REPERES - IREM. N° 116 - juillet 2019

UNE ANALYSE DES EXERCICES D'ALGORITHMIQUE ET DE PROGRAMMATION DU BREVET 2017

Commission inter-Irem
Informatique (C3i)(*)

Ce texte est également consultable en ligne sur le portail des Irem, onglet : Repères IREM http://www.univ-irem.fr/

Résumé : Nous proposons une analyse critique de l'ensemble des exercices de l'épreuve de mathématiques 2017 du brevet ayant trait au thème « Algorithmique et programmation » du programme de cycle 4. Certains de ces exercices ne mettent en jeu que de manière très superficielle des compétences spécifiquement informatiques, mais portent plutôt sur des compétences mathématiques « traditionnelles », liées par exemple à l'algèbre ou la géométrie. Notre analyse met en regard ces différentes compétences, et propose des adaptations de certains énoncés afin d'en permettre une exploitation plus riche.

1. — Introduction

Nous proposons une analyse critique des exercices de mathématiques des épreuves du Diplôme National du Brevet (DNB) de 2017 relevant du domaine "algorithmique et programmation", thème nouvellement introduit à la suite des changements des programmes officiels du cycle 4 survenus en 2016. Nous avons également analysé les sujets de DNB de mathématiques et de technologie du centre d'examen de Pondichéry de 2018, l'informatique intervenant conjointement dans ces deux disciplines scolaires.

Certains de ces exercices ne mettent en jeu que de manière très superficielle des compétences spécifiquement informatiques, mais por-

* Liste des co-auteurs : Sylvie Alayrangues (Irem de Poitiers), Emmanuel Beffara (Irem de Marseille), Sébastien Daniel (Irem de Lorraine), Christophe Declercq (Irem de Nantes), Anne Héam (Irem de Besançon), Jean-Vincent Loddo (Irem de Paris Nord), Philippe Marquet (Irem de Lille), Jean-Christophe Masseron (Irem de Paris), Antoine Meyer (Irem de Paris), Malika More (Irem de Clermont-Ferrand), Florence Nény (Irem de Marseille), Vincent Pantaloni (Irem d'Orléans), Gaëtan Perrin (Irem de Clermont-Ferrand), Cécile Prouteau (Irem de Paris), Georges Saliba (Irem de Bordeaux), Sylviane Schwer (Irem de Paris Nord), Fabien Tarissan (Irem de Paris), Chloé Ubera (Irem de Bordeaux), Jean-Marc Vincent (Irem de Grenoble), Emmanuel Volte (Université de Cergy-Pontoise)

tent plutôt sur des compétences mathématiques « traditionnelles », en particulier liées à l'algèbre ou la géométrie. D'autres requièrent de la part de l'élève une connaissance spécifique de certaines fonctionnalités de l'environnement Scratch, ceci en dépit du fait qu'aucun langage de programmation particulier n'est imposé aux enseignants. Il semble donc *de fait* indispensable à un enseignant de familiariser les élèves à Scratch afin de préparer l'épreuve du DNB.

Notre analyse de ces exercices expose et met en regard ces différentes connaissances et compétences; elle suggère d'adapter certains de ces exercices pour en permettre une exploitation en classe favorisant le travail ou l'évaluation des compétences spécifiques à ce thème. Cette analyse est susceptible d'intéresser aussi les formateurs et les candidats au CRPE[1] et au CAPES[2] dans la mesure où ces concours comportent des questions portant sur les compétences mises en jeu par de tels exercices.

Pour chaque exercice, ce document comporte une version de l'énoncé reproduite d'après les documents mis à disposition par Denis Vergès sur le site de l'APMEP[3] [2], suivie d'une mise en perspective. Pour certains exercices, nous proposons une ou plusieurs alternatives pour une utilisation en classe. Nous regroupons les divers exercices de brevet contenant de l'informatique selon quatre thèmes principaux : programmes de calcul, tracé de figures (frises et rosaces puis spirales et formes similaires), exercices traitant de programmation événementielle (dont l'exécution dépend des actions de l'utilisateur), et enfin simulations en probabilités.

1 CRPE : Concours de recrutement des professeurs des écoles.
2 CAPES : Certificat d'aptitude au professorat de l'enseignement du second degré.
3 APMEP : Association des professeurs de mathématiques de l'enseignement public.

Dans l'ensemble de cet article, le lecteur trouvera des programmes rédigés en Scratch 2. Une présentation rapide de cet environnement se trouve en annexe.

2. — Programmes de calcul

En enseignement des mathématiques, on appelle *programme de calcul* une procédure composée d'une suite d'opérations arithmétiques (choisir un nombre, le multiplier par 3, ajouter 2 au résultat...). Ce concept est utilisé en particulier pour enseigner le calcul littéral, la notion de fonction et la résolution d'équations. Il permet une entrée progressive dans la compréhension des expressions algébriques pour leur aspect procédural et structural en lien avec les priorités opératoires.

L'usage courant met davantage l'accent sur la tâche *produire une expression littérale* à partir d'un programme de calcul, que sur la réciproque, *rédiger un programme de calcul* à partir d'une expression littérale. Cette dernière activité, qui relève d'une démarche semblable à celle de *l'analyse syntaxique* en informatique, permet de mettre en évidence, en plus du caractère procédural des expressions numériques ou algébriques, l'aspect *structural* de ces expressions. Il nous semblerait pertinent de développer ce point, en particulier dans le contexte d'un enseignement d'algorithmique.

La notion de variable ne recouvre pas le même concept en mathématique et en informatique. Les points de divergence entre les deux, et les spécificités de la variable informatique seront présentés en détails dans un article en préparation par la C3i. Étant donné cette fausse proximité, enseigner ces deux notions de variable de façon concomitante est un exercice difficile voire périlleux. C'est pourquoi nous recommandons que les exercices mettant en relation programme de calcul et informatique soient utilisés

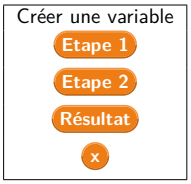
avec précaution et qu'une attention particulière soit portée à cette question dans la formulation de l'énoncé. Ce n'est pas toujours le cas dans les trois exercices analysés, mais nous proposons des variantes qui nous paraissent plus satisfaisantes sur ce point.

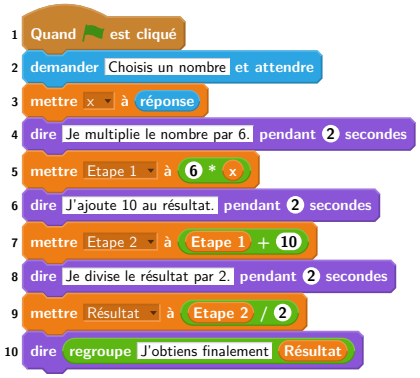
Dans les trois sujets étudiés ci-dessous, l'aspect informatique semble avoir été plaqué artificiellement sur un exercice classique de type *programme de calcul*. Leur point commun est d'être très pauvres sur le plan de l'algorithmique et de la programmation. En particulier, la simple traduction en langage Scratch d'un programme de calcul habituel ne contient ni condition ni boucle. Soulignons que les programmes de calcul sont des outils performants pour l'introduction de l'algèbre mais, malgré leur nom, ils ne présentent en eux-mêmes que très peu d'intérêt du point de vue de l'algorithmique et de la programmation. Par conséquent, les questions posées dans les énoncés suivants ne mobilisent pas vraiment les connaissances et savoir-faire spécifiques à ce thème.

2.1 *Pondichéry*, 2 *mai* 2017

2.1.1 *Énoncé (exercice* 3, *noté* 7 *points sur* 50*)*

On considère le programme de calcul ci-contre dans lequel x, Etape 1, Etape 2 et Résultat sont quatre variables.





1. (a) Julie a fait fonctionner ce programme en choisissant le nombre 5. Vérifier que ce qui est dit à la fin est : « J'obtiens finalement 20 ».

   (b) Que dit le programme si Julie le fait fonctionner en choisissant au départ le nombre 7 ?
2. Julie fait fonctionner le programme, et ce qui est dit à la fin est : « J'obtiens finalement 8 ». Quel nombre Julie a-t-elle choisi au départ ?
3. Si l'on appelle x le nombre choisi au départ, écrire en fonction de x l'expression obtenue à la fin du programme, puis réduire cette expression autant que possible.
4. Maxime utilise le programme de calcul ci-contre :

   Peut-on choisir un nombre pour lequel le résultat obtenu par Maxime est le même que celui obtenu par Julie ?

- Choisir un nombre.
- Lui ajouter 2.
- Multiplier le résultat par 5.

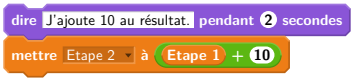
2.1.2 *Commentaires*

L'intention affichée de cet exercice est de questionner le fonctionnement de la transcription en Scratch d'un programme de calcul. Nous pouvons distinguer des instructions de deux natures différentes : des instructions d'affichage en mauve et des instructions d'affectation de variables en orange. Il semble qu'au-delà de cette concomitance, cet exercice, dans sa formulation, permet d'éviter de tester la compréhension de l'attendu affiché par le sujet « questionner le déroulement des instructions d'affectation ». En effet la simple lecture des instructions d'affichage en bleu permet de répondre à l'ensemble des questions sans s'interroger sur le contenu des variables présentes dans l'énoncé et le programme.

D'autre part, les différentes variables qui apparaissent dans l'exercice et leurs noms illustrent les difficultés liées à l'enseignement simultané des notions de variables mathématiques et informatiques au collège.

Le programme utilise quatre variables informatiques, dont trois portent des noms riches, et une s'appelle seulement x. On note que ce *x* est le nom de la variable mathématique utilisée dans l'exercice. Les variables Etape 1 et Etape 2 contiennent des résultats intermédiaires, ce que leur nom, bien que riche, ne rend pas explicite. Peut-être aurait-il été préférable de choisir par exemple Résultat 1, Résultat 2 et Résultat final, ou même de réutiliser plusieurs fois une même variable Résultat, ce qui permettrait en plus de vérifier la compréhension de la notion de variable informatique.

En outre, les instructions « dire » intermédiaires décrivent le calcul à la façon d'un programme de calcul ordinaire. En particulier, elles utilisent le mot « résultat » pour faire référence au résultat intermédiaire précédent :

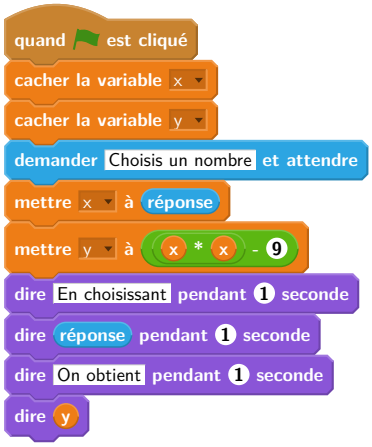


Ceci peut de nouveau entraîner de la confusion vis-à-vis des variables informatiques utilisées, qui portent des noms différents (Étape 1, etc.).

2.2 *Polynésie*, 14 *septembre* 2017

2.2.1 *Énoncé* (*exercice* 6, *noté* 6 *points sur* 50)

La figure ci-après est la copie d'écran d'un programme réalisé avec le logiciel « Scratch ».



1. Montrer que si on choisit 2 comme nombre de départ, alors le programme renvoie – 5.
2. Que renvoie le programme si on choisit au départ :
   (a) le nombre 5 ?
   (b) le nombre – 4 ?
3. Déterminer les nombres qu'il faut choisir au départ pour que le programme renvoie 0.

