
LE LOGICIEL SCRATCH AU COLLEGE :

Un mariage de raison entre mathématiques et informatique

Michel CHEVALLIER
Christelle PAISNEL
Jean-Luc DE SEEGER

Irem de Rouen.

*Ce texte est également consultable en ligne sur
le portail des IREM (onglet : Repères IREM) :
<http://www.univ-irem.fr/>)*

Les programmes de mathématiques mis en place à la rentrée scolaire 2016 introduisent une nouvelle partie nommée « Algorithmique et programmation » : « *En outre, un enseignement de l'informatique est dispensé conjointement en mathématiques et en technologie.* »¹. Bien que ces mêmes programmes précisent : « *Au cycle 4, les élèves s'initient à la programmation, en développant dans une démarche de projet quelques programmes simples, sans viser une*

connaissance experte et exhaustive d'un langage ou d'un logiciel particulier. », le logiciel Scratch² est fortement préconisé, pour ne pas dire imposé, comme l'attestent les documents ressources³ parus sur ce thème, l'exercice 4 du sujet⁴ 0 de l'épreuve de mathématiques, physique-chimie, sciences de la vie et de la terre et technologie du DNB 2017 ou encore les différentes formations académiques disciplinaires dispensées en 2016. C'est pourquoi nous avons basé nos travaux de

1 Bulletin Officiel de l'Éducation Nationale spécial n° 11 du 26 novembre 2015.

2 Scratch est un projet du groupe Lifelong Kindergarten au MIT Media Lab. On utilise pour cet article la version 2 du logiciel.

3 http://cache.media.eduscol.education.fr/file/Algorithmique_et_programmation/67/9/RA16_C4_MATH_algo-

[rithmique_et_programmation_N.D_551679.pdf](#) consulté le 15 septembre 2016.

4 http://cache.media.eduscol.education.fr/file/DNB/81/4/DNB_2017_Sujet_zero_MathsSciences2_MPCT_563814.pdf consulté le 15 septembre 2016. Les différents sujets de mathématiques proposés en mai et juin au DNB 2017 ont confirmé la place de Scratch pour l'enseignement de cette partie du programme.

recherches sur l'utilisation de ce logiciel en classe sans, pour autant, s'interdire des incursions vers d'autres langages comme Algobox qui nous semble constituer une transition intéressante entre Scratch et un langage davantage textuel⁵.

De plus, les instructions officielles proposent deux voies pour travailler cette partie du programme : l'une porte sur la réalisation de jeux, l'autre relève d'activités intra-mathématiques⁶. Ces mêmes documents, ainsi que les documents ressources publiés sur ce sujet, proposent principalement des scénarios d'usages de Scratch pour l'enseignement basés sur les jeux. La documentation explorant cette voie est dense, tant sur les sites institutionnels⁷ que dans certains livres destinés au grand public⁸. A contrario, nous ne trouvons que très peu de propositions d'apprentissage de la programmation en intra-mathématiques.

Pourtant, pour le lycée, les programmes de mathématiques en algorithmique insistent sur ce point : « *Il serait souhaitable d'intégrer l'écriture d'algorithmes dans tous les domaines du*

programme ... »⁹ ou encore « *L'algorithmique a une place naturelle dans tous les champs des mathématiques et les problèmes posés doivent être en relation avec les autres parties du programme (analyse, géométrie, statistiques et probabilités, logique) ...* »¹⁰, recommandations confirmées par les aménagements du programme de mathématiques pour la classe de seconde : « *L'algorithmique a une place naturelle dans tous les champs des mathématiques et les problèmes ainsi traités doivent être en relation avec les autres parties du programme (fonctions, géométrie, statistiques et probabilité, logique) mais aussi avec les autres disciplines ou la vie courante.* »¹¹. C'est pour toutes ces raisons que nous avons porté plus particulièrement notre réflexion sur la construction et l'analyse d'activités de programmation en lien avec les autres domaines des mathématiques. Vous trouverez à ce propos, sur le site de l'Irem de Rouen, dans l'onglet groupes puis images mentales et TICE, un lien¹² vous permettant d'accéder à un ensemble d'activités autour de la programmation pour les quatre années du collège ainsi qu'une proposition de progression pour travailler l'ensemble de ce thème.

5 L'aménagement du programme de mathématiques pour la classe de seconde précise, pour la partie « Algorithmique et programmation » : « Dans le cadre de cette activité, les élèves sont entraînés à décrire des algorithmes en langage naturel ou dans un langage de programmation, à en réaliser quelques-uns à l'aide d'un programme simple écrit dans un langage de programmation textuel, à interpréter des algorithmes plus complexes ».

6 « [L]es domaines du socle et ces thèmes du programme ne sont évidemment pas étanches. [...] Pour ce faire, une place importante doit être accordée à la résolution de problèmes, qu'ils soient internes aux mathématiques, ou liés à des situations issues de la vie quotidienne ou d'autres disciplines. » B.O.E.N spécial n° 11 du 26 novembre 2015.

7 Au travers, par exemple, des TRAAMs (Travaux Académiques mutualisés) disponibles sur le site <http://eduscol.education.fr/maths/animation/actions-specifi/traam2015> 2016 consulté le 15 septembre 2016.

8 Nous pouvons citer deux livres disponibles aux éditions Eyrolles : Scratch pour les kids et Cahiers Scratch pour les kids. Ces livres peuvent vous permettre de développer votre instrumentalisation sur Scratch.

9 Documents ressources en algorithmique pour le lycée http://cache.media.eduscol.education.fr/file/Programmes/17/8/Doc_ress_algo_v25_109178.pdf consulté le 15 septembre 2016.

10 Programmes de 1ère S : http://cache.media.education.gouv.fr/file/special_9/21/1/mathsS_155211.pdf,

B.O.E.N. spécial du 30 septembre 2010.

