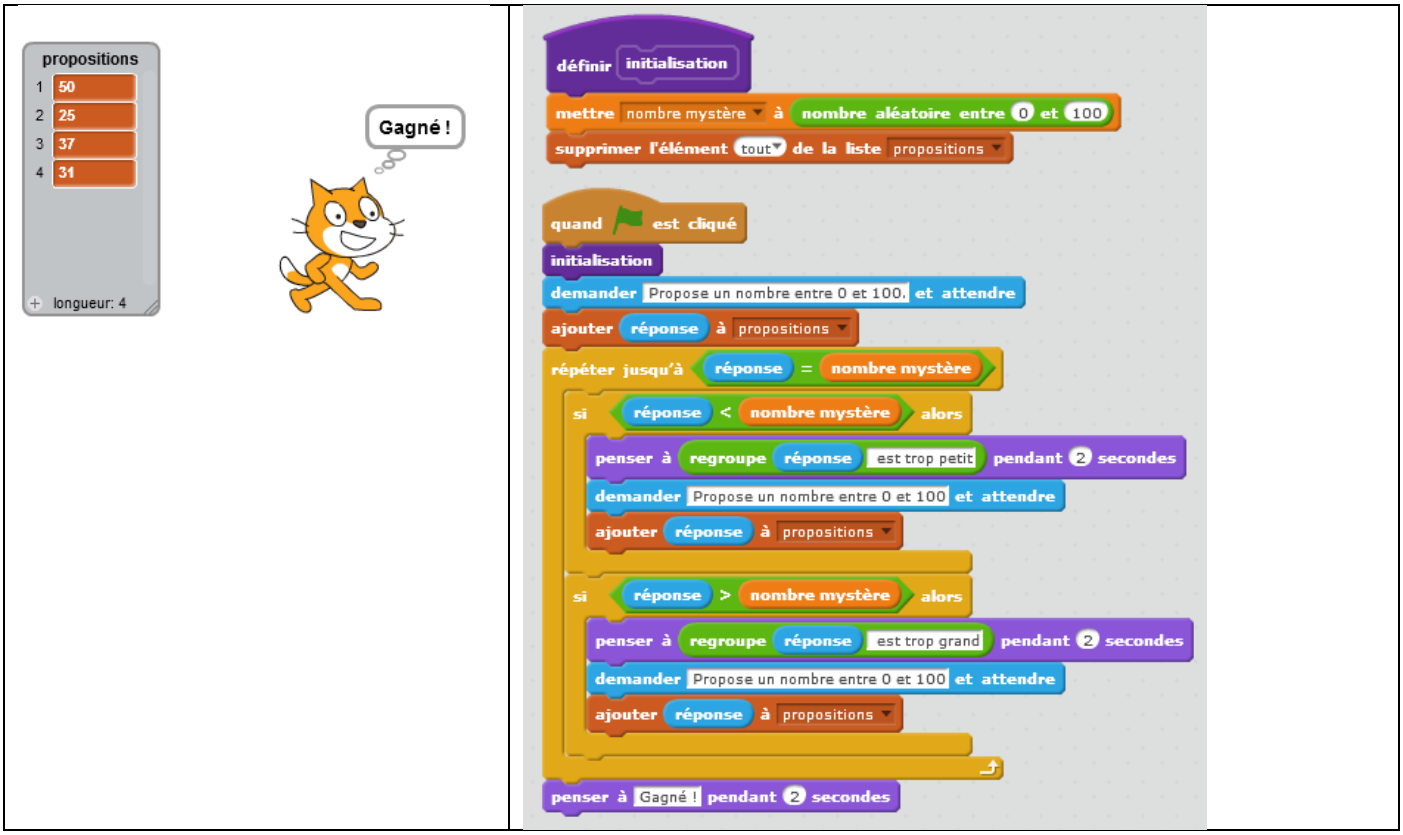


Des évolutions possibles :

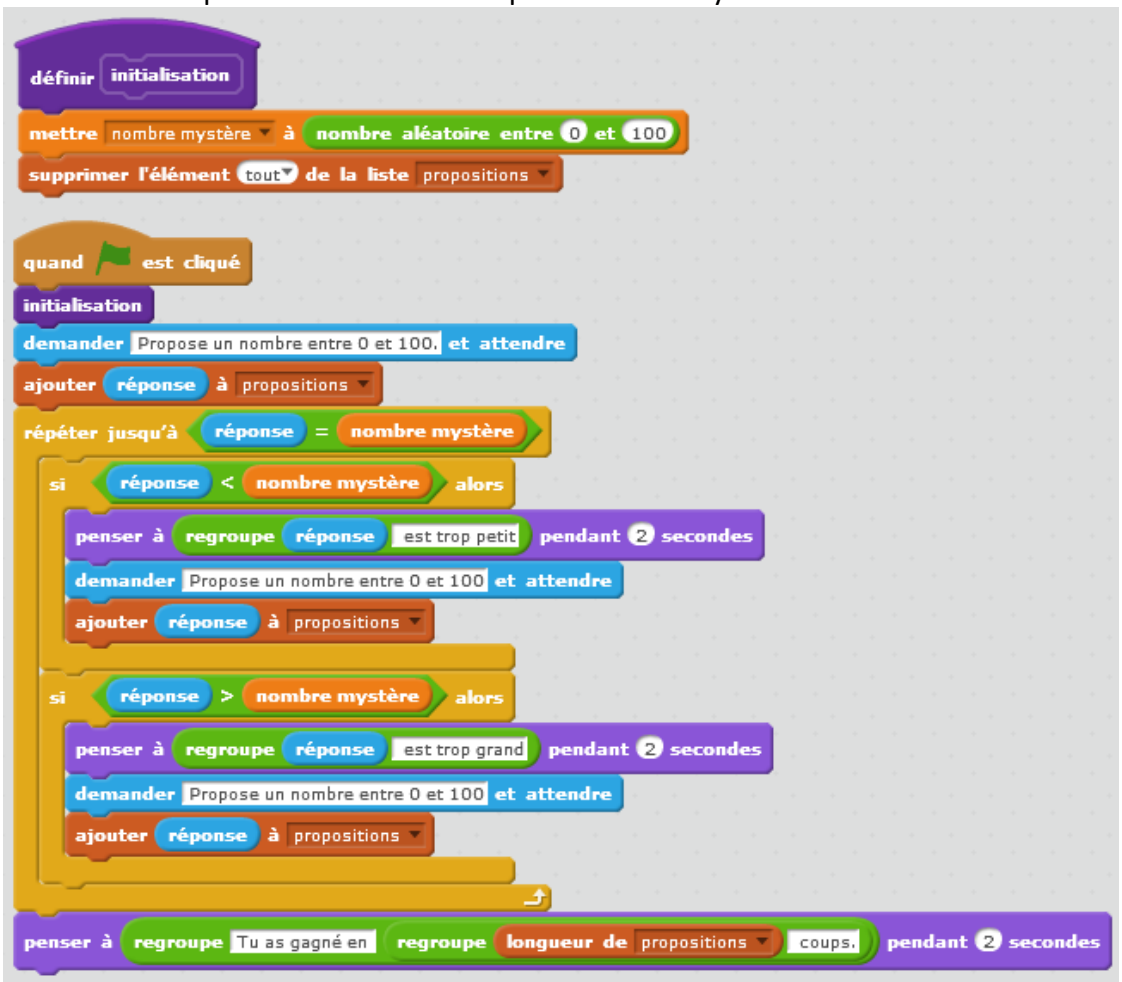
- Indiquer quel nombre a été proposé par l'utilisateur