2.2.2 *Commentaires*

Cet exercice peut être associé à un programme de calcul ou à la production de l'expression de la fonction qui à x associe $x^2 - 9$. Le choix du nom de variable 

. nous fait tendre vers le domaine des fonctions. Le programme permet alors de calculer l'image de la valeur saisie par l'utilisateur (récupérée dans la variable 

puis mémorisée dans

) et de la mémoriser dans la variable 

.

Du point de vue mathématique, les deux premières questions demandent de calculer trois images par la fonction f . Du point de vue informatique, c'est la connaissance de la manière dont on programme une expression arithmétique combinée qui permet d'identifier l'expression produite (l'opérateur binaire de multiplication imbriqué dans celui de la soustraction).

Quant à la dernière question, elle demande de résoudre l'équation $x^2 - 9 = 0$ et d'ainsi identifier les antécédents de 0 par la fonction f. L'élève peut soit procéder par essai-erreur, soit résoudre l'équation équivalente $x^2 = 9$ ou encore factoriser l'expression $x^2 - 9$ en $(x - 3)(x + 3)$. Le saut de difficulté est intrinsèque aux mathématiques.

Concernant la formulation des questions de cet exercice, les auteurs utilisent l'expression *ce que renvoie le programme* pour faire référence au contenu de la variable 

, ou à ce que l'exécution de ce programme affiche. Cela n'est pas judicieux.

Du point de vue informatique, la seule compétence évaluée est l'exécution à la main d'un algorithme, et dans une très faible mesure seulement puisque l'algorithme proposé ne contient ni condition ni boucle. Pour finir, les instructions 

et 

relèvent de la connaissance des spécificités du langage Scratch et n'apportent rien à l'exercice puisque la scène n'est pas visible.

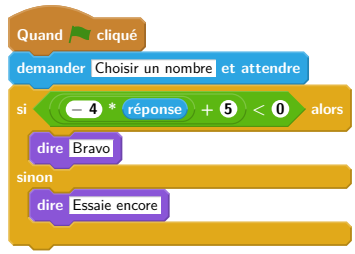
2.3 *Polynésie, 23 juin* 2017

2.3.1 *Énoncé* (*exercice* 5, *noté* 7 *points sur* 50)

On considère le programme de calcul suivant :

- Choisir un nombre ;
- Le multiplier par – 4 ;
- Ajouter 5 au résultat.

1. Vérifier que lorsque l'on choisit – 2 avec ce programme, on obtient 13.
2. Quel nombre faut-il choisir au départ pour obtenir – 3 ?
3. Salomé fait exécuter le script suivant :



   (a) Quelle sera la réponse du lutin si elle choisit le nombre 12 ?

   (b) Quelle sera la réponse du lutin si elle choisit le nombre – 5 ?

4. Le programme de calcul ci-dessus peut se traduire par l'expression littérale $-4x + 5$ avec x représentant le nombre choisi. Résoudre l'inéquation suivante :
$$-4x + 5 < 0$$
5. À quelle condition, portant sur le nombre choisi, est-on certain que la réponse du lutin sera « Bravo » ?

2.3.2 *Commentaires*

Cet exercice a le mérite de faire travailler le lien entre un programme de calcul, effectuant une succession d'opérations une par une, et une expression littérale équivalente. Cependant, le travail de transcription proprement dit n'est pas à la charge de l'élève ici. De plus, il est demandé à l'élève d'admettre l'équivalence entre le programme de calcul et l'expression $-4x + 5$ à la question 4, alors que cette information aurait pu servir à justifier la création du programme Scratch de la question 3.

On remarque une nette séparation entre les questions portant sur les aspects algébriques (questions 1, 2 et 4) et les questions portant sur la programmation (3 et 5). L'originalité du programme Scratch proposé est de ne pas constituer une transcription exacte du programme de calcul, du fait de la présence d'une expression composée et d'un bloc conditionnel. Malgré tout, comme dans la majorité des exercices du DNB 2017, le niveau de connaissances et compétences évalué en algorithmique et programmation reste très modeste (compréhension élémentaire du fonctionnement général de Scratch, du bloc conditionnel, du bloc « demander », etc.).

Plus problématique, et typique de certains exercices portant sur des programmes, la formulation de la question 5 n'interdit pas a priori une réponse triviale du type « la réponse sera Bravo si $-4 *$ réponse $+ 5 < 0$ », ce qui n'est évidemment pas la réponse attendue.

Une reformulation possible serait : « En déduire les nombres pour lesquels la réponse du lutin sera "Bravo". », mais cela reste peu ambitieux en termes de niveau de connaissances mis en œuvre.

2.4 *Propositions d'exercices*

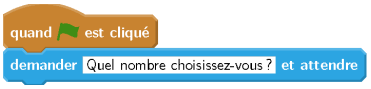
2.4.1 *Traduire un programme de calcul en Scratch*

On fournit des blocs individuels, numérotés, chacun réalisant une affectation à une variable Résultat, et un programme de calcul en français. Le but de l'exercice est de composer un programme Scratch équivalent au programme de calcul.

Exemple d'énoncé de type brevet. Thomas utilise le programme de calcul ci-dessous :

- Choisis un nombre ;
- Multiplie-le par 3;
- Ajoute 2 au résultat.

1. Thomas choisit 2, quel résultat obtient-il ?
2. Leïla décide de traduire ce programme de calcul en langage Scratch, pour vérifier le calcul de Thomas. Voici le début de son programme :



On rappelle qu'après l'exécution de ces deux blocs, la variable réponse contient une valeur saisie au clavier. Ordonnez et complétez les blocs suivants pour aider Leïla à terminer son programme :

Notions travaillées :

— Programmation : notion de variable informatique, d'expression, d'affectation, séquences d'instructions.

— Mathématiques : utiliser un programme de calcul.

Variantes : En salle machine, il est possible de proposer le même exercice sans fournir les blocs de base. Une autre variante, également adaptée à un contrôle sur table, consiste à proposer plusieurs programmes en Scratch et à demander à l'élève de choisir celui qui correspond au programme de calcul.

2.4.2 *Décomposer une expression arithmétique en un programme de calcul*

Ce type d'exercice consiste à fournir à l'élève une expression littérale contenant des nombres entiers, plusieurs opérations arithmétiques, éventuellement des parenthèses et une indéterminée x. La consigne est de concevoir un programme Scratch calculant la valeur de l'expression pour toute valeur choisie de x. L'énoncé peut éventuellement fournir une liste de blocs individuels à ordonner, ou une liste de programmes dont il faut déterminer lequel résout le problème.

Il est à noter que même si la décomposition du programme de calcul en calculs « élémentaires » n'est pas suggérée ou imposée par l'énoncé, l'interface de Scratch impose néanmoins à l'élève-programmeur de choisir et de combiner les opérations arithmétiques de façon appropriée, ce qui correspond bien au savoir-faire visé par cet exercice.

Réciproquement, on peut fournir un programme Scratch réalisant un calcul complexe et demander à l'élève de rédiger le programme de calcul correspondant, ou de décomposer le programme en opérations élémentaires, ou encore de dessiner la structure de l'expression sous la forme d'un arbre[4] .

Notions travaillées :

— Programmation : notion de variable informatique, d'expression, d'initialisation.

— Mathématiques : identifier la nature de l'expression (addition, multiplication...), ce qui revient à identifier la racine de l'arbre syntaxique de l'expression; connaître la priorité des opérations; notion d'évaluation d'une expression.

2.4.3 *Résoudre une équation par différentes méthodes*

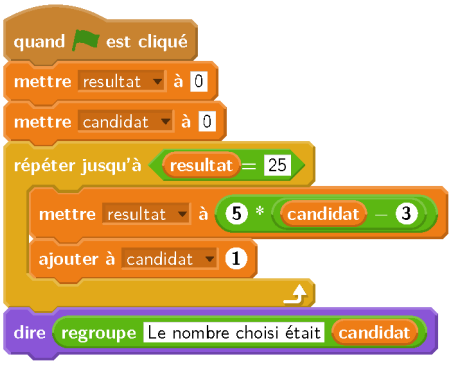
Ce type d'exercice vise à faire travailler la résolution d'équations du premier degré par différentes méthodes : essai / erreur, approche de type dichotomie, recherche exhaustive (sous l'hypothèse d'existence de solution entière), passage par la fonction réciproque, méthode algébrique. On fournit une expression littérale contenant des nombres entiers, plusieurs opérations arithmétiques, éventuellement des parenthèses et une variable x. La consigne est de concevoir un programme Scratch permettant de déterminer la valeur de x pour laquelle l'expression prend une valeur donnée.

Exemple. Ryan utilise le programme de calcul suivant :

- Choisis un nombre;
- Soustrais 3 au nombre choisi;
- Multiplie par 5 le résultat précédent.

4. Ce type d'arbre, dont chaque nœud interne correspond à un opérateur et dont chaque feuille correspond à un littéral (nombre ou variable), est appelé arbre syntaxique de l'expression ou du calcul. C'est un objet important, qui intervient par exemple dans l'analyse du texte d'un programme informatique en vue de son exécution.

1. Ryan obtient 25, quel était le nombre départ ?
2. Mathilde a écrit le programme Scratch ci-dessous pour retrouver le résultat de Ryan.



Mathilde obtient 9, et dit à Ryan qu'il s'est trompé. Qui a raison ? Justifier.

3. Jérémie cherche de son côté, il propose le bloc suivant pour résoudre le problème directement :

Compléter ce bloc.

4. On donne un programme de calcul avec des paramètres *a* et *b* du type :

   - Choisis un nombre ;
   - Multiplie par *a* le nombre choisi ;
   - Ajoute *b* au résultat précédent.

(a) Écrire un programme qui calcule le résultat à partitr d'un nombre choisi.

(b) Écrire un second programme qui, lorsque l'on donne le résultat du calcul, retrouve le nombre choisi au départ.

Variantes et prolongements : On peut décliner cet exercice en choisissant d'autres variables didactiques (nombres décimaux, entiers négatifs, nombres rationnels) afin de différencier, et de travailler l'ensemble des techniques de résolution possibles.

Par exemple, l'approche proposée dans la question 2 ne fonctionne que si la solution recherchée est un entier strictement positif. Il faut donc envisager les cas où :

— la solution recherchée est un entier négatif – dans ce cas il faudra concevoir un autre type de boucle ;

— la solution recherchée n'est pas entière – dans ce cas, il faudra modifier la condition d'arrêt, et rechercher un encadrement de la solution, avant éventuellement de rechercher un encadrement plus précis.

3. — Tracé de figures : Frises et rosaces

La compétence d'exécution à la main d'un algorithme est évaluée ici sur des cas plus intéressants que dans les exercices de type « Programmes de calcul », puisque les programmes proposés dans ces exercices contiennent des boucles. Par ailleurs, ces programmes utilisent des blocs personnalisés (blocs violets du menu « Ajouter blocs »), ce qui a probablement posé quelques difficultés aux élèves qui ne les avaient pas forcément vus en classe. Par contre, ils n'utilisent aucune variable informatique.

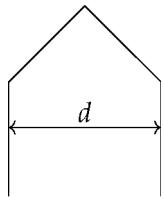
Il est à noter que dans les deux exercices présentés ci-dessous, le dernier bloc « tourner » sert à remettre le lutin dans son orientation initiale, ce qui est indispensable dans le cadre du programme complet. Ce bloc ne joue aucun rôle sur le dessin de la maison elle-même. Il aurait pu être intéressant de poser une question à propos de l'état du système avant et après l'exécution du bloc (nous détaillons ce point au paragraphe 3.4.1).