11 Aménagements de l'enseignement des mathématiques en seconde parus au BO n° 18 du 4 mai 2017

http://cache.media.education.gouv.fr/file/18/95/3/ensel512_maths_757953.pdf consulté le 22 juin 2017.

12 Ou directement à <http://irem.univ-rouen.fr/tuic/algo-college>.

Dans cet article, nous allons nous intéresser plus particulièrement à l'influence de l'apprentissage de la programmation sur l'introduction de la lettre au collège et sur l'utilisation du « si ... alors ... » dans les démonstrations de géométrie. Les programmes proposés dans cet article ne sont là que pour servir de prétexte afin d'analyser le fonctionnement de certaines instructions du logiciel et le comportement des « éléments variables ». Nous proposons en annexe un éclairage concernant certains éléments informatiques à propos des variables et des attributs des objets qui pourront être lus ultérieurement. Les instructions officielles nous interpellent en effet sur l'impact que peut avoir cet enseignement sur l'apprentissage de certaines autres notions en mathématiques.

Tout d'abord, le projet de programme pour le cycle 4 du 9 avril 2015, mis à jour le 15 avril 2015, précisait : « *Enfin, l'introduction de l'algorithmique et de la programmation renouvelle l'enseignement du raisonnement, éclaire l'introduction du calcul algébrique et fournit un nouveau langage pour penser et communiquer.* ». Ensuite, le programme définitif, modulait ce propos en indiquant : « *En créant un programme, ils développent des méthodes de programmation, revisitent les notions de variables et de fonctions sous une forme différente, et s'entraînent au raisonnement.* ». En quoi donc l'introduction de la programmation va-t-elle, comme l'indiquent les instructions officielles, *renouveler, éclairer, revisiter*, l'apprentissage de la lettre et du raisonnement ?

Variable informatique versus variable mathématique.

Au collège, l'introduction de la lettre se fait très progressivement durant le cycle 4 à travers ses différents statuts (lettre évaluée¹³, indéterminée, inconnue, variable).

Quelle place peut prendre la variable informatique dans cet ensemble ? Peut-elle être première dès le cycle 3 ? Permet-elle d'*éclairer* l'usage de la lettre en mathématiques, de l'introduire ou bien ne risque-t-elle pas de faire obstacle à l'appropriation de la lettre en mathématiques durant le cycle 4 ? En quoi ces deux types de variables se différencient-ils ou bien s'apparentent-ils ?

La construction d'un carré

La construction d'un carré en sixième avec le logiciel Scratch repose sur un déplacement et un changement de direction que l'on va répéter quatre fois en utilisant le stylo.

Le déplacement peut être commandé par trois instructions :

- « avancer de ... »,
- « aller à x : ... y : ... »
- « glisser en ... secondes à x : ... y : ... ».

Les deux premières provoquent un saut visuel avec l'apparition quasi-instantanée du trait alors que la troisième instruction construit le trait au fur à mesure du déplacement du lutin.

Le changement de direction est piloté par deux instructions :

- « s'orienter à : ... »
- « tourner ... de ... degrés ».

La première repose sur une orientation absolue du lutin dans la scène alors que la seconde est relative et modifie l'orientation initiale du lutin.

La perception des effets de ces deux instructions diffère également : « tourner ... de

¹³ « Du numérique au littéral », ressources d'accompagnement des anciens programmes, Eduscol, mars 2016.

... degrés » évoque davantage le dynamisme parce qu'il s'appuie sur le déplacement du lutin alors que « s'orienter à : ... » évoque plutôt une modification d'une caractéristique intrinsèque au lutin.

Le choix du lutin pour une telle construction n'est pas neutre. Par exemple, un crayon (Pencil), dont on aurait bloqué la rotation du costume, pourrait être utilisé : on se rapprocherait du tracé papier/crayon. Notre choix s'est plutôt porté sur le scarabée vu de dessus (Beetle) qui nous semble plus adapté à un tracé en deux dimensions qu'un lutin comme le chat qui est plutôt en vue de face ou de côté. Ce lutin est orienté par défaut vers la droite. Les programmes ainsi proposés s'apparentent à ceux utilisés, dès le cycle 1, pour piloter un robot qui devrait se déplacer sur un trajet ayant la forme d'un carré¹⁴.

Un programme très basique utiliserait les instructions « avancer de ... » et une des deux instructions de changement de direction précédentes. Son exécution produirait un carré quasi-instantané sans véritablement percevoir le tracé de la figure.

Nous nous sommes fixé deux objectifs pour réaliser le programme de construction du carré. Le premier est de construire une figure dynamique qui s'appuie sur certaines propriétés géométriques du carré. Le second objectif est de mettre en lumière certaines caractéristiques propres au logiciel Scratch liées aux variables et aux attributs¹⁵.

Pour cela, nous avons choisi l'instruction « glisser en ... secondes à x : ... y : ... » pour introduire une temporalité dans la construction, une continuité visuelle proche du tracé papier-crayon mais aussi pour pointer dans cet article les différences et les ressemblances entre les attributs du lutin utilisés dans l'instruction « glis-

ser en ... secondes à x : ... y : ... » et d'éventuelles variables déclarées.

Pour le changement de direction, l'instruction « tourner ... de ... degrés » s'est imposée afin de positionner simplement le lutin dans le sens du déplacement. L'orientation peut sembler superflue pour l'enseignant avec l'utilisation de « glisser » mais, lors de la construction visuelle à l'écran, il est important que dès les premières utilisations, un élève se repère et se positionne mentalement : le lutin doit suivre cette orientation pour favoriser cette appropriation.

On s'aperçoit lors d'activités en classe en sixième ou en cinquième, que cette orientation par écran interposé est loin d'être évidente : il n'est pas rare que l'élève soit amené à mimer le lutin pour appréhender l'orientation et le déplacement. Le choix du lutin s'avère finalement important pour conserver la dimension de la direction dans le déplacement visuel bien que l'instruction « glisser » ne nécessite pas d'orientation particulière.