- Garder mémoire de ces nombres



- Indiquer en combien de coups le nombre mystère a été trouvé



On remarque ici que la liste sert de compteur.

Comment estimer la difficulté de programmation d'un script ?

Il s'agit ici de donner quelques éléments d'analyse amenant à connaitre le niveau de difficulté de programmation pour un élève.

1) Le nombre de variables

L'usage de la variable informatique est une difficulté majeure pour les élèves. Cela devient alors très délicat dès lors que plusieurs variables doivent être utilisées et modifiées dans le programme lui-même.

Créer une variable NOM et une variable PRÉNOM pour ensuite les demander à l'utilisateur ne pose pas de problème : ces variables sont juste saisies et réutilisées sans être traitées davantage.
En revanche, utiliser deux variables COMPTEUR1 et COMPTEUR2 ainsi qu'une variable pour mémoriser le mot de départ et une autre pour le mot crypté dans un programme tel que le code César est beaucoup plus délicat. L'un des compteurs permet de repérer de quelle lettre de l'alphabet il s'agit, l'autre sert à choisir les lettres du mot de départ une par une afin de lancer le cryptage.

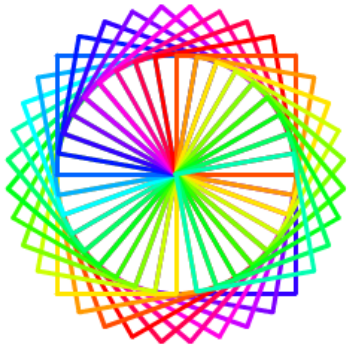
➔ Pistes pour ne pas accentuer les difficultés :

- Choisir des noms de variable signifiants : il est plus facile de comprendre le script lorsque la variable est nommée COMPTEUR1 plutôt que A.
- Penser aux listes : Le programme 2 présenté page 7 utilise une liste qui simplifie la compréhension du processus : il est facile de visualiser le décalage des lettres.
- Penser une progressivité dans l'utilisation des variables en tenant compte de leur nombre et de leur traitement.

2) Le nombre de boucles

Utiliser une boucle de type « répéter 4 fois » pour programmer la construction d'un carré est simple et accessible dès la classe de 6e.
Répéter cette construction en la faisant tourner peut rendre le programme plus difficile à comprendre.

Élément de programme A

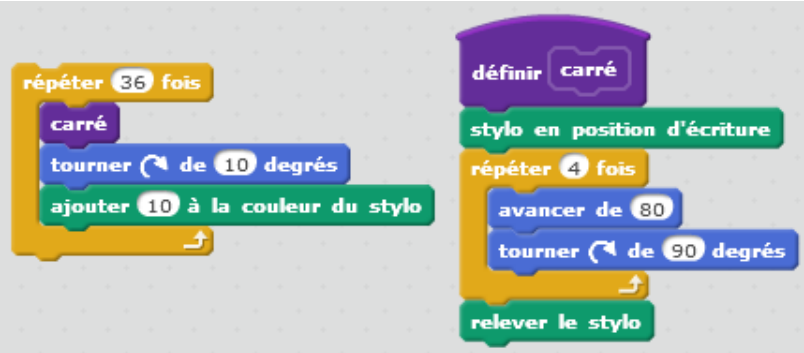


L'élément de programme A ci-dessus permet de réaliser la figure jointe, mais les boucles imbriquées rendent la lecture du programme plus difficile.

➔ Piste pour ne pas accentuer les difficultés :

L'usage d'un sous-programme « carré », un bloc donc, permet de séparer les boucles en les associant à leurs fonctions : l'une permet de tracer le carré, l'autre permet de donner l'effet visuel du carré qui tourne.

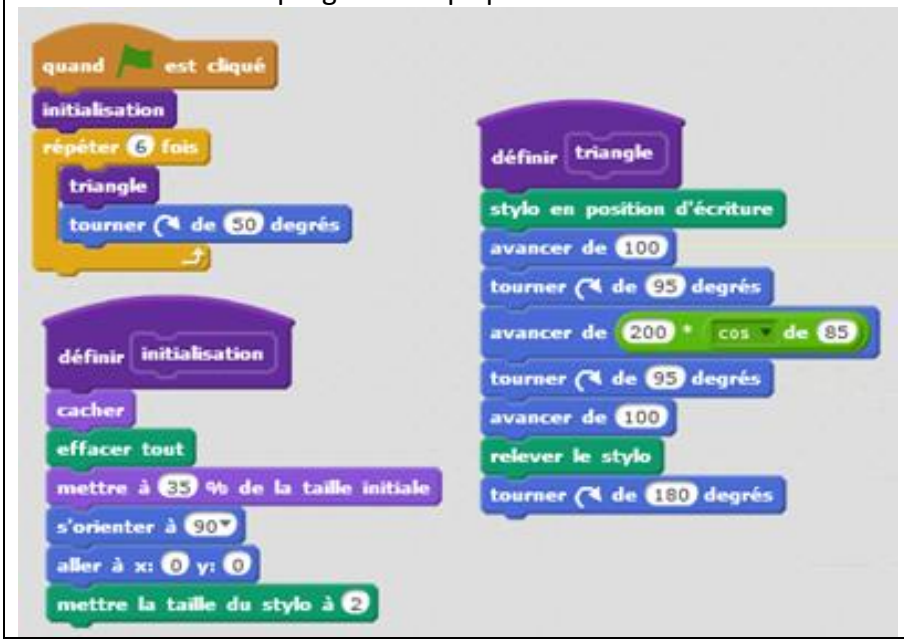
Élément de programme B

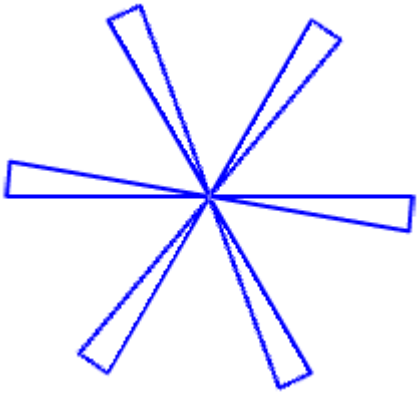


3) Dessiner ou estampiller ?