La question du contexte d'exécution (ou état du système) est délicate en Scratch. Elle recouvre par exemple la position et l'orientation du lutin, l'état du stylo (couleur, intensité, épaisseur, position d'écriture ou pas), le fait que chaque lutin soit caché ou pas, etc. Cette question est encore plus problématique dans le cadre d'exercices sur papier que lorsque les élèves travaillent sur machine. En effet, ni la position ni l'orientation initiales du lutin ne sont immédiatement visibles. C'est pourquoi il est particulièrement important que l'énoncé soit le plus précis possible sur ces points. Des efforts en ce sens sont faits mais c'est encore améliorable.

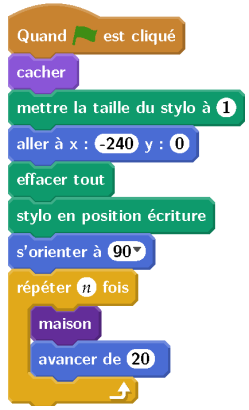
3.1 *Centres étrangers*, 17 *juin* 2017

3.1.1 *Énoncé* (*exercice* 6, *noté* 9 *points sur* 50)

Pour tracer une « rue », on a défini le tracé d'une « maison ».

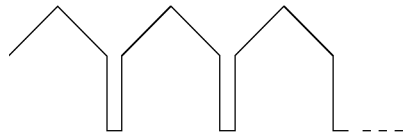


Tracé de la « maison »



Programme principal

1. Vérifier que d est environ égal à 71 à l'unité près.
2. Un point dans une fenêtre d'exécution de votre programme a son abscisse qui peut varier de – 240 à 240 et son ordonnée qui peut varier de – 180 à 180. Quel est le plus grand nombre entier n que l'on peut utiliser dans le programme principal pour que le tracé de la « rue » tienne dans la fenêtre de votre ordinateur où s'exécute le programme ?

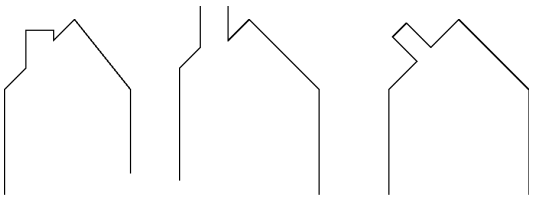


Vous pourrez tracer sur votre copie tous les schémas (à main levée ou non) qui auront permis de répondre à la question précédente et ajouter toutes les informations utiles (valeurs, codages, traits supplémentaires, noms de points...)

3. Attention, cette question est indépendante des questions précédentes et la « maison » est légèrement différente. Si on désire rajouter une sortie de cheminée au tracé de la maison pour la rendre plus réaliste, il faut faire un minimum de calculs pour ne pas avoir de surprises.

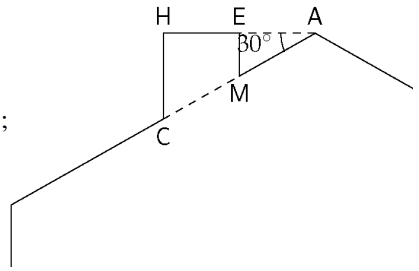
TNSVP

Exemples :



On suppose que :

- les points H, E et A sont alignés;
- les points C, M et A sont alignés;
- [CH] et [EM] sont perpendiculaires à [HA] ;
- AM = 16 ;
- MC = 10 ;
- $\widehat{HAC} = 30°$.



Calculer EM, HC et HE afin de pouvoir obtenir une belle sortie de cheminée.

3.1.2 *Commentaires*

— Dans la question 1, du point de vue mathématique, la figure n'est pas considérée pour son orientation et c'est le sens de cette première question. Cependant du point de vue informatique, la figure produite est dépendante de l'état du lutin.

Il aurait été sans doute pertinent de scinder cette question en trois : une première question portant sur l'état initial du lutin, que l'on peut déduire du programme principal, une deuxième demandant de dessiner le tracé produit par l'exécution du bloc « maison » sur une annexe comportant un repère et un état initial particulier, et une troisième portant sur le calcul de la distance *d*.

Le traitement de la première question relative à l'état initial du lutin relève d'une connaissance du langage Scratch puisque

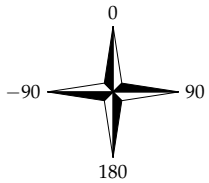
signifie que le lutin « regarde » vers la droite de la scène. Ce fait est d'ailleurs rappelé par le rédacteur dans d'autres sujets analysés dans ce texte.

— La question 2 fait appel à des connaissances à la fois informatiques et mathématiques, puisque la question « quel est le plus grand entier *n*... » nécessite d'avoir compris le fonctionnement d'une boucle. Ici, le contexte est spécifié dans le programme principal (position, direction et visibilité du lutin, taille et position du stylo).

Toutefois une clarification est nécessaire sur le fait que la dernière maison tracée doive être complètement dessinée. Une reformulation possible est « Quel est le nombre maximal de maisons complètes qui forment la rue dans la scène ? ».

— Concernant la question 3, il nous semblerait intéressant de proposer un bloc « cheminée » permettant de tracer la partie du toit

Nous avons défini un bloc cheminée. Les instructions ci-dessous permettent d'obtenir la figure de droite, qui représente le pan complet du « toit » sur lequel se trouve la cheminée. Quelles modifications de la définition du bloc maison permettent de produire la maison complète ?

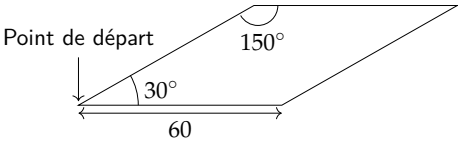


sur laquelle repose la cheminée, et de demander quelles modifications du bloc maison permettent ainsi de produire la maison avec une cheminée. Nous proposons ci-dessus une ébauche de question. L'adaptation proposée fait alors apparaître un découpage en sous-problèmes, et permet de questionner la capacité de l'élève à ordonner l'exécution des instructions du programme et l'état de la variable orientation du lutin.

3.2 *Wallis et Futuna, 2 décembre* 2017

3.2.1 *Énoncé* (*exercice* 4, *noté* 5 *points sur* 50)

1. On souhaite tracer le motif ci-dessous en forme de losange. Compléter sur l'annexe 1, le script du bloc Losange afin d'obtenir ce motif.



2. On souhaite réaliser la figure ci-contre construite à partir du bloc Losange complété à la question 1.

On rappelle que l'instruction 

signifie que l'on se dirige vers la droite.

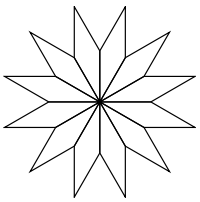
Parmi les instructions ci-dessous, indiquer sur votre copie, dans l'ordre, les deux instructions à placer dans la boucle pour finir le script ci-contre.

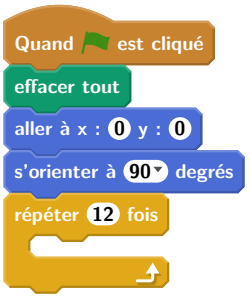
① 

② 

③ 

④ 

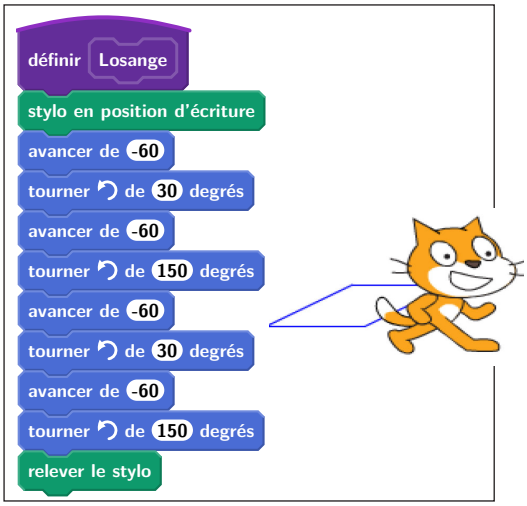




3.2.2 *Commentaires*

— Cet exercice comporte un travail mathématique intéressant sur les angles supplémentaires. En effet, les réponses attendues sont les valeurs des angles extérieurs à la figure. Ce type d'activité est une occasion de faire vivre ce vocabulaire, mais qui n'est pas exploitée par l'énoncé de brevet.

— Le schéma de losange présent dans l'énoncé ne précise pas l'orientation initiale du lutin. Par conséquent il serait normal d'admettre des réponses peu évidentes a priori, comme le script ci-contre, qui provoque le tracé situé à droite.



Pour éviter toute ambiguïté sur la réponse attendue, il paraît alors souhaitable que l'orientation initiale du lutin par rapport à la figure à tracer soit précisée.

— La question 2, demandant de compléter le script de production de la rosace, contient lui aussi une ambiguïté puisque le tracé

peut se faire indifféremment à l'aide d'une rotation de 30° ou de 150°. Un choix différent des blocs à insérer aurait été souhaitable. Il aurait également été intéressant de faire travailler l'élève sur le nombre de répétitions à utiliser.

Enfin, un travail de justification du choix de l'angle et du nombre de répétitions aurait permis de mettre à l'épreuve les connaissances géométriques et arithmétiques de base et la capacité d'argumentation de l'élève.

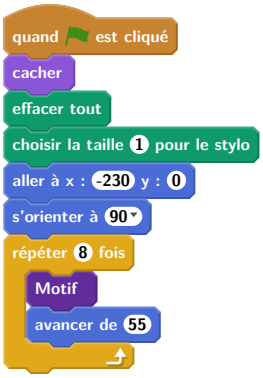
3.3 *Sujet zéro du* DNB

3.3.1 *Énoncé* (*exercice* 4)

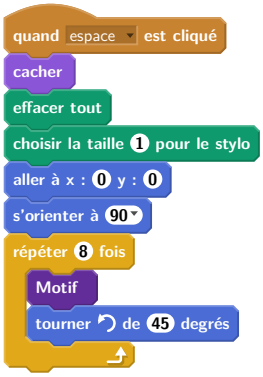
1. Pour réaliser la figure ci-dessus, on a défini un motif en forme de losange et on a utilisé l'un des deux programmes A et B ci-dessous. Déterminer lequel et indiquer par une figure à main levée le résultat que l'on obtiendrait avec l'autre programme.

Motif

Programme A



Programme B



2. Combien mesure l'espace entre deux motifs successifs ?
3. On souhaite réaliser la figure ci-dessous :

Pour ce faire, on envisage d'insérer l'instruction 

dans le programme utilisé à la question 1. Où faut-il insérer cette instruction ?

3.3.2 *Commentaires*

Une fois encore, répondre à la première question relève de connaissances sur l'environnement Scratch, ici, la scène dont les points ont des abscisses comprises entre – 240 et 240. Par ailleurs même si cette connaissance est absente, la présence du tourner de 45° dans la boucle permet elle aussi d'éliminer le programme B. L'absence complète de déplacement en dehors du tracé du motif est une indication forte. La discrimination des deux programmes ne peut alors attester de connaissances informatiques.

Par ailleurs, la formulation même de la question est problématique. En effet il y a une confusion entre le motif (le losange) produit et le nouveau bloc que l'on définit le bloc motif.

En ce qui concerne la question 3, une réponse de formulation vague telle que « dans la boucle » est acceptable puisqu'en effet, cette instruction peut être placée à n'importe quel endroit de la boucle (trois positions possibles). L'instruction peut aussi être placée au début ou à la fin du jeu d'instructions constituant la définition du bloc motif (avant ou après les blocs définissant l'état du stylo). Il y a alors au moins sept réponses correctes différentes à cette question. Cependant, positionner l'instruction dans la définition du bloc motif n'atteste pas nécessairement de la compréhension de la notion de boucle. En effet une formulation du type « chaque motif doit faire augmenter la taille du stylo » est suffisante.