Les diverses pauses évocatrices¹⁶ (« penser », « attendre ») permettent de percevoir les différents moments de la construction du carré (le déplacement, le changement d'orientation) nécessaires à la connaissance de la figure.

Les trois programmes proposés ci-contre ne sont pas des attendus pour des élèves de sixième mais peuvent servir de base pour réaliser une vidéo de la construction pas à pas d'un

14 Catherine Prunet, « Ozobot à l'école maternelle », Revue PLOT n° 58, 2ème trimestre 2017, APMEP. Initiation à la programmation aux cycles 2 et 3, http://cache.media.eduscol.education.fr/file/Initiation_a_la_programmation/92/6/RA16_C2_C3_MATH_initiation_programmation_doc_maitre_624926.pdf consulté le 22 juin 2017.
15 Voir annexe.

16 Antoine de La Garanderie, *Comprendre et imaginer : les gestes mentaux et leur mise en œuvre*, Bayard Editions, Paris, 1987.

Script 1

```

quand le drapeau est cliqué
mettre la couleur du stylo à
mettre la taille du stylo à 5
stylo en position d'écriture
dire Je commence à dessiner pendant 2 secondes
glisser en 3 secondes à x: abscisse x + 100 y: ordonnée y
attendre 0.2 secondes
tourner de 90 degrés
penser à Eh hop !!! pendant 2 secondes
attendre 1 secondes
glisser en 3 secondes à x: abscisse x y: ordonnée y + 100
attendre 0.2 secondes
tourner de 90 degrés
penser à Eh hop !!! pendant 2 secondes
attendre 1 secondes
glisser en 3 secondes à x: abscisse x - 100 y: ordonnée y
attendre 0.2 secondes
tourner de 90 degrés
penser à Eh hop !!! pendant 2 secondes
attendre 1 secondes
glisser en 3 secondes à x: abscisse x y: ordonnée y - 100
attendre 0.2 secondes
tourner de 90 degrés
relever le stylo
dire Le carré est terminé pendant 2 secondes
    
```

Script 3

```

quand le drapeau est cliqué
mettre abscisse à abscisse x
mettre ordonnée à ordonnée y
mettre la couleur du stylo à
mettre la taille du stylo à 5
stylo en position d'écriture
dire Je commence à dessiner pendant 2 secondes
glisser en 3 secondes à x: abscisse + 100 y: ordonnée
attendre 0.2 secondes
tourner de 90 degrés
penser à Eh hop !!! pendant 2 secondes
attendre 1 secondes
glisser en 3 secondes à x: abscisse + 100 y: ordonnée + 100
attendre 0.2 secondes
tourner de 90 degrés
penser à Eh hop !!! pendant 2 secondes
attendre 1 secondes
glisser en 3 secondes à x: abscisse y: ordonnée + 100
attendre 0.2 secondes
tourner de 90 degrés
penser à Eh hop !!! pendant 2 secondes
attendre 1 secondes
glisser en 3 secondes à x: abscisse y: ordonnée
attendre 0.2 secondes
tourner de 90 degrés
relever le stylo
dire Le carré est terminé pendant 2 secondes
    
```

Script 2

```

quand le drapeau est cliqué
mettre la couleur du stylo à
mettre la taille du stylo à 5
stylo en position d'écriture
mettre abscisse à abscisse x
mettre ordonnée à ordonnée y
ajouter à abscisse 100
dire Je commence à dessiner pendant 2 secondes
glisser en 3 secondes à x: abscisse y: ordonnée
attendre 0.2 secondes
tourner de 90 degrés
penser à Eh hop !!! pendant 2 secondes
attendre 1 secondes
ajouter à ordonnée 100
glisser en 3 secondes à x: abscisse y: ordonnée
attendre 0.2 secondes
tourner de 90 degrés
penser à Eh hop !!! pendant 2 secondes
attendre 1 secondes
ajouter à abscisse -100
glisser en 3 secondes à x: abscisse y: ordonnée
attendre 0.2 secondes
tourner de 90 degrés
penser à Eh hop !!! pendant 2 secondes
attendre 1 secondes
ajouter à ordonnée -100
glisser en 3 secondes à x: abscisse y: ordonnée
attendre 0.2 secondes
tourner de 90 degrés
relever le stylo
dire Le carré est terminé pendant 2 secondes
    
```

carré : ils sont là afin d'illustrer et balayer les particularités du logiciel Scratch, mais également pour aider les enseignants à prendre conscience des rôles précis et des spécificités de chaque instruction. Rappelons que Scratch est un logiciel de programmation visuelle contrairement à d'autres langages textuels.

Ces programmes permettent une construction du carré à partir de n'importe quelle position du lutin dans la scène, sous réserve de la place disponible, bien évidemment. Ils utilisent l'instruction « glisser » qui nécessite l'emploi des coordonnées. En conséquence, l'usage de variables est indispensable.

Analysons pour ces trois scripts la nature des variables utilisées et leur comportement durant le déroulement du script.

Les trois scripts utilisent dans Scratch les attributs de l'objet lutin « abscisse x » et « ordonnée y »¹⁷ : ils sont représen-

17 Voir annexe.

tés par des briques ayant la même forme qu'une variable informatique (voir l'onglet Données) mais avec une couleur différente liée à leur emplacement dans les menus, du fait de leur nature et de leur mode de fonctionnement. Les deux derniers scripts contiennent aussi des variables créées à partir de l'onglet Données : elles sont de couleur orange nommées « abscisse » et « ordonnée ». On peut noter que les attributs « abscisse x » et « ordonnée y » ne peuvent jamais être affectés à l'aide d'une instruction du type « mettre... » ou « ajouter à ... » comme pour une variable déclarée mais sont modifiés par des méthodes propres aux différents attributs. Le déplacement du lutin peut aussi modifier certains de ces attributs.