Le choix fait va souvent avoir des conséquences sur la difficulté du script mais aussi sur les notions travaillées par les élèves. *Ces programmes ont été fournis, sur papier, à des élèves de 4e pour tester leur compréhension avec la mention que la compréhension de toutes les instructions n'était pas nécessaire.*

La figure ci-contre peut être obtenue en traçant un triangle et en faisant tourner.
On obtient alors un programme qui peut être comme celui-ci :

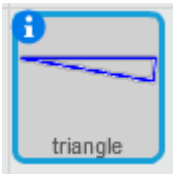




On peut aussi prendre un triangle comme lutin, bien positionner le centre et le faire tourner :

Le programme est beaucoup plus simple !

(Voir sujet du devoir avec consignes et rappels sur la notion « estampiller » en annexe 0)



En conclusion, nous voulons mettre en avant deux idées qui nous paraissent devoir guider nos pratiques avec les élèves :

- Il ne faut pas se sentir obligé d'attendre la fin du cycle pour faire utiliser aux élèves des variables, des instructions conditionnelles, on peut utiliser ces outils avec un degré de complexité assez modeste.
- Il ne faut pas se priver de proposer aux élèves, dès le début du cycle 4, des activités et des programmations de petits jeux dans lesquels l'élève peut avoir une part de liberté, de créativité.

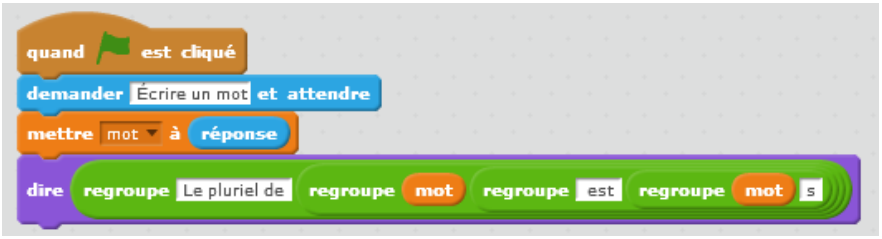
3) EVALUER ET DIFFERENCIER L'APPRENTISSAGE DE L'ALGORITHMIQUE

Axe 1 : Proposer des thèmes qui peuvent être traités à plusieurs niveaux de complexité

L'approche proposée, et qui a été testée dans l'académie, est d'amener les élèves dès la fin du cycle 3 ou dès le début du cycle 4 à utiliser les instructions « essentielles » de l'algorithmique : celles qui auront une place prépondérante dans les attentes du lycée, y compris dans l'usage d'un langage de programmation (tel que Python par exemple) : la variable, les tests conditionnels, les boucles, les sous-programmes.

Pluriel et féminin :

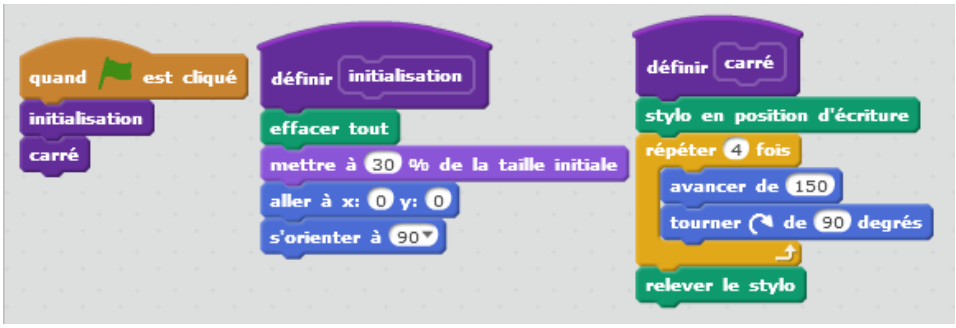
Les deux programmes précédents sont d'une grande simplicité : ils permettent de mettre au pluriel un mot auquel il suffit d'ajouter un « s » ou au féminin un mot auquel il suffit d'ajouter un « e ».



La structure identique des deux programmes permet d'en ressortir l'idée du sous-programme mais celui-ci donnera un script beaucoup plus délicat à réaliser, par exemple avec un bloc paramétré :

On peut raisonnablement penser que si le bloc est accessible très tôt dans la formation des élèves (dès le cycle 3), le bloc paramétré sera lui plutôt destiné à l'attention d'élèves ayant déjà une bonne maîtrise de l'outil.

Le script ci-dessous, utilisant deux blocs clairement identifiés, peut être utilisé par des élèves dès la fin du cycle 3 ou au début du cycle 4.



Axe 2 : Des pistes et des exemples pour l'évaluation

À compter de la session 2017, l'algorithmique est évaluée au brevet. Cette évaluation demande un traitement sur papier d'éléments d'algorithmique. On est là dans un cadre institutionnel, mettant en œuvre des attentes minimales qui pourront d'ailleurs évoluer au fur et à mesure du développement de l'algorithmique en collège jusqu'à ce qu'une cohorte ait pratiqué sur la totalité du cycle 4, voire du cycle 3 et du cycle 4.

Les enjeux de l'algorithmique en collège vont bien sûr bien au-delà de cette évaluation finale.
La maitrise des connaissances et des compétences liées à l'algorithmique – la pensée algorithmique – doit sans doute passer par une démarche active des élèves quant à ce qu'il convient pour chacun de retenir. Par exemple, à la suite d'une activité, des élèves peuvent eux-mêmes indiquer ce qui a été essentiel pour eux :

Quels blocs "importants" avez-vous utilisés ?	*A quoi servent-ils ?*	*A quoi faut-il faire attention dans leurs utilisations ?*
Cacher - montrer	apparaître ou non	si on attends ou pas, bien les placés.
chronomètre	chronomètre le temps de réaction	qu'il s'arrête.
quand le lutin est touché	s'arrête le chrono quand le lutin est touché	qu'ils soit mis au bon endroit

Quels blocs "importants" avez-vous utilisés ?	*A quoi servent-ils ?*	*A quoi faut-il faire attention dans leurs utilisations ?*
Chronomètre	Réaction le chronomètre	
nombre aléatoire entre 1 et 10	Choisir un nombre de 1 et 10	
Cacher	Chat cacher	
Montrer	Tout le chat montrer	

Une autre piste consiste à donner aux élèves les éléments qui sont apparus comme essentiels d'un point de vue algorithmique. Les élèves indiquent alors eux-mêmes ce qu'ils en ont retiré. (Voir annexe 4) Le même document peut servir sur l'ensemble du collège : la notion de variable informatique par exemple va nécessairement évoluer de la 6e à la 3e.