3.4 *Propositions d'exercices*

Nous ne proposons pas ici d'énoncés d'exercices complets, mais souhaitons mettre l'accent sur le type de notions que les activités de ce genre permettent de faire travailler.

3.4.1 *Sensibilisation à la notion d'état*

L'environnement d'exécution d'un programme Scratch est très riche. L'ensemble de tous les paramètres de la scène, la présence, la position et l'orientation de chaque lutin, les réglages de chaque stylo, etc. sont autant d'éléments qui constituent ce que l'on nomme généralement *l'état* du système. L'état initial du système au moment où l'utilisateur clique sur le bouton « démarrer » détermine très souvent l'issue observable du programme. C'est donc un aspect essentiel du travail.

Un grand nombre de blocs de Scratch ont un effet perceptible sur l'état. En informatique, on parle d'*effets de bord*. Par exemple, après l'exécution d'une instruction 

, le lutin actif a en général changé de position (même s'il est invisible), et un segment a peut-être été tracé sur la scène.

Spécifier l'effet d'un bloc sur l'état. Dans les trois énoncés rencontrés dans cette section, des blocs personnalisés (ou *procédures*) sont utilisés pour décrire des tracés de figures géométriques, l'objectif final étant d'utiliser ces blocs dans des boucles afin de tracer des rosaces ou des frises. On est ici confronté à la notion de la *spécification* des effets d'un bloc sur l'environnement, ou de *contrat* que le bloc remplit (effets de bord). Ici, les deux blocs traçant des polygones (Losange, Motif) respectent au moins les deux conditions suivantes :

1. Le tracé de la figure prend en compte la position et la direction du lutin au moment où le bloc est appelé;
2. À la fin de l'exécution du bloc, le lutin est revenu à la même position (et orienté dans la même direction) qu'au début de l'exécution du bloc.

Ainsi, on pourra obtenir les effets voulus en faisant varier l'état entre deux appels successifs (par exemple on obtient une rosace en faisant varier la direction du lutin, ou une frise en faisant varier sa position). Par contre, le bloc maison utilisé dans l'énoncé 3.1.1 ne préserve pas la position du lutin. Il est indispensable de prendre cet aspect en compte dans la conception du script principal.

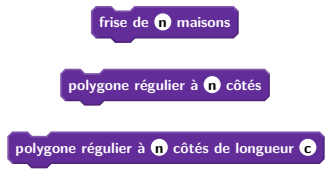
Une sensibilisation des élèves à ces problématiques liées à l'état de l'environnement d'exécution, et aux« contrats » qu'un programmeur associe aux éléments de ses programmes, nous paraît fondamentale. À cette fin, l'utilisation de commentaires (permise par Scratch) pourrait se révéler utile.

3.4.2 *Blocs à paramètres*

Même si les blocs personnalisés de Scratch ne sont pas, à proprement parler, des fonctions au sens usuel de ce mot en programmation, puisqu'ils ne fournissent pas un résultat mais produisent uniquement des effets de bord (comme nous l'avons décrit ci-dessus), ils permettent de commencer à initier les élèves à la notion de fonction, en introduisant notamment le concept de paramètre. Les activités tournant autour du dessin et de la géométrie sont un cadre idéal pour cela.

Par exemple, une activité pourrait proposer aux élèves de définir des blocs dessinant des carrés de côtés de différentes longueurs, et leur faire observer qu'il est fastidieux et source d'erreurs de reproduire le même morceau de programme plusieurs fois. L'activité proposerait ensuite d'introduire la notion de bloc à paramètre, et de simplement définir un bloc 

. Voici quelques autres exemples de blocs à paramètres qu'il est envisageable de définir avec les élèves :



4. — Tracé de figures : triangles, spirales, et autres carrés

Les exercices de ce type sont assez intéressants du point de vue informatique. Ils font appel à des compétences d'exécution et de modification d'algorithmes comportant des boucles et d'utilisation de variables informatiques.

Sur le plan de la rédaction, ils présentent cependant systématiquement une confusion entre une unité de longueur en Scratch et un pixel, qui ne sont pas identiques. Par exemple, lorsqu'on met la scène de Scratch en format "petite scène", elle contient autant d'unités que lorsqu'elle est en format normal. Il nous semble donc préférable de parler d'unités de longueur ou de pas.

Du point de vue informatique, ces exercices utilisent tous une variable globale pour modifier la taille de la figure dessinée par un bloc, alors qu'il serait préférable d'utiliser un bloc à paramètre (comme nous l'évoquions au paragraphe 3.4.2). Même si ce choix a peut-être été fait par souci de ne pas utiliser de concepts trop élaborés (le bloc à paramètre), il nous semble préférable de privilégier de bonnes pratiques de programmation : imaginons par exemple qu'un autre lutin accède à la variable globale en cours d'exécution et change sa valeur, le résultat obtenu ne serait certainement pas celui attendu. C'est une problématique importante en

informatique. Par ailleurs, le programme de Seconde de mathématiques insiste sur l'écriture de fonctions informatiques. Les blocs à paramètres sont ce qui s'en rapproche le plus en Scratch, et il nous semble à cet égard intéressant de familiariser les élèves avec leur utilisation dès le collège.

4.1 *Métropole – La Réunion – Antilles-Guyane, 29 juin* 2017

4.1.1 *Énoncé* (*exercice* 2, *noté* 6 *points sur* 50)

On donne le programme suivant qui permet de tracer plusieurs triangles équilatéraux de tailles différentes. Ce programme comporte une variable nommée « côté ». Les longueurs sont données en pixels.

On rappelle que l'instruction 

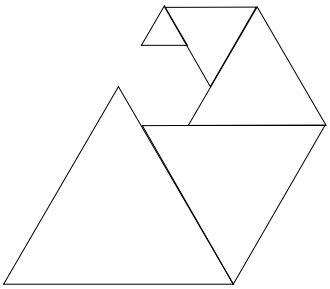
, signifie que l'on se dirige vers la droite.

Script

Le bloc triangle

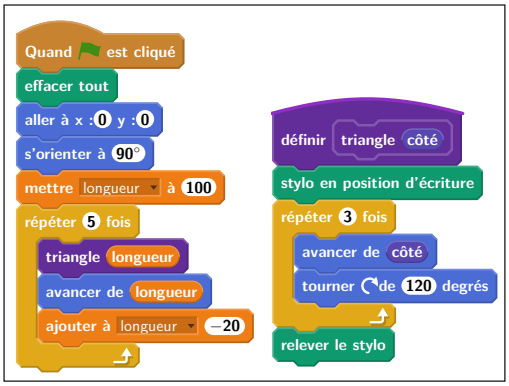
1. Quelles sont les coordonnées du point de départ du tracé ?
2. Combien de triangles sont dessinés par le script ?
3. (a) Quelle est la longueur (en pixels) du côté du deuxième triangle tracé ?
   (b) Tracer à main levée l'allure de la figure obtenue quand on exécute ce script.
4. On modifie le script initial pour obtenir la figure ci-contre.
   Indiquer le numéro d'une instruction du script après laquelle on peut placer l'instruction 

pour obtenir cette nouvelle figure.



4.1.2 *Commentaires*

Pour éviter l'écueil d'une variable globale, il est possible d'utiliser un bloc avec un paramètre comme illustré dans le programme ci-contre.



4.2 *Métropole – La Réunion – Antilles-Guyane, 14 septembre* 2017

4.2.1 *Énoncé* (*exercice* 3, 6 *points sur* 50)

Voici trois figures différentes, aucune n'est à l'échelle indiquée dans l'exercice :

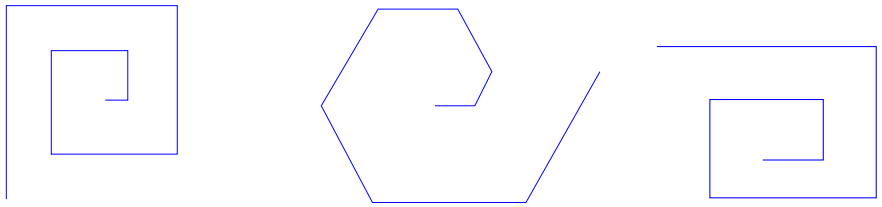
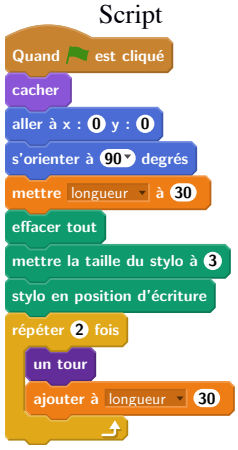


figure1 figure2 figure3

Le programme ci-dessous contient une variable nommée « longueur ».

Script



Le bloc : **un tour**

On rappelle que l'instruction 

, signifie que l'on se dirige vers la droite avec le stylo.

1. (a) Dessiner la figure obtenue avec le bloc « un tour » donné dans le cadre de droite ci-dessus, pour une longueur de départ égale à 30, étant orienté vers la droite avec le stylo, en début de tracé. On prendra 1 cm pour 30 unités de longueur, c'est-à-dire 30 pixels.

   (b) Comment est-on orienté avec le stylo après ce tracé ? (aucune justification n'est demandée)
2. Laquelle des figures 1 ou 3 le programme ci-dessus permet-il d'obtenir ? Justifier votre réponse.
3. Quelle modification faut-il apporter au bloc « un tour » pour obtenir la figure 2 ci-dessus ?

4.2.2 *Commentaires*

Cet exercice assez riche comporte en particulier des questions d'interprétation de code. On note une bonne précision générale et globalement une bonne rédaction de l'énoncé (exception faite du problème des unités de longueur précédemment mentionné). La prise en compte de l'état "caché" (position, orientation) est intéressante. L'exercice porte à la fois sur des compétences mathématiques et informatiques.

Signalons cependant deux maladresses d'expression dans la question 1(b) : on ne sait pas qui représente le « on » et l'expression « orientation du stylo » n'est pas appropriée, puisque c'est le lutin qui est orienté.

Quant au programme lui-même, il pourrait être aussi quelque peu amélioré. Nous proposons donc trois versions modifiées de ce programme (détaillées en figure 1, 2 et 3 sur la page ci-contre). Le problème de la variable globale déjà mentionné est réglé dans la version 3 via l'utilisation d'un bloc avec un paramètre. Notons également que le nom du bloc "un tour" n'est peut-être pas des plus pertinents surtout lorsqu'il s'agit de modifier le-dit bloc pour obtenir la figure 2. Le nommer « motif » présenterait peut-être moins de risque de confusion.

D'autre part, il est un peu curieux de modifier la valeur de la variable longueur une fois dans le bloc et une fois dans la boucle. Il serait plus cohérent de réaliser les deux mises à jour de la variable dans le bloc lui-même (cf Version 1). Ceci fait, le bloc pourrait alors être écrit de manière un peu plus concise (cf Version 2). On aurait aussi pu définir un motif de base ne comportant que 2 côtés au lieu de 4 et construire la figure en répétant ce motif 4 fois au lieu de 2 (cf Version 3).

Nous insistons sur le fait que les blocs de Scratch ne sont pas des fonctions, puisqu'on ne peut pas renvoyer une valeur précise, mais plutôt des procédures, ce qui en fait un système peu pratique.

Enfin, le paramètre en entrée, comme dans la version 3, n'est pas une variable au sens strict, puisque sa valeur ne peut pas être modifiée à l'intérieur du bloc, mais seulement être utilisée par les autres instructions.