Les attributs et les variables ont une forme commune et peuvent être placés dans les mêmes instructions. On pourrait penser que ces éléments ont une nature semblable. Pourtant, leur comportement est assez différent.

Les attributs « abscisse x » et « ordonnée y » ont accès aux coordonnées du lutin attribué (usage privé) pour fournir en sortie chacune d'elles : ce phénomène se produit à tout moment durant le déroulement du script (il suffit d'afficher leur valeur pour observer leurs modifications). En cela, ces attributs diffèrent des variables déclarées qui ne peuvent être modifiées que par une instruction d'affectation. Dans le premier script, l'instruction « glisser en 3 secondes à x : abscisse x +100... » est une méthode qui conduit à déplacer le lutin de sa position initiale à une nouvelle position dont l'abscisse est augmentée de 100. Cependant l'attribut « abscisse x » va évoluer durant le déplacement alors que la valeur « abscisse x + 100 » utilisée dans l'instruction est une instance de l'attribut qui est égale à l'abscisse initiale augmentée de 100. Lorsqu'on appelle la méthode « glisser » en utilisant l'attribut « abscisse x » dans les coordonnées à atteindre, c'est la valeur de

cet attribut qui est donnée en paramètre à la méthode : les coordonnées à atteindre sont évaluées une seule fois au départ en utilisant les valeurs des attributs. C'est ce qu'on appelle un passage par valeur. N'observe-t-on pas ici deux comportements différents d'une même brique au même instant ? Le problème de la temporalité s'exprime ici : temporalité entre le déclenchement d'une instruction et son déroulement, temporalité entre deux instructions. Dans la deuxième instruction « glisser » de ce premier script, l'attribut « abscisse x » ne renvoie pas la position initiale du lutin mais sa position finale après le premier glissement.

Que renvoient les attributs « ordonnée y » dans la troisième instruction « glisser » et « abscisse x » dans la quatrième ? On observe quatre instructions « glisser » différentes pour quatre déplacements différents.

Dans le deuxième script, les variables créées « abscisse » et « ordonnée » (usage privé ou public suivant la définition choisie : pour ce lutin ou pour tous les lutins)¹⁸ ne changent pas durant les déplacements du lutin mais varient au gré des instructions « ajouter à ... ». En parallèle, les attributs « abscisse x » et « ordonnée y » varient en permanence mais ne sont utilisées qu'une seule fois lors de l'initialisation des variables déclarées dans le script. On observe ainsi que l'on a écrit quatre fois la même instruction « glisser » qui réalise pourtant des déplacements différents.

Dans le dernier script, les variables créées « abscisse » et « ordonnée » sont initialisées comme précédemment mais elles ne varient jamais durant l'exécution du script. Elles se comportent comme des lettres évaluées, statut utilisé en mathématiques. Ces deux variables contiennent les coordonnées de la position de

18 Voir Annexe.

départ du lutin et les instructions « glisser » utilisent les coordonnées relatives des trois autres sommets du carré par rapport au premier sommet.

Cette première analyse montre la difficulté, pour un élève de 6ème, voire de début de 5ème, de donner sens à ces différents éléments variables et à leur fonctionnement en lien avec les instructions qui les utilisent.

On remarque que le logiciel Scratch utilise à la fois des variables informatiques déclarées et des attributs dans différents onglets du menu qui semblent se comporter comme ces variables (ici : abscisse x , ordonnée y). Ces attributs et leurs méthodes constituent une spécificité de ce logiciel qui propose une programmation dite orientée objet (POO) : en effet, d'autres langages de programmation, moins visuels, comme Algobox, nécessitent l'usage exclusif de variables informatiques définies par l'utilisateur. Est-ce que cette spécificité, à ce niveau d'enseignement, ne constitue pas un obstacle à la construction du statut de variable informatique ? Le travail de programmation devrait donc peut-être limiter, dans un premier temps, l'utilisation de variables informatiques et aussi éviter le mélange de ces variables avec les attributs des lutins.

Pour terminer au sujet de ces constructions, on peut utiliser la méthode « glisser » à condition de définir la position fixe de départ du lutin : la méthode « aller à $x : \dots y : \dots$ » peut être utilisée pour positionner le lutin au départ et donc éviter les « sorties de l'écran ». On peut également, si nécessaire, préciser à cette occasion l'orientation du lutin vers la droite. Cette instruction d'orientation, ainsi que les instructions « tourner ... de ... degrés », ne sont utiles que pour aider à l'appréhension visuelle de la construction (avec le lutin « Beetle » par exemple). Avec ce positionnement initial du lutin,

on peut utiliser les coordonnées des trois autres sommets dans les instructions « aller à ... » ou « glisser en ... à ... » et se dispenser ainsi de faire appel aux attributs de position du lutin.

Au niveau des programmes, les coordonnées sont restées dans le cycle 4 (et il n'y a pas de repère de progressivité sur ce point même s'il semble cohérent de les garder sur le niveau 5ème).

Ainsi la stratégie la plus pertinente semble donc d'utiliser l'instruction « avancer de ... » en lieu et place de l'instruction « glisser » : les problèmes des coordonnées et des variables sont ainsi écartés et les élèves doivent alors utiliser en acte certaines des propriétés du carré pour réaliser le programme demandé. Mais « avancer de... » donne plutôt l'impression d'un saut du point de départ vers le point d'arrivée désigné par la distance et l'orientation (coordonnées polaires) ; l'instruction « glisser » épouse davantage le déplacement évoquant le tracé du segment bien qu'elle s'appuie sur les coordonnées cartésiennes de l'extrémité... D'autre part, la mesure d'angle utilisée dans l'instruction « tourner ... de ... degrés » ne correspond pas à celle utilisée pour le tracé sur papier.