Dans ces deux approches, nous observons que l'évaluation est conçue essentiellement comme une auto-évaluation où la trace écrite personnelle de l'élève sert de support.

Pour compléter la formation des élèves et varier les dispositifs d'évaluation, rapides ou plus formels, nous utilisons également deux types d'activités débranchées :

- Des questions types « activités rapides avec Scratch » en début de séance, permettant à la fois de développer et de travailler des capacités en algorithmique et en calcul ou géométrie.
- Des exercices ou situations problèmes, avec des copies d'écran du logiciel, dans certains temps de recherche en classe, de travail personnel ou de devoir sur table. Une banque de tels exercices est en ligne sur le site académique nantais.

4) DES MOMENTS DE REGULATION A LA TRACE ECRITE

Les temps de régulation

Il est rare que tous les élèves d'une classe aient besoin d'un temps de régulation lors d'une activité, sauf éventuellement en 6e/5e dans une phase de découverte du logiciel.
Une des clés de ces temps de régulation, si ce n'est la clé principale, sera l'algorithmique débranchée : les élèves enlèvent la main de la souris, idéalement quittent l'ordinateur, et la réflexion porte sur la structure de programme attendu.

Il est clair alors que deux types de temps de régulation sont à prévoir :

- ceux qui se feront pendant la séance en salle informatique : un groupe d'élèves se retrouve en difficulté face à une situation et il s'agit d'amener les éléments qui permettront de les débloquer ;
- ceux qui se feront lors des séances « classiques » : outre l'entretien des connaissances par le biais de questions flash, de devoirs maisons, de questions algorithmiques dans des activités ou devoirs, il peut aussi y avoir des analyses a posteriori de séances.

Deux situations ayant amené des régulations différentes :

En 4e (annexe 1)
Il s'agit de réaliser une montre : les images (cadran, petite aiguille et grande aiguille) sont fournies. L'utilisateur indique l'heure et les aiguilles se positionnent correctement.
Les élèves comprennent tous la consigne : la demande correspond à quelque chose qu'ils maitrisent parfaitement dans leur quotidien. Cela a sans doute anesthésié leur esprit d'analyse !
Le problème est pris « de front », sans chercher à le décortiquer, et au bout d'une heure, malgré les demandes répétées du professeur pour prendre le temps de réfléchir avant de programmer, le constat est clair : non seulement aucun programme ne fonctionne, mais il n'y a même pas le début d'un commencement. Le fait de travailler en groupes ne change rien !
La régulation porte naturellement alors sur la réflexion à avoir avant de programmer : analyse de la demande, décomposition en sous-programmes simples, …
Lors d'une activité ultérieure posant problème, des élèves ont su prendre un temps de réflexion pour analyser la situation en lâchant le clavier et la souris.

En 3e (annexe 2)
Il s'agit pour les élèves de réaliser une conversion en morse. Le travail est prévu sur quatre séances s'étalant sur 6 semaines. Les élèves travaillent en groupe et beaucoup se passionnent pour ce travail : des groupes poursuivent en dehors des heures en salle informatique.
Pour autant, quelques groupes se retrouvent en difficulté sur quelques points.
Le temps de régulation collectif (un groupe autonome n'y participe pas) porte sur des éléments de programmation permettant
1- de conforter ce qui a été fait pour certains ;
2- de proposer d'autres pistes envisageables ;
3- de valider la décomposition en sous-programmes simples en renvoyant à la classe des éléments de programmes obtenus dans certains groupes.

Les traces écrites

Faire suivre les régulations par une trace écrite qui récapitule ce qui a fait obstacle, les méthodes pour surmonter les difficultés, les points essentiels qui ont été soulignés peut être une aide efficace pour rappeler ce qui a été conduit lors d'une séance précédente et faciliter la poursuite du travail des élèves. Ces traces peuvent être réinvesties au besoin pour d'autres activités algorithmiques ou pour être complétées.
Les documents figurant en annexe (1 à 4) ont été distribués aux élèves à cet effet.

Communication sur un programme

Une feuille de régulation (en particulier celle en 4e) n'est pas distribuée directement. Elle arrive en conclusion d'un temps de discussion dans la classe où les élèves sont amenés à parler des difficultés rencontrées. Elle illustre la nécessité de communiquer sur un programme, notamment sur les éléments qui ont fait obstacles. De la même façon, la structure d'un programme (en particulier s'il est élaboré) devra pouvoir être expliquée par l'élève (le groupe) qui l'aura réalisé.

5) VERS LE LYCEE

Dans le programme de seconde proposé pour la rentrée 2017, on peut lire :
« Au cycle 4, en mathématiques et en technologie, les élèves ont appris à écrire, mettre au point et exécuter un programme simple. Ce qui est proposé dans ce programme est une consolidation des acquis du cycle 4 autour de deux idées essentielles :
- *la notion universelle de fonction d'une part, et*
- *la programmation comme production d'un texte dans un langage informatique d'autre part. »*

Commencer la programmation, au collège ou en primaire, à l'aide d'un langage de programmation graphique permet d'envisager cet apprentissage dans la durée. En arrivant au lycée, les élèves auront en effet des notions beaucoup plus importantes de l'algorithmique et de la programmation qu'actuellement. On peut s'attendre à ce qu'ils aient une maitrise raisonnable des variables, des instructions conditionnelles et de la boucle « répéter ». Les élèves ont naturellement tendance à privilégier une entrée par la manipulation en direct de blocs dont ils peuvent immédiatement vérifier les effets.