Version 1

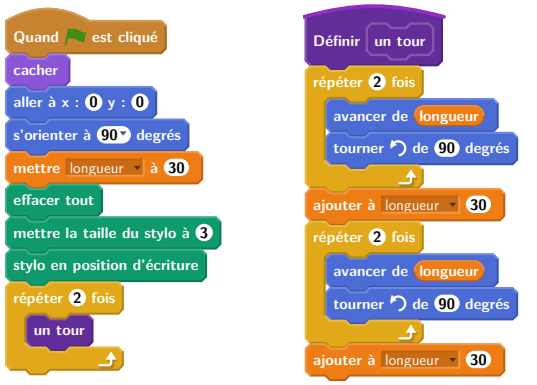


Figure 1 – Version modifiée n◦1 du programme de dessin de spirale.

Version 2

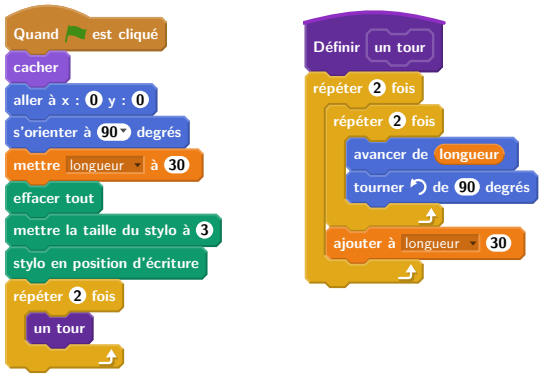


Figure 2 – Version modifiée n◦2 du programme de dessin de spirale.

Version 3

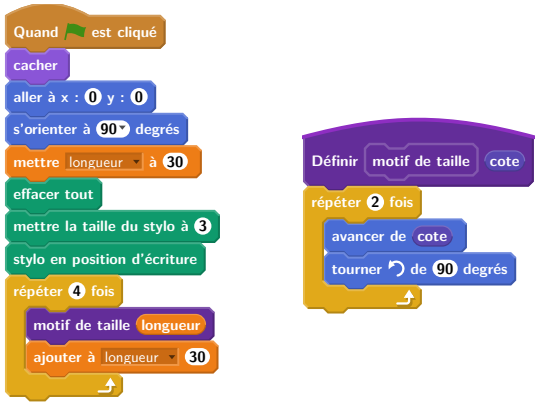


Figure 3 – Version modifiée n◦3 du programme de dessin de spirale.

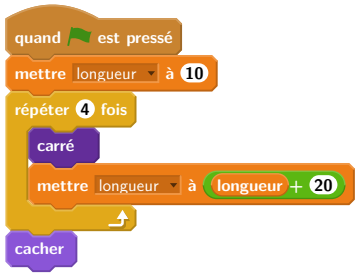
4.3 *Amérique du Sud*, 30 *novembre* 2017

4.3.1 *Énoncé* (*exercice* 6, *noté* 6,5 *points sur* 50)

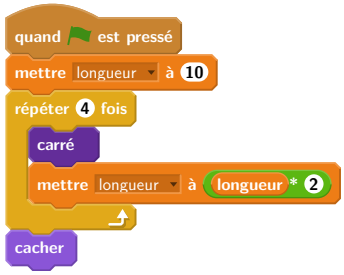
Le bloc d'instruction « carré » ci-dessous a été programmé puis utilisé dans les deux programmes ci-contre :

Rappel : L'instruction « avancer de 10 » fait avancer le lutin de 10 pixels.

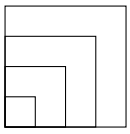
Programme n° 1



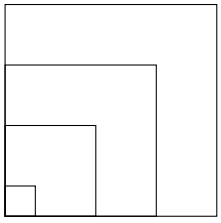
Programme n° 2



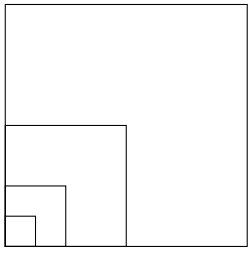
1. Voici trois dessins :



Dessin n° 1



Dessin n° 2



Dessin n° 3

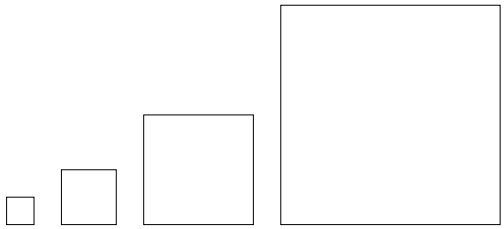
(a) Lequel de ces trois dessins obtient-on avec le programme n° 1 ?

(b) Lequel de ces trois dessins obtient-on avec le programme n° 2 ?

(c) Pour chacun des deux programmes, déterminer la longueur, en pixel, du côté du plus grand carré dessiné.

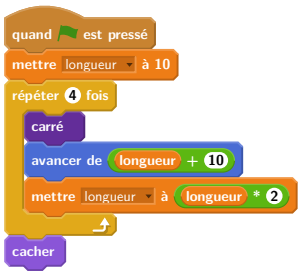
… / …

2. On souhaite modifier le programme n° 2 pour obtenir le dessin ci-dessous.

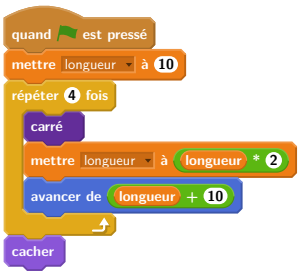


Parmi les trois modifications suivantes, laquelle permet d'obtenir le dessin souhaité ? Aucune justification n'est attendue pour cette question.

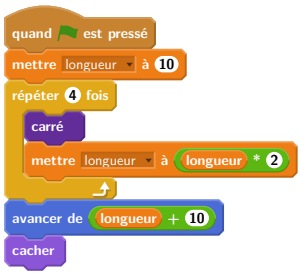
Modification 1



Modification 2



Modification 3



4.3.2 *Commentaires*

La compréhension d'algorithme est intéressante mais se limiter à ce type d'exercice ne permet pas d'évaluer toutes les compétences attendues en fin de cycle 4. Il est ainsi dommage qu'il ne soit pas demandé de réaliser un dessin, ou *a minima* de modifier un des programmes proposés pour obtenir un dessin donné. Il faudrait regarder la longueur globale du sujet de mathématiques pour évaluer la possibilité de compléter cet exercice dans ce sens.

4.4 *Propositions d'exercices*

Une fois les défauts évoqués précédemment corrigés, ces exercices sont tout à fait pertinents dans le cadre du brevet. Nous ne proposons donc pas ici de nouvel exercice mais recommandons, dans cette catégorie de problèmes, de combiner des questions de compréhension et analyse de programmes avec des exercices de conception d'algorithmes de dessin éventuellement par modification d'un programme donné.

5. — Exercices sur les aspects événementiels

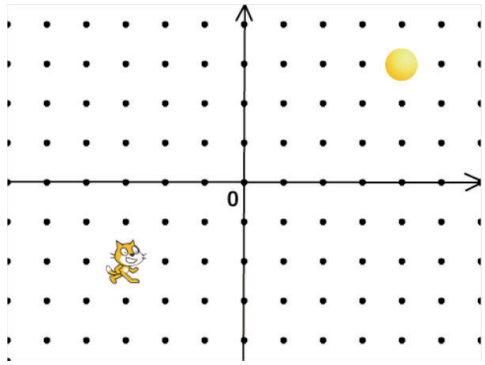
La possibilité de faire interagir plusieurs lutins, ou encore de piloter au clavier les mouvements d'un lutin, font partie des points forts qui assurent le succès de Scratch auprès des

enfants. Cet aspect événementiel du langage permet d'écrire rapidement des programmes très ludiques. Comme, de plus, il s'agit d'un domaine important de l'informatique, il serait bien entendu dommage de ne pas le mettre en avant avec les élèves. Pour autant, la bonne utilisation des messages et des événements présente quelques subtilités, comme en témoigne l'analyse du premier exercice ci-dessous. C'est pourquoi, dans le cadre d'une évaluation, nous recommandons la plus grande rigueur dans ce type d'exercices.

5.1 Amérique du Nord, 7 juin 2017

5.1.1 *Énoncé* (*exercice* 5, *noté* 4,5 *points sur* 50)

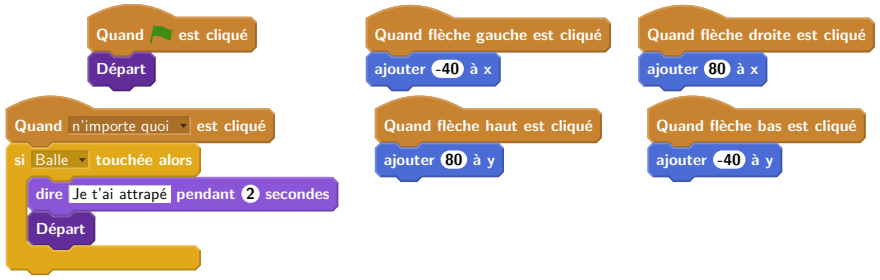
L'image ci-dessous représente la position obtenue au déclenchement du bloc départ d'un programme de jeu.



L'arrière-plan est constitué de points espacés de 40 unités. Dans cette position, le chat a pour coordonnées (– 120 ; – 80).

Le but du jeu est de positionner le chat sur la balle.

1. Quelles sont les coordonnées du centre de la balle représentée dans cette position ?
2. Dans cette question, le chat est dans la position obtenue au déclenchement du bloc départ. Voici le script du lutin « chat » qui se déplace :



a. Expliquez pourquoi le chat ne revient pas à sa position de départ si le joueur appuie sur la touche → puis sur la touche ←.

b. Le joueur appuie sur la succession de touches suivante : →→↑←↓.

… / …

Quelles sont les coordonnées x et y du chat après ce déplacement ?

c. Parmi les propositions de successions de touches ci-dessous, laquelle permet au chat d'atteindre la balle ?

Déplacement 1	Déplacement 2	Déplacement 3
→→→→→→→↑↑↑↑↑	→→→↑↑↑→↓←	↑→↑→↑→→↓↓

3. Que se passe-t-il quand le chat atteint la balle ?

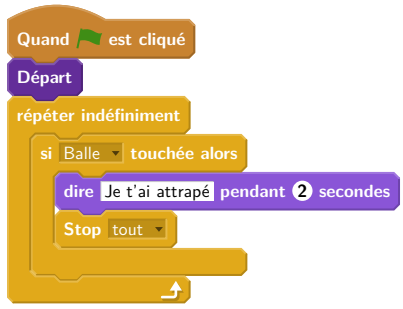
5.1.2 *Commentaires*

Il s'agit d'un exercice mettant en jeu les aspects événementiels dans le cadre d'un jeu. La graduation du repère, avec échelle donnée permet à l'élève de bien se repérer. Néanmoins, l'exercice est très ambigu, fortement basé sur la connaissance de Scratch même si, curieusement, les blocs utilisés ne correspondent pas aux blocs fournis par Scratch : par exemple, lorsqu'il s'agit d'un événement déclenché par l'appui sur une touche, le vocabulaire utilisé par Scratch est bien "pressé" et non "cliqué".

Plus ennuyeux, l'exercice semble méconnaître plusieurs points importants au sujet de la programmation en parallèle. Dans un script, il faut à tout prix éviter le déclenchement de deux scripts par le même événement car on ne sait pas quel script s'exécute en premier. Concrètement, l'expérience montre qu'en Scratch lorsque deux scripts sont concurrents, c'est celui qui a été déplacé ou modifié en dernier qui s'exécute en premier (une analyse poussée de ces problèmes et des concepts sous-jacents est proposée par Anne Heam dans [1]).