On peut noter que cette manière de dessiner le carré n'est ni celle utilisée en papier crayon, ni celle utilisée avec un logiciel de géométrie dynamique comme GeoGebra. On pourrait imaginer revenir sur l'activité « dessine-moi un carré » en 5ème, non pas en travaillant sur l'aspect géométrique mais par l'aspect plus numérique du repérage dans le plan. L'exercice peut alors être fait en utilisant les instructions « aller à ... » pour donner les coordonnées de départ et « glisser à ... » (avec des coordonnées déterminées en amont) pour effectuer les déplacements. Un prolongement possible pourrait alors être de créer des figures plus complexes à partir des coordonnées des som-

LE LOGICIEL
SCRATCH AU COLLEGE...

```

quand cliqué
demander Choisir un nombre et attendre
mettre nombre à réponse
montrer la variable nombre
dire J'ajoute 5 à ce nombre pendant 2 secondes
mettre nombre à nombre + 5
penser à nombre pendant 2 secondes
dire Je multiplie le résultat par 6 pendant 2 secondes
mettre nombre à nombre * 6
penser à nombre pendant 2 secondes
dire Je soustrais 27 à ce résultat pendant 2 secondes
mettre nombre à nombre - 27
penser à nombre pendant 2 secondes
dire Je divise le résultat obtenu par 3 pendant 2 secondes
mettre nombre à nombre / 3
penser à nombre pendant 2 secondes
dire Le résultat final est pendant 2 secondes
dire nombre
attendre 10 secondes
cacher la variable nombre
stop tout
    
```

Script 1

```

quand cliqué
demander Choisir un nombre et attendre
mettre nombre à réponse
montrer la variable nombre
dire J'ajoute 5 à ce nombre pendant 2 secondes
mettre résultat à nombre + 5
penser à résultat pendant 2 secondes
dire Je multiplie le résultat par 6 pendant 2 secondes
mettre résultat à résultat * 6
penser à résultat pendant 2 secondes
dire Je soustrais 27 à ce résultat pendant 2 secondes
mettre résultat à résultat - 27
penser à résultat pendant 2 secondes
dire Je divise le résultat obtenu par 3 pendant 2 secondes
mettre résultat à résultat / 3
penser à résultat pendant 2 secondes
dire Le résultat final est pendant 2 secondes
dire résultat
attendre 10 secondes
cacher la variable nombre
stop tout
    
```

Script 2

```

quand cliqué
demander Choisir un nombre et attendre
mettre nombre à réponse
montrer la variable nombre
dire J'ajoute 5 à ce nombre pendant 2 secondes
penser à nombre + 5 pendant 2 secondes
dire Je multiplie le résultat par 6 pendant 2 secondes
penser à nombre + 5 * 6 pendant 2 secondes
dire Je soustrais 27 à ce résultat pendant 2 secondes
penser à nombre + 5 * 6 - 27 pendant 2 secondes
dire Je divise le résultat obtenu par 3 pendant 2 secondes
penser à nombre + 5 * 6 - 27 / 3 pendant 2 secondes
dire Le résultat final est pendant 2 secondes
dire nombre + 5 * 6 - 27 / 3
attendre 10 secondes
cacher la variable nombre
stop tout
    
```

Script 3

mets par exemple et en faisant des allers-retours entre la programmation et le papier-crayon¹⁹. On voit ici des exemples des différentes ouvertures vers les mathématiques que peut offrir la programmation.

Un programme de calcul

Analysons maintenant une autre situation destinée au cycle 4. Elle porte sur un programme de calculs et sur les scripts associés. Voici trois méthodes différentes pour traduire une suite de calculs qui produit le résultat $((x + 5) \times 6 - 27) \div 3$ sur le nombre de départ nommé x . Comme dans la construction du carré, nous souhaitons faire apparaître successivement les étapes du programme de calcul et les résultats intermédiaires. Une nouvelle fois, ces trois scripts nous servent de prétexte pour

19 On pourrait envisager des activités autour de l'écriture de chiffres ou de lettres, par exemple.

analyser le comportement des variables déclarées et de l'attribut « réponse ».

Par souci de cohérence pour cette analyse, nous utilisons une nouvelle fois le logiciel Scratch bien que nous pensions que cet outil ne soit pas le plus pertinent pour une telle activité. Un langage comme Algobox semble en effet plus adapté, en particulier ici pour l'écriture de l'expression littérale finale qui est proche de l'écriture mathématique : avec Scratch, l'absence de parenthésages, remplacés par l'utilisation de blocs de calcul imbriqués, ne rend pas aisée la lecture de la dernière instruction et la production du résultat final lorsque le script est fourni aux élèves.

Cependant, lors de l'écriture d'un script à partir d'une expression algébrique donnée, cette nécessité d'empiler les différentes briques de calculs oblige l'élève à une analyse pertinente de l'expression algébrique prenant en compte les priorités opératoires. Cet aller-retour entre empilement de briques et parenthésage peut dans certains cas renforcer la compréhension du fonctionnement des priorités opératoires. Cependant certains calculs nécessitent d'empiler les briques dans un certain ordre alors que le calcul mathématique n'en nécessite pas comme par exemple pour $3 \times x \times x$. Cela pourrait induire des usages erronés des priorités opératoires déjà marqués chez les élèves en difficulté. On retrouve cependant avec Algobox les mêmes stratégies pour traduire la suite de calculs qui induisent donc la même perception de la variable informatique.