Les élèves au lycée doivent dorénavant s'initier à la programmation textuelle. Cette appropriation d'un langage de programmation nécessite de leur part un investissement individuel pour la maîtrise de la syntaxe. En langage textuel (Calculatrice, Python, Processing, Javascript, BricxCC …), les élèves ne profitent plus de la couleur des blocs qui est très pratique sur Scratch. Ils doivent apprendre à fermer les structures (accolades fermantes, fin d'indentation..) ce qui se fait automatiquement avec Scratch. Ils doivent, avec certains langages, mettre un point-virgule en fin d'instruction. Les oublis de ces codes sont source d'erreurs et, si les erreurs à déboguer sont nombreuses, peuvent facilement conduire à une gestion très difficile de la séance par l'enseignant. On peut donc légitimement se demander **comment ce passage à la programmation textuelle peut être facilité par la programmation graphique pratiquée en primaire et au collège.**

a) Continuer l'apprentissage de l'autonomie

Comme dans toutes activités mathématiques, il est important que les élèves prennent conscience qu'ils doivent travailler de manière autonome aussi en programmation. Plutôt que d'appeler leur professeur dès que leur programme ne fonctionne pas, ils doivent avoir le réflexe de trouver par eux-mêmes, ou avec l'aide de leurs camarades, les solutions à leurs problèmes. Construire chez les élèves cette habitude dès l'apprentissage de l'algorithmique est un atout pour aborder des séances de programmation plus exigeantes au lycée. Non seulement cette autonomie facilite l'efficacité des apprentissages, mais elle construit également la nécessaire confiance en soi pour surmonter les éventuelles difficultés rencontrées.

b) Nommer ses fichiers, organiser ses espaces informatiques personnels

L'apprentissage de cette autonomie se construit aussi dans la gestion par les élèves de leur environnement de travail. Bien que cela ne soit pas propre à l'enseignement de l'algorithmique et concerne toutes les disciplines, on peut attendre des élèves qu'ils ne quittent pas le collège sans savoir organiser leurs fichiers et leurs dossiers informatiques.

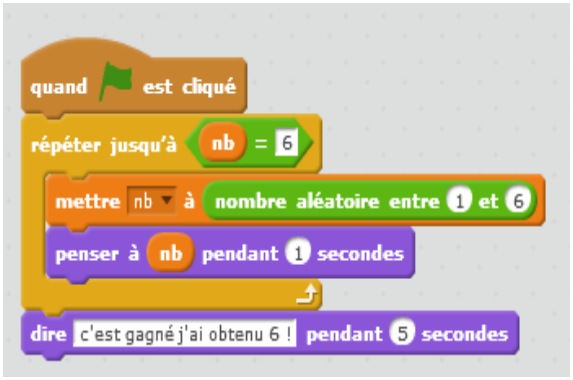
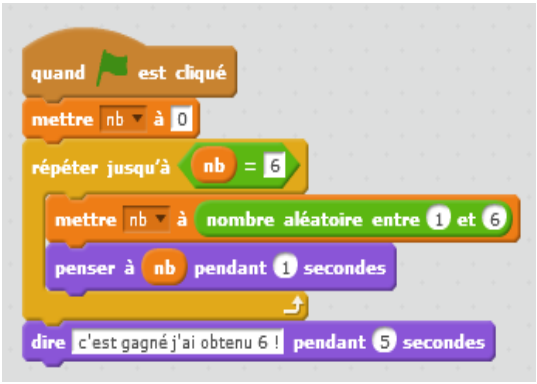
Exemples illustrant l'importance de cette organisation :
- D'une habitude installée au collège, celle de nommer efficacement les fichiers produits (on évite un fichier nommé « TP maths » ou « algo » qui ne sera pas identifié parmi plusieurs travaux), on glissera au lycée vers une nomenclature sans espace ni accentuation pour la dénomination des fichiers (qui sera reconnu par tous les systèmes et logiciels utilisés).
- Lorsqu'on manipule une image pour créer un lutin avec Scratch, l'adresse relative de cette image par rapport au fichier Scratch n'a aucune importance car l'image est ''chargée'' dans le fichier Scratch. Mais avec les logiciels de programmation (ou encore, dans certains enseignements de spécialité, avec le html/CSS/Javascript - utilisé par exemple pour faire des diaporamas) l'adressage relatif des images est très important et source de nombreuses erreurs chez les lycéens.

c) Comprendre les types de variables / savoir les initialiser

Sans que cela ne devienne un objectif de maîtrise, il peut être judicieux de saisir les occasions qui se présentent pour construire une familiarité progressive des élèves avec la notion **de type d'une variable,** déjà sous-jacente avec Scratch. On peut, en effet, créer deux types de variables : les variables et les listes. L'utilisation des blocs conduit à identifier le type de paramètres utilisés.

L'initialisation des variables et/ou leur initialisation par défaut, elle, va naturellement être travaillée au collège. Des exemples de jeux simples, où le fait de ne pas remettre la variable à zéro conduit à ce que le jeu ne marche plus, permettent de poser ce problème.

Exemple pour montrer l'importance de l'initialisation :

Si on lance deux fois de suite le code ci-dessous, lors de la deuxième utilisation nb vaut 6 et on ne rentre pas dans la structure répétitive.	L'initialisation de la variable nb étant faite, le programme peut être utilisé plusieurs fois de suite.
	

d) Commenter ses programmes

Commenter son code est une bonne habitude à prendre pour développer la compétence « communiquer » aussi bien à l'écrit qu'à l'oral. On commente son code pour vérifier que l'on a bien compris ce que font les blocs que l'on a assemblés. On commente son code pour permettre à un camarade ou à son professeur de comprendre ce que l'on a voulu faire. Parler de son code contribue à la maitrise du bon vocabulaire, à la formulation précise de questions face aux obstacles rencontrés, au développement de l'expression orale, indispensable à travailler aussi en mathématiques. Lors des oraux d'informatique que passent certains lycéens, on peut constater que le vocabulaire attendu est loin d'être maitrisé. Se pose légitimement la question d'identifier les moments de la scolarité où un temps a été accordé à cet apprentissage. Développer cette habitude dès le début de la programmation graphique ne peut que servir les élèves, au-delà même de l'enseignement de l'algorithmique et même des mathématiques.

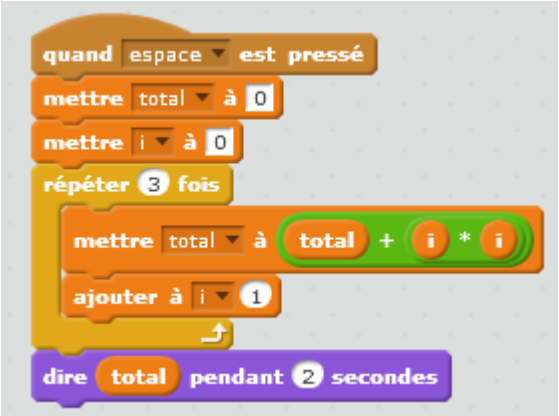
	Initialisation de la variable x Incrémentation de la variable x
	Structure conditionnelle
	Structures répétitives
	Ce sont des instructions

e) Utiliser des tableaux d'instructions

En ce qui concerne le passage à l'écrit, l'utilisation de tableaux d'instructions au lycée aide beaucoup les élèves. Leur utilisation peut être aussi utile lors des travaux faits avec Scratch. Après une phase de codage qui dysfonctionne (ou pas !), une telle pratique amène les élèves à se poser et à essayer de comprendre ce que fait leur programme. Notons qu'au baccalauréat, de nombreux tableaux d'instructions sont proposés.