Dans l'exercice, lorsque l'on presse une touche "flèche", il est impossible de savoir si le dernier script « n'importe quoi cliqué » s'exécute avant ou après le déplacement. Il existe donc une possibilité que le lutin dépasse la balle sans que la collision soit détectée. Il y a également un problème lié à la file d'attente des instructions. Selon que le déplacement a lieu avant ou après le test, quelle suite d'instructions est-elle exécutée ? Cet exercice formulé tel quel pose donc problème dans le cadre d'un examen, mais il peut être intéressant en classe, en cours de formation.

En outre, il est tout à fait possible d'imaginer un nouveau programme ayant le comportement que l'on attend. Pour éviter d'avoir deux scripts déclenchés par le même événement, on peut, par exemple, utiliser, dans le script déclenché par le drapeau vert, une boucle infinie contenant une instruction conditionnelle qui réalise le test d'intersection :



Le programme ainsi conçu utilise un mécanisme appelé *attente active*, c'est-à-dire une boucle qui teste de manière répétée si une condition est vérifiée avant de laisser la suite

du programme s'exécuter. C'est un peu comme si quelqu'un voulant entrer dans une pièce déjà occupée, frappait de manière répétée à la porte plutôt que d'attendre d'être prévenu par la personne lorsqu'elle libère la pièce. D'autre part, même si on peut supposer que le rythme de répétition de la boucle « répéter indéfiniment » est suffisant pour garantir un bon fonctionnement, cela n'est toujours pas garanti (par exemple si un utilisateur laisse une touche appuyée).

Une solution plus simple est d'utiliser le bloc « attendre » fourni par Scratch, qui permet de suspendre l'exécution d'un script jusqu'à ce qu'un certain événement se produise :

Ce script est essentiellement équivalent à la solution précédente par attente active, la boucle infinie précédente étant ici implicitement prise en charge par l'environnement Scratch.

Une dernière solution consiste à inclure le test de collision à la fin de chacun des 4 blocs. Après tout, on n'a besoin de vérifier que le lutin touche la balle que lorsqu'il a réalisé un déplacement. Cela garantit également une uniformité de comportement, puisque le test n'est effectué à aucun autre moment. La proposition de programme de l'encadré ci-dessous suit cette approche plus cohérente, et permet de faire faire cet exercice sans changer beaucoup de questions.

5.2 *Brevet des collèges Asie 27 juin* 2017

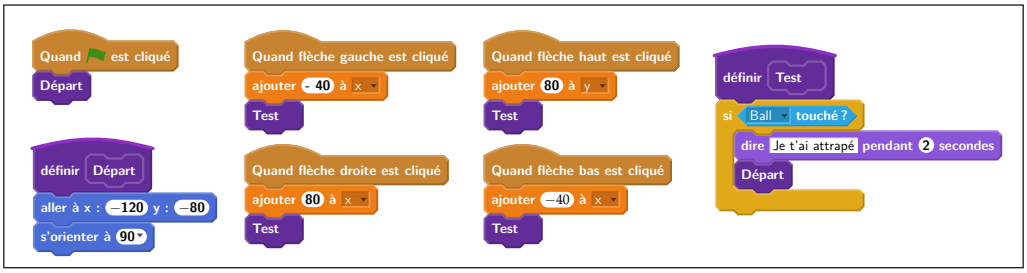
5.2.1 *Énoncé* (*exercice* 4, *noté* 4 *points sur* 50)

[voir page ci-contre]

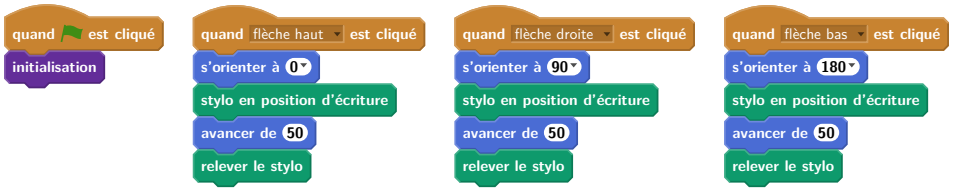
5.2.2 *Commentaires*

Les compétences évaluées par ce problème semblent être la compréhension de la programmation événementielle, la lecture d'un programme Scratch et la capacité à exécuter un programme pas à pas. L'énoncé comporte quelques erreurs et maladresses de formulation. Le bloc « initialisation » ne fait pas ce qu'il devrait faire puisque le lutin est déplacé avant de lever le stylo, un trait en trop va donc être tracé entre la position du crayon avant initialisation et la position "à gauche de l'écran". Par ailleurs la position du stylo sur l'axe des ordonnées après initialisation n'est pas spécifiée.

Dans la question 2, les modifications apportées par Julie au programme de Margot ne sont pas explicitées. Pour répondre, les élèves doivent donc d'abord les repérer. Or, le but de l'exercice n'est pas de trouver les modifications

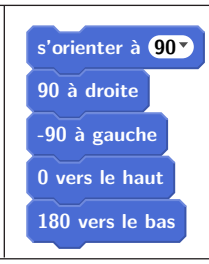


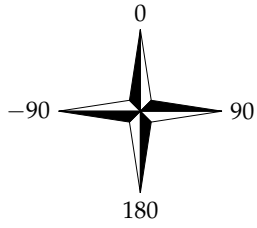
Margot a écrit le programme suivant. Il permet de dessiner avec trois touches du clavier.



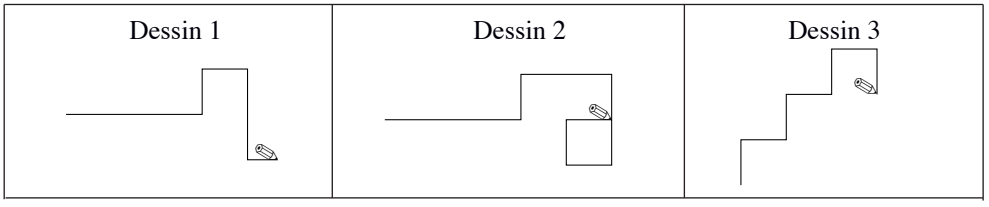
Pour information

Ce bloc efface le dessin précédent, positionne le crayon à gauche de l'écran et relève le stylo.



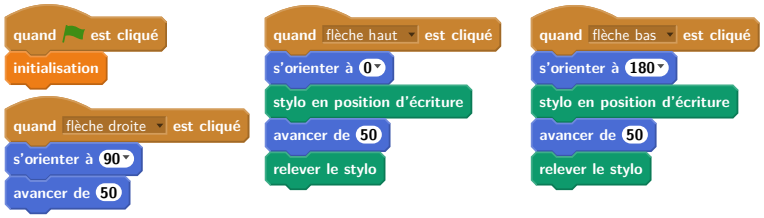


1. Parmi les trois dessins suivants, un seul ne pourra pas être réalisé avec ce programme. Lequel ? Expliquer.



2. Julie a modifié le programme de Margot (voir ci-dessous). Que devient alors le dessin 3 avec le programme modifié par Julie ?

Programme modifié par Julie



mais de comprendre ce qu'elles font. Aussi, serait-il préférable d'indiquer explicitement dans le texte où ont eu lieu les changements, afin de ne pas ajouter, pour certains élèves, une difficulté supplémentaire sans rapport avec les objectifs du sujet.

Cette question contient également quelques problèmes de formulation. En annonçant dans la question que le dessin 3 change, on donne un indice sur la question précédente, en excluant de fait ce dessin. De plus, le dessin 3 ne « change » pas, puisqu'il est le produit de la séquence d'instructions précédente. Julie crée une nouvelle séquence d'instructions qui donne une nouvelle figure.

Enfin, il pourrait être intéressant d'ajouter une question intermédiaire entre les questions 1 et 2, formulée par exemple ainsi : « Pour les deux autres dessins, quelle est la séquence de touches pressées par l'utilisateur ? ». La question 2 pourrait alors devenir : « Julie a modifié le programme de Margot (voir ci-dessous). Que dessine le lutin avec le programme modifié par Julie pour les séquences de touches trouvées à la question précédente ? ».

5.3 *Propositions d'exercices*

5.3.1 *Jeu Tout blanc tout noir*

On peut proposer, pour travailler sur les événements, un travail autour du jeu « Tout blanc, tout noir ». Ce jeu se définit comme suit :

— un tableau carré contient des cases, 9 au départ par exemple ;

— chaque case peut être soit blanche, soit noire ;

— lorsque l'on clique sur une case, la case cliquée et les cases adjacentes horizontalement et verticalement changent de couleur;

— le but du jeu est, à partir d'une situation initiale, par exemple un tableau tout blanc, d'arriver à un tableau entièrement noir.

Dans Scratch, on peut créer des lutins, chacun représentant une case, avec deux costumes (l'un noir, l'autre blanc). On duplique ce lutin autant de fois que nécessaire. Il reste à programmer chaque lutin pour qu'il occupe la place d'une des cases du tableau. Pour la suite, on considérera que les cases sont numérotées ainsi :

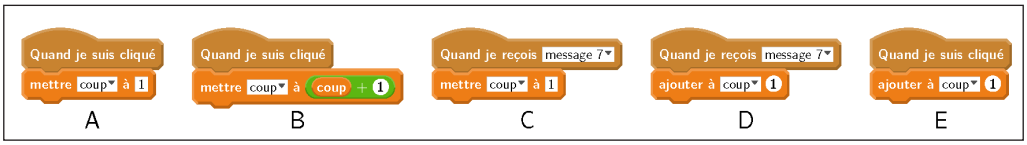
1	2	3
4	5	6
7	8	9

Pour programmer un tel jeu avec Scratch, on utilisera le système de passage de messages entre lutins. Lorsqu'on clique sur une case, un message est envoyé par le lutin correspondant. On considérera ici que le fait de cliquer sur la case *n* (et donc sur le lutin qui s'y trouve) envoie le message *n*.

En Scratch, un message envoyé par un lutin peut être reçu par tous les autres, mais seuls les lutins des cases voisines à la case n seront programmés pour réagir au message n en changeant de couleur (c'est-à-dire en passant du blanc au noir et inversement). Par exemple, si depuis l'état initial du jeu (toutes cases blanches) on clique sur le lutin (et donc la case) 2, les lutins des cases 1, 3 et 5 réagissent au message et changent de couleur. On obtient donc la figure suivante :

1	2	3
4	5	6
7	8	9

Si on clique ensuite sur le lutin (donc la case) 6, le message 6 est envoyé, les lutins des cases

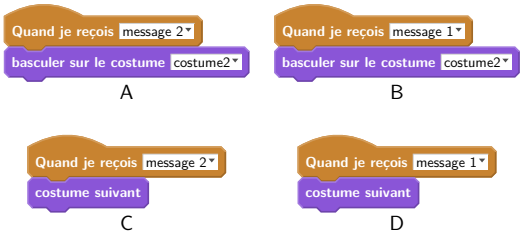


3, 5 et 9 réagissent au message et on obtient la figure suivante :

1	2	3
4	5	6
7	8	9

Voici un exemple de questions que l'on pourrait poser aux élèves au sujet de ce jeu. On note que tout cet exercice suppose une certaine connaissance de la sémantique propre du langage Scratch.

1. Quelles sont les cases que l'on peut cliquer pour faire changer la case 7 de couleur ?
2. On clique sur la case 1. Choisir le programme approprié pour le lutin de la case 2.



3. Combien de scripts de changement de costume le lutin 5 possède-t-il (ne pas oublier qu'une case doit changer de couleur quand on clique dessus) ?
4. Donner les numéros des messages qui font changer la case 5 de couleur.
5. On souhaite ajouter un compteur de coups, pour savoir qui réussit le jeu le plus rapidement. On crée la variable pour tous les lutins. Quel(s) programme(s) associé(s) au lutin 7 parmi les programmes ci-dessus permet(tent) de rajouter 1 au compteur de coups quand la case 7 est cliquée ?