Le premier script proposé n'utilise qu'une seule variable nommée « nombre ». L'attribut « réponse » est un attribut commun à tous les lutins (attribut de classe d'objets)²⁰ : il ne sert en réalité ici que de mémoire tampon pour capturer la valeur du nombre entré au départ. En

POO, « réponse » est considérée comme une propriété statique et l'instruction « demander... et attendre », une méthode statique associée. On pourrait d'ailleurs imaginer ne pas utiliser cette fonctionnalité pour récupérer ce nombre mais simplement l'écrire directement dans l'instruction « mettre nombre à ... » : on devrait alors modifier directement le script pour appliquer le programme sur une autre valeur. Mais est-ce là une image correcte ou acceptable de la variable mathématique liée à la notion de fonction ? L'écriture du nombre directement dans le script réduit ce dernier en un calcul d'une image, la variable déclarée « nombre » se comportant comme une lettre évaluée et non comme une variable mathématique.

Dans ce script, la valeur de la variable « nombre » évolue à chaque nouveau calcul : elle sert à la fois de variable libre (nombre de départ) et de variables dépendantes (nombres intermédiaires et nombre d'arrivée). On retrouve ici l'idée de « boîte noire » (ou plutôt transparente dans ce cas...) : on entre un nombre dans la boîte et il se modifie au gré des calculs et un résultat en ressort... Le fait de « montrer la variable... » permet de visualiser les différentes valeurs de cette variable « nombre ». Cette stratégie de programmation relève davantage d'une démarche d'informaticien que de celle de mathématicien. Elle ne facilite pas la perception de la fonction numérique comme une relation entre deux nombres : l'antécédent et l'image ne peuvent en effet être présents simultanément mais existent successivement. La notion de variable informatique est ici assez éloignée du statut de variable mathématique introduit au collège. Au lycée, certaines tâches mathématiques se rapprochent de ce travail d'affectation du type « mettre nombre à nombre + 6 » : par exemple, lorsqu'on étudie la parité d'une fonction, l'élève doit remplacer x par $-x$ ou encore lorsqu'il doit produire le terme de rang $n + 1$ d'une suite connaissant le terme de rang n .

²⁰ Voir Annexe.

Dans le deuxième script, les deux variables (libre et dépendante) sont, cette fois, clairement identifiées sous les terminologies « nombre » et « résultat ». Leurs valeurs peuvent être affichées simultanément dans la scène. Le fait de « penser... » le « résultat » à chaque étape du programme de calcul permet de visualiser le contenu de cette variable et évoque l'idée que l'on retient dans sa tête (en mémoire) une valeur intermédiaire... Cependant, la difficulté liée à l'affectation par valeur d'une variable perdure : par exemple, dans l'instruction « mettre résultat à résultat - 27 » la variable « résultat » et sa valeur portent le même nom, ce qui prête à confusion. De plus, la dernière instruction ne contient pas l'ensemble des opérations effectuées : chaque calcul a été absorbé par la variable nommée « résultat » au fur et à mesure du déroulement du script. Cette stratégie de programmation ne permet pas une vision globale du programme de calcul mais offre seulement un aspect procédural pas à pas des actions effectuées comme on peut le faire dans un tableur avec des références relatives de cellules.

Le script 4 ci-contre est une variante de ce deuxième script qui permet de distinguer le calcul à partir de la variable « calcul » (aspect procédural) du résultat à partir de la variable « résultat » (aspect structural). On peut noter ici l'importance du choix des noms des variables informatiques qui évoquent leur rôle dans le script. Cependant, cela n'améliore pas la perception du programme de calcul dans sa globalité. Pourtant, bien que cela alourdisse le script, cette distinction dans l'écriture évite de recourir à des instructions du type « mettre *résultat* à *résultat* + 5 » par exemple dont la compréhension reste difficile à ce niveau d'enseignement.

Le troisième script proposé initialement rapproche les instances informatique et mathématique. Nous y retrouvons les deux variables

Script 4

Script 5

libre et dépendante bien que cette dernière ait toujours la valeur qui évolue au cours du déroulement du programme. Les deux aspects procédural et structural y sont bien présents : nous trouvons dans les dernières instructions une

expression qui est semblable à l'expression littérale attendue : $((x + 5) \times 6 - 27) \div 3$. Une tâche mathématique du type « Trouver un programme de calcul équivalent mais plus simple. » peut alors être engagée.

Le script 5 de la page précédente est une variante du troisième script. L'attribut « réponse » joue le rôle de la variable informatique « nombre ». Sa rédaction s'appuie sur la démarche mathématique de programme de calcul et ne requiert pas de variable informatique : cette situation peut donc être proposée assez tôt au cycle 4. Mais ce procédé de saisie est spécifique au logiciel Scratch : dans d'autres langages, la saisie de données nécessite le recours à des variables informatiques. Cette démarche peut permettre de travailler la lettre en mathématiques sans la mettre en conflit avec le statut de variable informatique. Il sera toujours temps de travailler ce statut dans la seconde moitié du cycle 4.

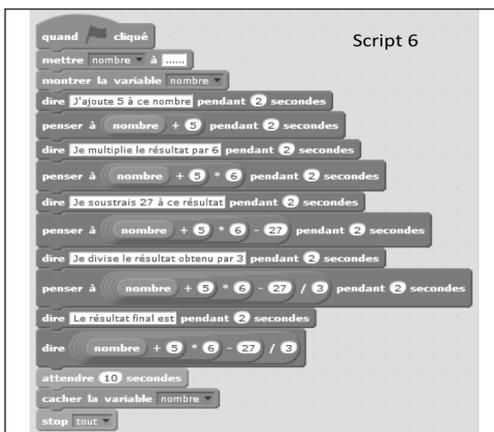
Cette dernière variante (script 6, ci-dessous) utilise une variable informatique nommée « nombre » dont le contenu est saisi par l'utilisateur directement dans le script avant son exécution. La rédaction du script suit le program-

me de calcul et permet ainsi de faire le lien avec les mathématiques. La variable informatique « nombre » se rapproche alors du statut de variable mathématique : la saisie de la valeur de « nombre » évoque le statut de lettre évaluée. Dans ces deux derniers scripts, il n'y a pas de variable dépendante (image) qui contient le résultat final : celui-ci est simplement évoqué à travers l'instruction « dire ». On aurait pu définir une variable « image » qui prendrait la valeur de ce résultat final : cela renforcerait l'idée d'antécédent et d'image pour se rapprocher de la notion mathématique de fonction (à envisager en fin de cycle 4).