Exemple de tableau des instructions :

Instructions	total	i	Affichage de total
1	0	/	/
2	0	0	/
3	0	0	/
4	0	1	/
5	1	1	/
6	1	2	/
7	5	2	/
8	5	3	/
9	14	3	14



f) Préciser les structures répétitives :

Il peut être utile de progressivement faire prendre conscience aux élèves qu'il y a deux structures répétitives : les structures dont on ne connaît pas a priori le nombre d'itérations (''répéter jusqu'à …'') et les structures dont on connaît le nombre d'itérations (''répéter … fois'').

a) La structure « répéter jusqu'à … » de Scratch et la structure « Tant que » (ou while) :

La transition se fera sans trop de problème dès lors que les élèves auront compris que :

- la structure « while » des langages de programmation exécute une instruction ou un bloc d'instructions **tant que la *condition* est vérifiée ;**
- le bloc **« répéter jusqu'à »** exécute une instruction ou un bloc d'instructions **jusqu'à ce que la *condition* soit vérifiée.**

Les élèves seront donc amenés à utiliser des conditions d'arrêt avec deux formulations différentes, une première forme sous scratch et une formulation qui s'apparente à son contraire dans la plupart des autres logiciels de programmation.

Sur l'exemple ci-contre, la programmation « répéter jusqu'à x inférieur à y » de Scratch devient « tant que x est inférieur ou égal à y … »

Scratch	Python

	`from math import *`
	`x=25`
	`y=3`
	`n=0`
	`while x>=y:`
	`    x=x-y`
	`    n=n+1`
	`print(n)`

b) la structure « répéter…fois » de scratch et la structures « Pour » (ou for) des langages textuels :

Cette transition est plus délicate car les élèves ont toujours beaucoup de difficultés à comprendre à quoi sert le compteur. En effet, ils se posent les questions suivantes :

- À quel moment ce compteur est-il incrémenté ? En début de structure Pour ou à la fin ?
- De combien est-il incrémenté à chaque fois ? Est-ce toujours un entier ?
- À quel moment sort-on d'une structure Pour ? Etc ….

Dans l'exemple ci-dessous le compteur est utilisé dans le calcul de l'instruction contenue dans la structure répétitive. Avec Scratch, on est obligé d'ajouter l'instruction de l'incrémentation. Avec Python ou Processing c'est inutile.

Scratch	Python	Processing

	`from math import *`
	`i=0`
	`total=0`
	`for i in range(10):`
	`  total=total+i*i`
	`print(total)`
		Total=0 ;
		for(int i=0;i<10;i++)
		{total=total+i*i;}
		println(total) ;

Problème n°1 : un tracé de fils?

La variable « k » est une variable qui représente un nombre entier.

Expliquer ce que fait ce programme ?

. . . . . .

. . . . . .

. . . . . .

Il semble que cet algorithme ne se termine pas au point $(200;-150)$.

Modifiez-le pour que le dernier segment d'extrémités (0;100) et (200 ;-150) soit tracé.

. . . . . .

. . . . . .

Appelez le professeur pour validation.

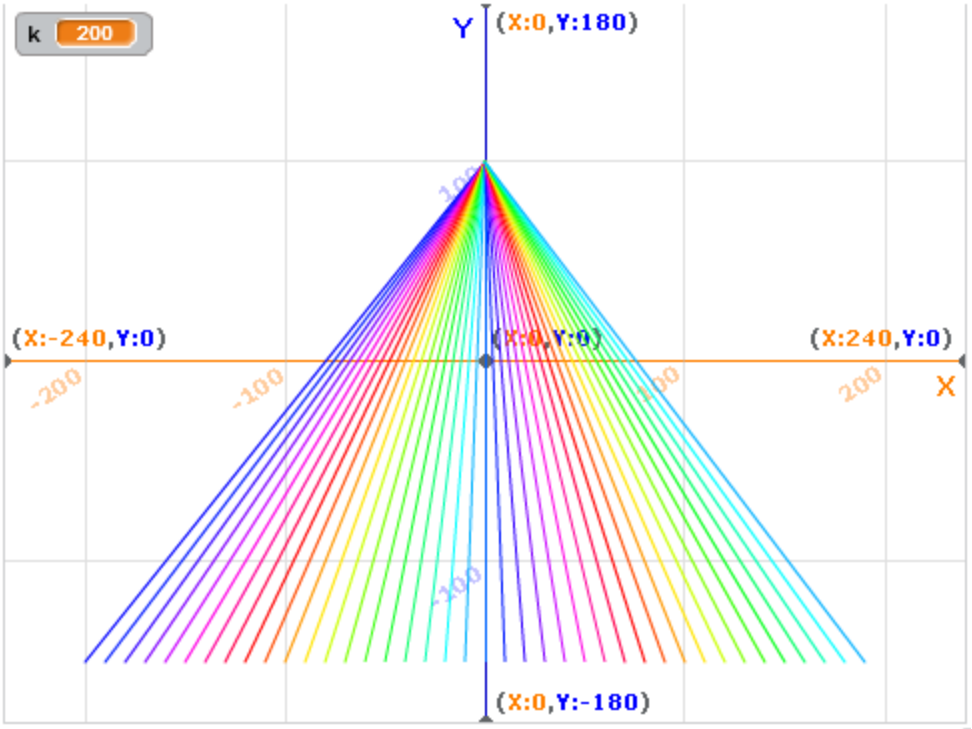
Problème n°2 : un nouveau tracé de fils.

On utilise la même base de programme. On souhaite maintenant tracer les « fils » du point (-75 ;-150) au point $(55;-150)$ avec un pas de 5. Modifier l'algorithme en conséquence.

Appelez le professeur pour validation.

Voici un programme d'une structure filaire à la calculatrice TI 82.
Ces structures sont indispensables pour pouvoir résoudre les problèmes demandés en utilisant la calculatrice (c'est ce que les élèves ont à disposition le jour du bac).

```
PROGRAM:FIL
:EffDess
:For(K,-4,5,1)
:Ligne(0,K,K,K)
:End
```



Conclusion et perspectives

À l'issue de la réflexion conduite cette année dans ce groupe action formation, quatre idées clés nous semblent importantes à retenir :

Idée 1 : Une progression par notions ne facilite pas l'entrée dans l'algorithmique et ne respecte pas les démarches préconisées par le programme pour cet enseignement. Il convient de penser plutôt la progression en termes de complexité du programme.