Réponses aux questions.

1. On peut cliquer sur les cases 4, 7 et 8 pour changer la couleur de la case 7.
2. C'est le programme D qui convient. En effet, la commande « basculer sur le costume » affecte un costume spécifique au lutin, et il ne pourra pas en changer au prochain clic.
3. Le lutin 5 est associé à 5 changements de costume. Il ne faut pas oublier qu'il change de couleur lorsque l'on clique sur lui-même.
4. Les messages qui font changer le lutin 5 sont les messages 2, 4, 6 et 8. Le lutin ne reçoit pas de message externe pour changer de couleur lui-même.
5. Les séquences B et E permettent de rajouter une unité au compteur de coups. La première séquence initialise la variable à 1 systématiquement, les séquences C et D ne concernent pas le bon événement, puisque c'est le lutin 7 lui-même qui est cliqué.

6. — Simulation en probabilités

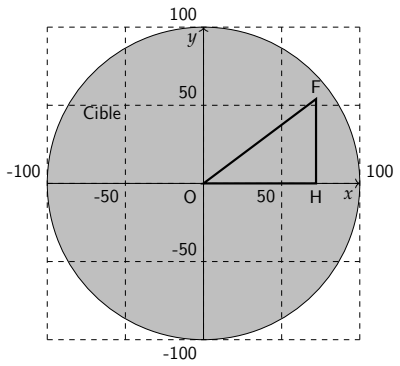
6.1 *Brevet des collèges Pondichéry 2 mai* 2018

6.1.1 *Énoncé* (*exercice* 5, 20 *points sur* 100)

ANALYSE DES EXERCICES D'ALGORITHMIQUE
ET DE PROGRAMMATION DU BREVET 2017

Dans tout l'exercice l'unité de longueur est le mm.

On lance une fléchette sur une plaque carrée sur laquelle figure une cible circulaire (en gris sur la figure), Si la pointe de la fléchette est sur le bord de la cible, on considère que la cible n'est pas atteinte.



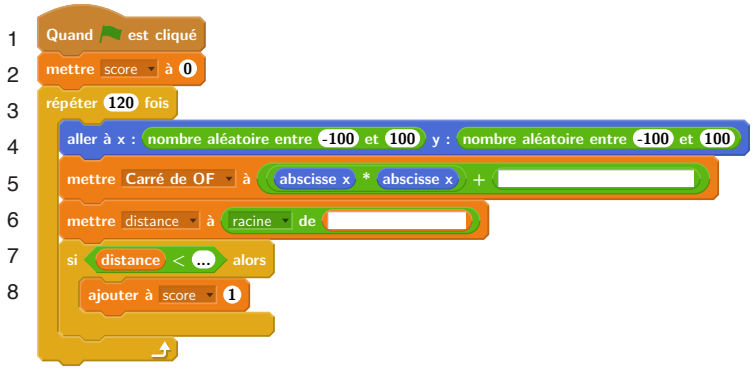
On considère que cette expérience est aléatoire et l'on s'intéresse à la probabilité que la fléchette atteigne la cible.

— La longueur du côté de la plaque carrée est 200.

— Le rayon de la cible est 100.

— La fléchette est représentée par le point F de coordonnées $(x\,;\,y)$ où x et y sont des nombres aléatoires compris entre – 100 et 100.

1. Dans l'exemple ci-dessus, la fléchette F est située au point de coordonnées (72 ; 54). Montrer que la distance OF, entre la fléchette et l'origine du repère est 90.
2. D'une façon générale, quel nombre ne doit pas dépasser la distance OF pour que la fléchette atteigne la cible ?
3. On réalise un programme qui simule plusieurs fois le lancer de cette fléchette sur la plaque carrée et qui compte le nombre de lancers atteignant la cible. Le programmeur a créé trois variables nommées : **carré de OF**, **distance** et **score**.



(a) Lorsqu'on exécute ce programme, combien de lancers sont simulés ?

(b) Quel est le rôle de la variable **score** ?

(c) Compléter et recopier sur la copie uniquement les lignes 5, 6 et 7 du programme afin qu'il fonctionne correctement.

(d) Après une exécution du programme, la variable **score** est égale à 102. À quelle fréquence la cible a-t-elle été atteinte dans cette simulation ? Exprimer le résultat sous la forme d'une fraction irréductible.

4. On admet que la probabilité d'atteindre la cible est égale au quotient : aire de la cible divisée par aire de la plaque carrée. Donner une valeur approchée de cette probabilité au centième près.

6.1.2 *Commentaires*

Cet exercice propose à l'élève de réfléchir à une implémentation en Scratch d'une estimation de surface par une méthode probabiliste de type Monte-Carlo. L'exercice met surtout l'accent sur l'aspect géométrique du problème (calcul de la distance du point F à l'origine) ainsi que sur le calcul de la fréquence à laquelle la cible est touchée. L'objectif de cette expérience n'est pas explicité dans l'énoncé. Le lien entre la probabilité de toucher la cible et la surface du disque gris est évoqué dans la question 4, mais n'est pas ou peu exploité. L'élève n'est pas invité à comparer la fréquence empirique obtenue sur l'exemple et la fréquence théorique. De plus, nulle part il n'est mentionné que l'un des intérêts de cette expérience est d'estimer le nombre π en cherchant une approximation de l'aire du disque.

Il est intéressant de noter que le bloc « nombre aléatoire entre – 100 et 100 » induit une discrétisation plutôt grossière du problème. En effet, les nombres renvoyés par cette expression sont des entiers. Cette caractéristique (ainsi que la non prise en compte des points appartenant au cercle lui-même) empêchent l'obtention d'une estimation très précise. Pour remédier à ce problème, on pourra par exemple tirer des nombres entiers plus grands et les diviser par une constante bien choisie pour se rapporter à l'intervalle de valeurs souhaité.

Plutôt que de tirer aléatoirement des points à coordonnées entières du plan, on pourrait d'ailleurs obtenir plus simplement une approximation de l'aire du disque en énumérant de manière exhaustive tous les points à coordonnées entières du carré blanc, et dénombrer ceux qui parmi eux appartiennent au disque. Dans le cas particulier où le tirage aléatoire est aussi grossier, une telle recherche exhaustive semble tout à fait justifiée.

En ce qui concerne les attendus du cycle 4 en algorithmique et programmation, une part importante de l'activité de programmation de l'expérience est prise en charge par l'énoncé.

L'exercice évalue notamment la capacité de l'élève à comprendre une boucle « répéter *n* fois ». Cependant la mise en œuvre des expressions arithmétiques et des instructions d'affectation demeurent à un niveau modeste. Plus ennuyeux, l'élève doit également disposer de connaissances spécifiques au logiciel Scratch : existence des variables spéciales « abscisse *x* » et « ordonnée *y* », instruction « aller à », expression « nombre aléatoire ». Comme dans la plupart des exercices de brevet étudiés ici, une connaissance spécifique de ce langage est donc évaluée.

Enfin, en ce qui concerne la rédaction du programme Scratch proposé, on peut s'interroger sur le choix du nom de la variable « carré de OF », qui fait référence à la figure, contrairement à la variable « distance » qui porte un nom générique.

6.2 *Propositions d'exercices*

6.2.1 *Estimation de π par la méthode de Monte-Carlo*

Cette proposition reprend le thème de l'exercice de brevet commenté ci-dessus, en mettant davantage l'accent sur la démarche scientifique qui motive cette expérience (approche fréquentiste des probabilités). En conservant un programme Scratch similaire, il nous semblerait intéressant de proposer les ajustements suivants. On ne se situe plus ici dans un exercice de type examen, mais plutôt dans une activité complète à réaliser en classe.

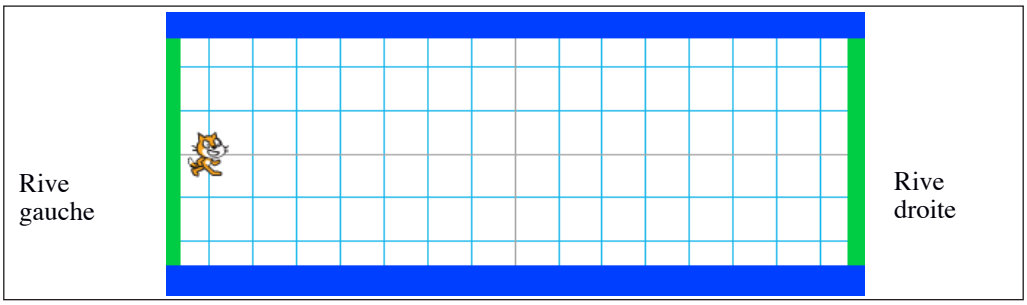
1. Activité préliminaire permettant de décrire l'expérience et son objectif, travail sur le lien entre le nombre π, l'aire du disque et la fréquence à laquelle la cible est touchée ;
2. Programmation sous Scratch du programme de lancer de fléchettes, éventuellement en remplaçant le nombre de tirages (borne de la boucle) par une variable ou un paramètre de bloc ;
3. Ajout des instructions permettant de calculer la fréquence et d'en déduire une estimation de π ;
4. Débat scientifique sur la qualité de l'approximation obtenue, corrélation avec le nombre de fléchettes lancées ;
5. Ouverture éventuelle sur d'autres démarches semblables (évaluation d'autres types de surfaces par exemple) ou sur d'autres méthodes d'approximation de π (méthode d'Archimède ?).

6.2.2 *Le pont des deux rives*

Le pont des deux rives est une situation de marche aléatoire pour laquelle la modélisation est compliquée dans le second degré, et où la simulation se justifie parfaitement.

Scratchy doit traverser un pont suspendu, de 450 unités Scratch de long et de 150 unités Scratch de large. Il commence sa traversée au centre du pont, sur la rive de gauche, il se trouve à la position (– 210 ; 0) sur la scène. À chaque étape, il fait un saut vers l'autre rive et son abscisse sur la scène augmente de 30 unités Scratch. Mais comme il y a du vent, son ordonnée peut augmenter de 30, rester la même ou diminuer de 30 (en unités Scratch) de manière équiprobable. Atteindre l'autre rive signifie que Scratchy ne doit pas tomber dans l'eau et doit atteindre une abscisse de 240 sur la scène. On veut estimer la probabilité qu'a Scratchy de sortir du pont sans tomber dans l'eau.

Partie 1. La scène de départ est donnée ci-dessous : Le pont est représenté par la zone quadrillée blanche, les rives par les rectangles verticaux verts. Les rectangles horizontaux bleus représentent de l'eau.



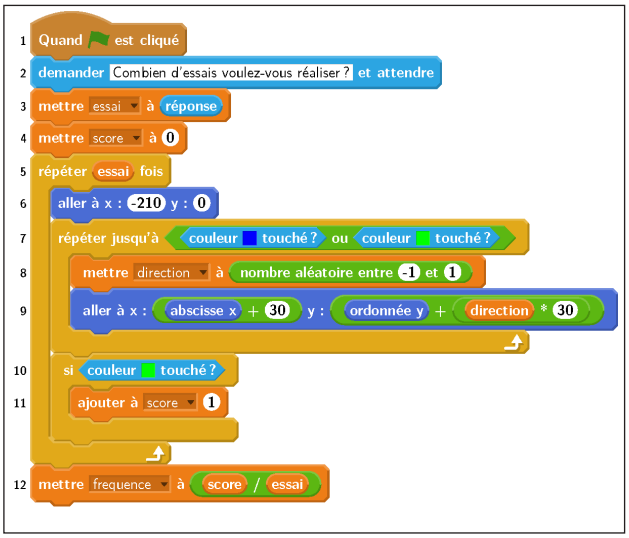
1. Quelle est la longueur d'une des cases en unités Scratch ?
2. Donner les trois couples de coordonnées possibles pour Scratchy après son premier déplacement.
3. Si Scratchy parvient à atteindre la rive de droite, combien de sauts a-t-il fait ?