D'autres scripts sont bien évidemment envisageables en fonction des objectifs fixés (lorsqu'on veut par exemple pouvoir modifier rapidement le script lorsque certains paramètres du programme de calcul changent) ou des calculs à effectuer.

Les deux activités présentées avec les différents scripts envisageables montrent des différences significatives entre les notions de variables informatique et mathématique. Leurs comportements et leurs traitements sont de natures différentes voire parfois antinomiques. Voici un autre exemple classique : l'affectation à la variable a de la valeur $a + 1$ qui peut se noter, suivant le langage employé, « $a:=a+1$ » ou tout simplement « $a=a+1$ » quand ce n'est pas « $a+1 \rightarrow a$ » ou « $a+=1$ » ou « $a++$ ».

Le statut de variable informatique s'oppose ici frontalement à celui d'inconnue de l'équation $a=a+1$. Les variables informatique et mathématique doivent donc naître et cohabiter mais ne peuvent pas dépendre l'une de l'autre. On peut cependant, dans certaines circonstances, les rapprocher comme dans le dernier script proposé. On doit également s'attacher à expliciter davantage la notion de variable dans les scripts. Pour cela, il faudra veiller à



un nommage clair des variables : plutôt que d'avoir recours à une lettre, un nom explicite (résultat, score, vies...) permettra à l'élève de s'approprier le sens et le comportement de cette variable. D'autre part, les instructions « ajouter à score 1 » ou « ajouter à vies 0-1 » (préférable en 6ème à « ajouter à vies -1 ») seront à privilégier par rapport à « mettre score à score+1 » ou « mettre vies à vies-1 ».

Egalités en mathématiques et en informatique.

L'égalité en informatique se décline suivant deux modalités : l'affectation et le test d'égalité. La première est orientée donc dissymétrique alors que le second est parfaitement symétrique. L'égalité s'écrit dans différents langages selon deux symboliques différentes. Par exemples, sous Scilab ou Python, on utilise le « = » pour l'affectation et le « == » pour le test d'égalité tandis que sur la calculatrice au lycée de Texas Instrument TI 82 ou 83, on utilise le symbole de stockage « → » pour la première et le « = » pour le second. Sous Scratch, l'affectation n'existe qu'en lien avec une variable créée (« mettre ... à ... ») alors que la brique « ... = ... » est disponible dans le menu des opérateurs pour le test d'égalité. Ainsi selon le langage utilisé, le symbole « = » peut désigner une affectation ou un test d'égalité. Cependant, l'affectation n'est pas une égalité en tant que telle mais utilise une symbolique qui renvoie à cette idée d'égalité.

En mathématiques, les statuts de l'égalité et de la lettre sont intimement liés. On peut retrouver des aspects de l'affectation en informatique et du test d'égalité mais la relation en mathématiques est symétrique et induit principalement la substitution des écritures. Par exemple, lorsque $a = 10$, on peut remplacer a par 10 mais également 10 par a ; ainsi l'expres-

sion $a + 10$ est égale à 20 ou bien $2a$. Même si l'expression $a = 10$ peut évoquer l'affectation, en mathématiques, on peut obtenir deux réponses alors qu'en informatique, $a = 10$ relève de l'affectation mais $10 = a$ est une erreur de syntaxe. Dans l'exemple $y = 3x + 7$, on retrouve l'affectation en informatique avec une relation univoque, à condition que la variable informatique x soit initialisée (sinon, en général, soit un message d'erreur est affiché, soit le résultat est indéterminé). En mathématiques, l'égalité $y = 3x + 7$ relève de la dépendance réciproque des deux variables x et y ($y = 3x + 7$ ssi $x = (y - 7)/3$) : cette égalité ne nécessite pas de valeurs initiales de ces deux variables. Cependant cette égalité peut être considérée comme une équation avec une valeur de vérité suivant les valeurs de x et de y : en effet, suivant les valeurs de ces deux inconnues, les égalités obtenues sont vraies ou fausses (elles seraient toutes vraies dans le cas d'une identité et les inconnues seraient alors des indéterminées). On est alors sur un test d'égalité comparable à ce que l'on trouve en informatique. On retrouve davantage cette idée d'affectation dans une expression mathématique du type $f(x) = 3x + 7$.

Que peut bien signifier l'expression $b = y = 3x + 7$? En mathématiques, bien que cette écriture ne soit pas sans difficulté pour des élèves, on peut y voir trois signifiants pour un signifié c'est-à-dire trois représentants du même objet. On peut aussi y voir la transitivité de l'égalité : $b = y$ et $y = 3x + 7$ implique $b = 3x + 7$. En informatique, les deux égalités n'ont pas toujours le même statut : tout dépend de l'interprétation de ce « égal ». Suivant le langage, il peut être considéré comme une affectation ou bien un test d'égalité. La succession de « égal » dans $b = y = 3 * x + 7$ peut donc être interprétée comme une affectation simultanée des variables b et y (c'est le cas en Python ou en langage C). Cependant au moins l'un des « égal » pourrait être un test égalité : il s'écrit

rait assez souvent à l'aide d'un double signe (\equiv) puisque le signe $=$ sert alors à l'affectation (comme en Scilab ou en Python).

On retrouve une écriture semblable avec un tableur comme Excel après avoir nommé des cellules ou des plages de cellules y et x : ainsi une cellule nommée b et contenant la formule $=y=3*x+7$ contiendra la valeur VRAI ou FAUX (0 ou 1 sur d'autres tableurs)²¹.