Idée 2 : Les enjeux de l'enseignement de l'algorithmique vont bien au-delà de l'évaluation qui peut être faite sur ce thème au brevet, évaluation contrainte par les conditions de l'examen et qui, même dans ce cadre, évoluera au fur et à mesure du déploiement de l'algorithmique au collège. C'est un enjeu de formation du futur citoyen de demain.

Idée 3 : L'enseignement de l'algorithmique s'inscrit avant tout dans une démarche de projet active et collaborative qui permet de développer les compétences de la résolution de problème, la prise d'initiative, l'autonomie et la créativité.

Idée 4 : On ne peut pas imaginer un enseignement de l'algorithmique qui ne s'inscrit pas dans une articulation collège-lycée. Il est donc important de connaître les attendus du lycée pour un professeur de collège, de connaître les modalités de l'enseignement de l'algorithmique au collège pour un professeur de lycée, de prendre appui sur ce que maitrisent les élèves au collège pour démarrer cet enseignement en lycée.

Perspectives :

L'enseignement de l'algorithmique contribue à préparer les jeunes à vivre dans le monde de demain.

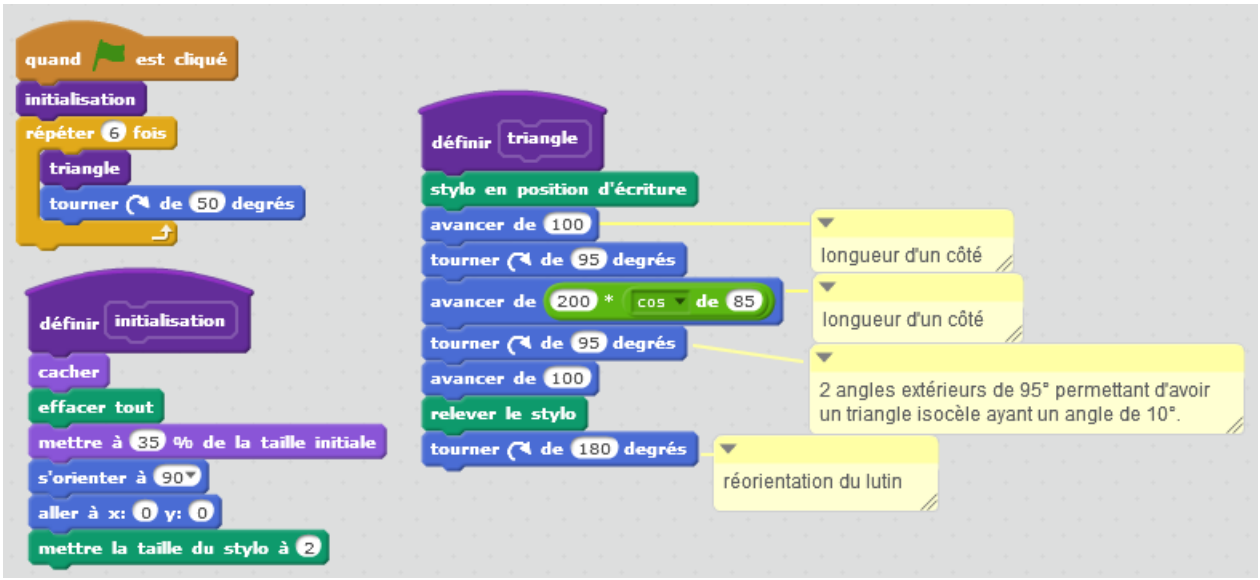
Le numérique a transformé, transforme et continuera à transformer de nombreux métiers et des habitudes de vie. Son enseignement ouvre vers les débouchés et les besoins scientifiques de demain.

Annexe 0 : devoir commun 4e

Exercice 6 :

1ère partie :

1) Tracer un triangle ABC isocèle en A tel que AB = 5 cm et $\widehat{BAC} = 10°$.
2) Voici un programme réalisé avec le logiciel Scratch. Il n'est pas demandé de comprendre la totalité des instructions de ce script.

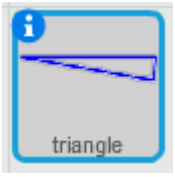


a) Le triangle défini dans ce script est isocèle.
Quelle longueur a été donnée aux deux côtés égaux ?

b) Quand on clique sur le drapeau vert, combien de triangles seront tracés à l'écran ?

2e partie :

Le programme ci-dessous est proposé avec un lutin triangle isocèle ayant un angle de 10° :



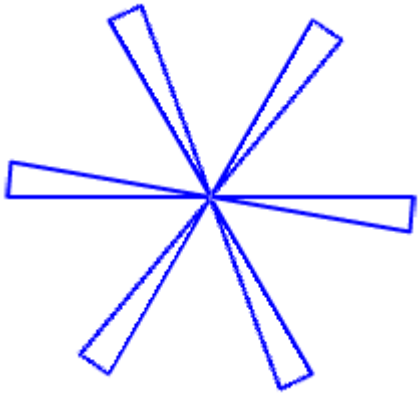
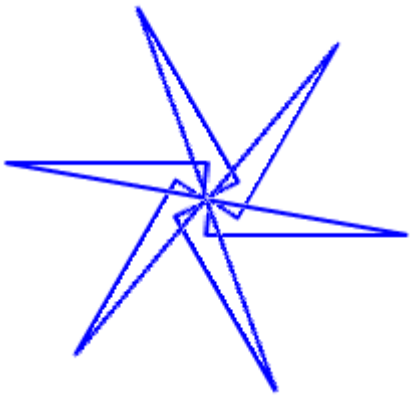
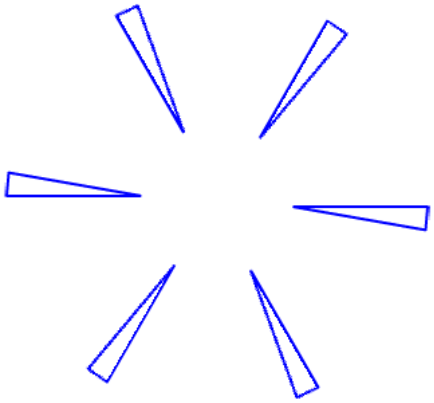
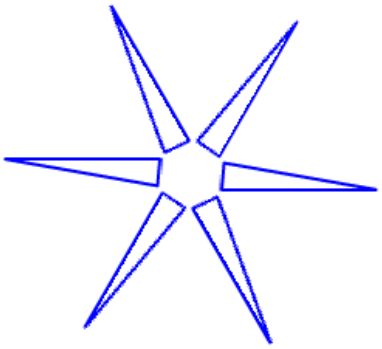
Les quatre figures suivantes ont été réalisées avec ce programme et ce triangle.
Quelle modification a dû être apportée pour obtenir ces figures différentes ?