Partie 2. Voici (cadre ci-contre) une proposition de programme qui simule cette situation. Il comporte trois variables :

— La variable , choisie par l'utilisateur, définit le nombre de tentatives de traversée à effectuer ;

— La variable enregistre le nombre de traversées réussies ;

— La variable indique la prochaine direction dans laquelle Scratchy se déplacera dans la scène : 1 si Scratchy doit monter, 0 s'il doit avancer horizontalement, – 1 s'il doit descendre.



1. Indiquer le numéro de ligne de ce programme qui permet le choix aléatoire de la direction.
2. Quelles valeurs peuvent-elles être obtenues par ce tirage aléatoire ?
3. Ligne 9, quelles sont les valeurs possibles de l'expression ?
4. Expliquer le choix des valeurs que peut prendre la variable direction.
5. Quelle est la signification de la condition ligne 10? Quelle situation permet-elle de représenter dans le problème de départ ?
6. Que permet d'estimer la fréquence calculée à la ligne 12 ?
7. On a réalisé plusieurs fois l'expérience avec Scratch et on a regroupé les résultats obtenus dans la feuille de tableur ci-dessous. Parmi les formules suivantes, laquelle doit-on saisir en B3 et recopier jusqu'en H3 pour calculer la fréquence de réussite de la traversée du pont ?

= B1*B2 = B1/B2 = B2/B1 = B1 + B2

8. D'après ce tableau, quelle est la fréquence de réussite si on réalise 1000 essais ?

6.2.3 *Commentaires*

	A	B	C	D	E	F	G	H
1	Nombre d'essais réalisés	50	100	200	300	400	500	1000
2	Nombre de sorties du pont réussies	23	39	65	93	136	154	333
3	Fréquence de réussite de la traversée du pont							

Dans cet exercice, nous avons choisi de mêler des connaissances informatiques et mathématiques. En particulier, au-delà des connaissances mathématiques liées à une approche fréquentiste des probabilités dans une situation où la modélisation du problème est inaccessible pour des élèves du secondaire, ce sont aussi les choix de représentation informatique de la situation qui sont questionnés. Nous avons choisi de ne pas faire intervenir de compteur, ni d'utiliser la variable spéciale « abscisse x » pour identifier si Scratchy traverse le pont. Nous avons préféré représenter la réussite de la traversée par l'événement « couleur verte touchée », qui joue alors le même rôle de booléen que le test « abscisse x > 210 » par exemple. Nous exploitons ainsi les potentialités de programmation événementielle de l'environnement. Ainsi un élève qui n'aurait pas réussi à traiter la première partie pourrait quand même s'engager dans la seconde.

De la même manière, dans la question 4 de la partie 2, nous questionnons le choix de représentation des directions possibles et demandons de l'expliquer. Le travail mathématique de justification convoque ici la notion d'état qui se traduira potentiellement par une disjonction de cas.

Cette proposition nous semble répondre à un certain équilibre entre connaissances informatiques et mathématiques. De plus, du point de vue informatique, elle dépasse la simple lecture d'instructions écrites et va jusqu'à questionner les choix de représentation qui sont faits dans le programme proposé.

7. — Conclusion

Dans cet article, nous nous sommes astreints à porter un regard purement informatique sur les exercices étudiés, sans nous préoccuper de leur éventuel intérêt mathématique. En particulier, nous avons relevé divers éléments des énoncés qui posent problème d'un point de vue informatique, ou qui peuvent être source de confusion avec des notions mathématiques.

Notons que, dans l'ensemble des douze sujets analysés, on ne relève que trois instructions de branchement conditionnel (si ... alors ... sinon ...) , et aucune boucle « non bornée ». Par contre, on relève sept boucles « bornées ». On peut donc dire que les notions fondamentales de l'algorithmique et de la programmation sont inégalement exploitées dans ces exercices. En ce qui concerne les notions plus évoluées, les procédures (ou « blocs » de Scratch) apparaissent dans six sujets sur douze, mais on peut regretter qu'il s'agisse à chaque fois de procédures sans paramètre, même lorsqu'il aurait été naturel, à au moins deux occasions, d'utiliser des blocs avec un paramètre dans le cadre de l'exercice posé. Enfin, la formulation de l'un des deux exercices faisant appel à l'aspect événementiel du langage est particulièrement maladroite du point de vue informatique.

Nous nous réjouissons de voir que des exercices d'algorithmique ont été proposés aux épreuves du DNB en 2017 et en 2018, et nous espérons que cette présence de l'informatique dans les épreuves terminales de la scolarité obligatoire est désormais pérenne. Faute de temps, nous n'avons pas analysé en détail les exercices posés au brevet en 2018, mais nous nous réjouissons de constater d'une part qu'ils étaient de nouveau bien présents, et d'autre part qu'ils sont dans l'ensemble plus intéressants sur le plan informatique que ceux de l'année précédente. Nous avons également consulté les exercices posés en 2017 aux Concours de Recrutement des Professeurs des Écoles. Nous espérons que ceux qui seront posés dans les éditions suivantes seront plus étoffés.
Dans les sujets étudiés, les mots « script » et «

ANNEXE

Présentation de Scratch 2

programme » sont utilisés indifféremment. Dans ce contexte, nous avons utilisé la même terminologie. Dans cette section, on trouvera un glossaire rapide de certains termes utilisés dans les sujets du DNB.

A.1 Fenêtres

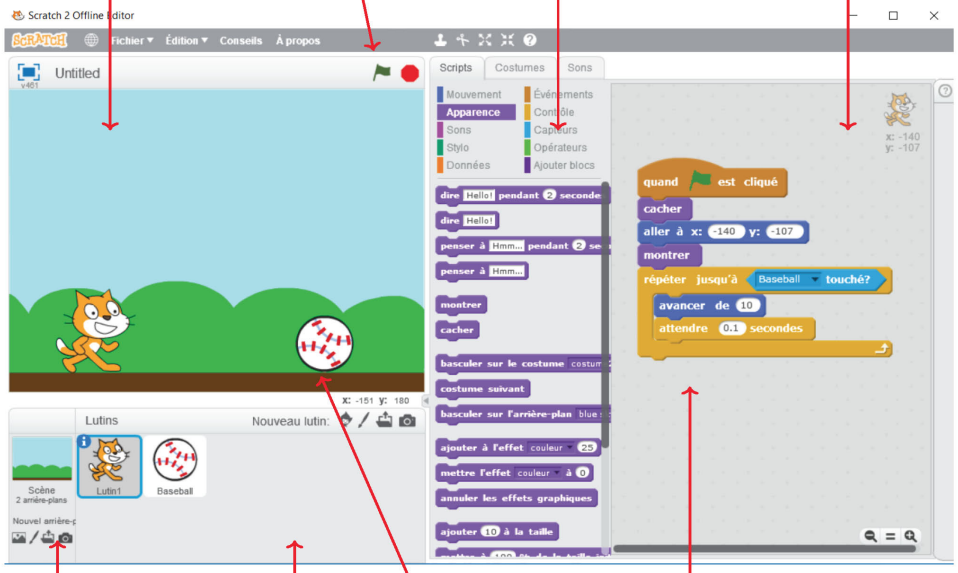
L'environnement de travail de Scratch présente plusieurs fenêtres dont chacune a sa spécificité.

Lorsqu'un programme est exécuté, ses effets sont visibles dans la **fenêtre d'exécution** ou **scène**.

Drapeau vert

Les blocs de commande sont accessibles via différents menus colorés depuis la **fenêtre des blocs**.

On glisse les blocs depuis la fenêtre des blocs dans la **fenêtre de programmation**.



Dans la **fenêtre des arrières-plans**, on peut sélectionner l'arrière-plan dont on modifie les propriétés (script, etc).

Dans la **fenêtre des lutins** on peut sélectionner le lutin dont on modifie les propriétés (script, etc).

Lutin : Personnage à qui on fait exécuter les instructions indiquées par **le script** qui lui est associé.

A.2 Programmation événementielle

L'exécution d'instructions est déclenchée par un évènement, symbolisé par un bloc de forme différente des autres, et placé en tête de script. Par exemple, cliquer sur le drapeau vert déclenche l'exécution du script associé au bloc :

La notion d'évènement n'est pas uniquement liée aux actions de l'utilisateur, les lutins peuvent aussi envoyer des messages pour déclencher l'exécution d'instructions chez d'autres lutins. On trouve par exemple les blocs suivants :

Des capteurs sont aussi disponibles afin de gérer des interactions entre lutins, ou avec des éléments de la scène, comme par exemple :

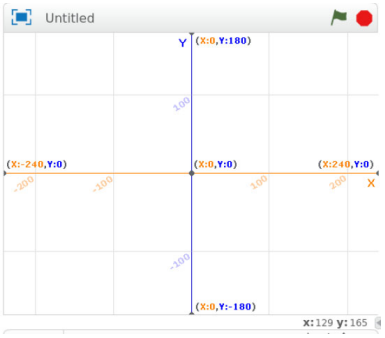
A.3 Repère

La fenêtre d'exécution est un plan repéré, dont l'origine est au centre de la fenêtre. Les plages de coordonnées ne peuvent pas être changées :

— les abscisses vont de –240 à 240 ;

— les ordonnées vont de –180 à 180.

Un point particulier associé à chaque lutin permet de le repérer dans ce système de coordonnées. Pour chaque lutin, deux variables réservées et permettent de stocker ses coordonnées.



Il existe un capteur associé à chacune de ces variables, qui permet de récupérer la position d'un lutin, par exemple . De nombreux blocs permettent de modifier ces coordonnées d'un point de vue absolu, ou d'un point de vue relatif :

A.4 Stylo

Chaque lutin est muni d'un stylo qui lui permet d'écrire sur la scène. La position de la pointe

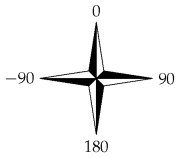
du stylo correspond aux coordonnées du lutin. Plusieurs paramètres sont associées au stylo (position d'écriture, couleur, taille, intensité). Par exemple :

A.5 Orientation

Le plan est orienté comme ci-contre. L'instruction permet de mettre le lutin dans une position

fixée par rapport à ce repère. Pour chaque lutin, une variable réservée permet de stocker l'orientation : .

Il existe trois instructions relatives aux angles qui permettent de modifier l'orientation en se plaçant soit d'un point de vue absolue, soit d'un point de vue relatif.

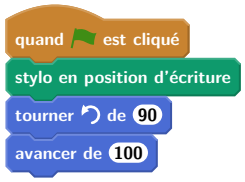


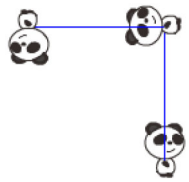
A.6 État du système

Au début de l'exécution d'un programme, certaines variables réservées de Scratch ne sont pas

réinitialisées, comme par exemple la position du stylo, la position du lutin, son orientation, etc. Par exemple, deux exécutions successives du script ci-dessous font tracer au lutin un angle droit.

Étant donné que le stylo est resté en position écriture, deux segments supplémentaires sont tracés.





Références

[1] Anne Héam. « La programmation événementielle avec Scratch : moins simple qu'il n'y paraît ». In : MathémaTICE (56 14 juin 2017). url :

http://revue.sesamath.net/spip.php?article998.

[2] Denis Vergès. APMEP : Brevet – sujets tous corrigés depuis 2008. APMEP. url :

https://www. apmep.fr/-Brevet-273-sujets-tous-corriges-.