Remarque : l'élément estampiller fonctionne comme un tampon encreur. Par exemple :

Ce programme

donne ceci

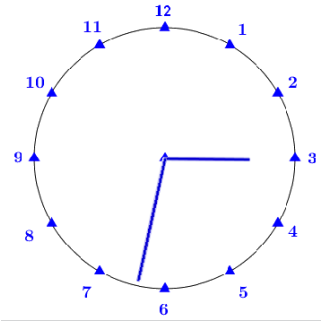
Il y a les 5 lutins chats reproduits plus le lutin lui-même.

Figure 1 	Figure 2 
Figure 3 	Figure 4 

Annexe 1 : régulation en 4e

Réaliser une montre sur Scratch *(voir scénario complet sur le site académique)*

1) Découverte de l'activité



L'activité consiste à utiliser 3 lutins :
- un lutin cadran,
- un lutin « grande aiguille »,
- un lutin « petite aiguille ».

Lorsque l'heure est indiquée par l'utilisateur, les aiguilles doivent tourner de façon visible pour se positionner à l'heure indiquée.

➔ **Analyser la difficulté et le degré de complexité de ce qui est demandé.**

Cela présente plusieurs difficultés. Par exemple : comment récupérer l'heure ? Sous quel format ? Comment faire tourner les aiguilles ? Les aiguilles doivent-elles avoir des spécificités pour que la rotation soit correcte ? …

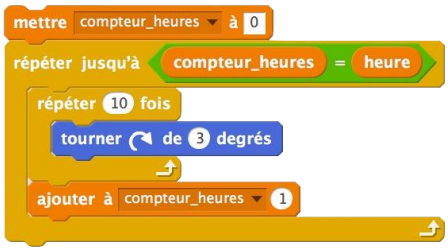
2) Analyse des premières difficultés et début de la programmation

En étant confronté à l'ensemble du programme, on se rend compte qu'il est très délicat, voire très difficile, de le mener à bien de façon immédiate et simple.

➔ **Décomposer le problème en sous-problèmes afin de structurer le programme.**

Cela permet :
- de démarrer l'activité ;
- d'avancer peu à peu en améliorant au fur et à mesure des étapes ;
- de comprendre en profondeur ce qui est demandé.

Il est convenu de commencer par :
- importer les 3 lutins et réaliser l'initialisation, c'est-à-dire programmer la position initiale de chaque lutin afin de pouvoir démarrer le programme ;
- ne s'occuper que d'une aiguille à la fois dans un premier temps ;
- demander l'heure du moment en séparant en deux temps : heures d'une part, minutes d'autre part.



exemple de script pour le lutin « aiguille des heures »

3) Perspectives

Une fois le programme commencé, élément par élément, on peut chercher des pistes d'amélioration.

➔ Chercher des évolutions permettant d'améliorer le programme

Deux pistes d'amélioration sont envisagées :
- obtenir un visuel dynamique qui donnera plus d'esthétique au rendu ;
- anticiper sur des erreurs de manipulation de l'utilisateur pour que le programme fonctionne malgré tout.

Annexe 2 : régulation en 3e

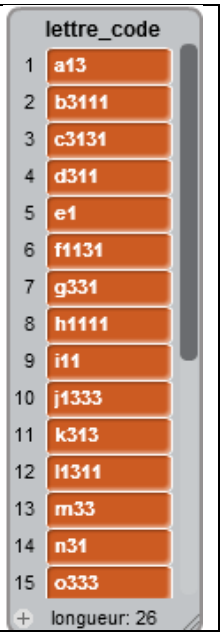
Code morse : bilan intermédiaire

Après deux séances en salle informatique, plusieurs groupes ont avancé sur une base de programme : il s'agit bien d'une version simplifiée qui pourra être améliorée peu à peu. Dans ce qui est précisé sur cette feuille, le programme demande une lettre et la code en morse en traits courts ou longs.

En général, deux listes ont été créées :

- l'une avec les lettres de l'alphabet (cette liste peut être remplacée par une variable : abcdefkhijklmnopqrstuvwxyz)
- l'autre avec le code morse codé ! Le trait court est remplacé par 1 et le trait long par 3.
  Deux raisons à cela : en morse, un trait long correspond à trois traits courts ; de plus, il est facile de manipuler des nombres en informatique.

Une autre option aurait été de faire une seule liste comme dans l'extrait ci-contre.



Il s'agit alors de décomposer le travail à réaliser en sous-programmes. Ce qui pourrait donner le programme principal ci-contre.

L'initialisation a été bien travaillée.
Les deux autres sous-programmes montrent les deux étapes qui suivent :
1/ repérer quelle est la lettre choisie par l'utilisateur

2/ une fois que cela est fait, lui associer son code morse.

Par exemple :

On remarquera l'usage de la boucle « répéter … jusqu'à …»

Annexe 3 : bilans élèves

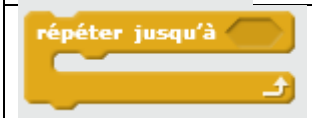
Quels blocs "importants" avez-vous utilisés ?	A quoi servent-ils ?	A quoi faut-il faire attention dans leurs utilisations ?
"Cacher" "Montrer"	ILS servent à cacher et à montrer le lutin.	
"chrono"	Il sert à savoir notre temps de réaction	
~~"toucher espace"~~ "presser ?"	Il sert à savoir quand l'on a vu le lutin apparaître	

Quels blocs "importants" avez-vous utilisés ?	A quoi servent-ils ?	A quoi faut-il faire attention dans leurs utilisations ?
cacher	à cacher le lutin	s'en attend au pas
REMETTRE le chronomètre	à REDÉMARRER le chronomètre	qu'il le fait bien qu'il remet bien à 0

Quels blocs "importants" avez-vous utilisés ?	A quoi servent-ils ?	A quoi faut-il faire attention dans leurs utilisations ?
Attendre 10 sec	pour le cacher	
nombre aléatoire	pour la surprise	
chronomètre	pour voir le temps	mettre ~~le chronomètre~~ sur bon moment
quand presser espace	pour voir notre temps de réaction	

Quels blocs "importants" avez-vous utilisés ?	A quoi servent-ils ?	A quoi faut-il faire attention dans leurs utilisations ?
les blocs "montrer et "cacher"	Ils servent à montrer et non le lutin	
attendre jusqu'à	Il sert à pouvoir attendre une donnée comme ... presser"	ne pas le confondre avec "si ... alors ..."

Annexe 4 :

La variable		
La boucle		
		
		
Le test conditionnel		
		
Le sous-programme