Enfin avec le logiciel Scratch qui réalise l'affectation d'une variable sans le signe $=$ (« mettre ... à ... »), on peut écrire avec l'opérateur « ... = ... » l'expression $b=y=3*x+7$: ici on a obligatoirement empilé deux briques  dans un certain ordre.

Il y a donc deux interprétations : soit $(b=y)=3*x+7$ qui retourne FAUX car $b=y$ possède une valeur de vérité alors que $3*x+7$ n'en possède pas, soit $b=(y=3*x+7)$ qui retourne VRAI ou FAUX suivant la situation (b pourrait contenir par affectation la valeur *true* ou la valeur *false* en utilisant l'instruction « mettre b à *true* », *true* étant à la fois une chaîne de caractère et une valeur de vérité reconnue par le logiciel).

Pour conclure, la dissymétrie de l'affectation constatée généralement en informatique est de nature à renforcer l'aspect dynamique du « = » en mathématiques²² très prégnant chez les élèves du collège alors que l'enseignant s'évertue à installer son aspect statique avec la substitution des écritures. Si on ajoute à cela les difficultés liées aux différences entre les lettres en

mathématiques et la variable en informatique, l'enseignant devra faire preuve d'une grande vigilance quant aux activités proposées et à leur progressivité sur ces sujets en cycle 4.

L'équation en mathématiques se rapproche cependant du test d'égalité en informatique, présent dans une instruction conditionnelle du type « Si ... alors ... (sinon ...) ».

Si ... alors ... versus si ... alors ... (test et implication)

Soient a , b et c trois nombres strictement positifs rangés dans l'ordre croissant.



Théorème T :

si $c \times c = b \times b + a \times a$, alors
on peut dire que le triangle est rectangle.

Les deux éléments ci-dessus se ressemblent dans leur rédaction et leur utilisation. Pourtant, ils sont très différents.

Le théorème T est constitué de deux propositions qui peuvent être vraies ou fausses. L'instruction de contrôle utilisée dans Scratch est composée de deux éléments de natures différentes : un test d'égalité qui retourne vrai ou faux et une instruction qui déclenche une action. Le test d'égalité est de nature semblable à la proposition qui suit le « si » dans le théorème T alors que ce qui suit le « alors » est de nature différente.

En mathématiques, le théorème T a une existence propre : c'est une proposition T vraie. Il induit une temporalité de la connaissance et non de l'action : le triangle est tel qu'il est et l'utilisation du théorème permet de mieux connaître

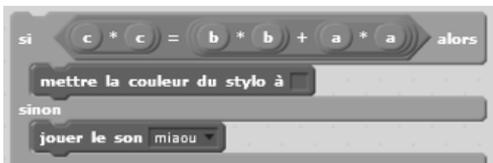
21 On peut trouver davantage de détails sur ce sujet dans l'article « Le sens de la formule » de Repères IREM n° 92 de juillet 2013 sur le tableur et les écritures littérales réalisés par les mêmes auteurs.

22 « Le sens de la formule », Repères IREM n° 92 pages 83-103, juillet 2013.

sa nature ; les deux propositions existent simultanément mais on y accède successivement. En informatique, le test d'égalité déclenche ou non une action suivant la valeur de vérité du test (VRAI ou FAUX). L'action est subordonnée au test ce qui induit une temporalité de l'action. C'est le temps de l'action et pas de la connaissance. Cette action peut très bien appartenir à un champ différent de celui du test ; par exemple :



On peut aussi compléter cette instruction conditionnelle à l'aide d'un « sinon... » qui déclenche alors une autre action dans le cas où le test retourne FAUX. Par exemple :



En mathématiques, le « sinon » n'est pas employé mais il est parfois sous-entendu lorsque la réciproque du théorème est vraie (et donc la contraposée de la réciproque également). Dans le cas du théorème T , on pourrait énoncer la propriété sous la forme :

si $c \times c = b \times b + a \times a$, alors
on peut dire que le triangle est rectangle.
sinon
on peut dire que le triangle n'est pas rectangle.

Souvent une implication mathématique induit la réciproque comme théorème élève, que cette dernière soit vraie ou fausse. Par exemple, la propriété vraie « si A est le milieu de $[BC]$

alors $AB = AC$ » est associée par certains élèves à la propriété fausse « si $AB = AC$ alors A est le milieu de $[BC]$ ». Ainsi si A n'est pas le milieu de $[BC]$ alors on ne peut rien dire à propos de l'égalité des longueurs (on peut avoir $AB = AC$ comme $AB \neq AC$). Par contre l'instruction conditionnelle utilisée en informatique ne permet pas le déclenchement d'une l'action lorsque le test est FAUX, à moins d'avoir programmé un « sinon ».

Pour conclure, le « si ... alors ... » en mathématiques et en informatique sont semblables dans leur apparence mais très différents dans leur fonctionnement. Le premier est utilisé par l'élève tandis que le second est conçu par lui dans un but précis et unique. Ceci peut donc constituer un obstacle dans les apprentissages. Cependant, travailler l'instruction conditionnelle en informatique consolide la notion de cause/conséquence du fait de l'aspect visuel et dynamique du logiciel.

En conclusion :

Au début du cycle 4, les élèves passent progressivement d'une géométrie descriptive à une géométrie de traitement. Ils sont ainsi amenés, peu à peu, à utiliser le « si ... alors ... » de l'implication dans la construction d'un raisonnement déductif : cette utilisation n'a lieu que lorsque la condition est vérifiée. Par cet aspect, travailler l'instruction conditionnelle en programmation peut *éclairer* le fonctionnement du raisonnement déductif en mathématiques.

Cependant, bien que la cause puisse être commune, la conséquence diffère par nature : l'enseignant devra donc expliciter clairement cette différence entre connaissance et action. Il devra pointer la nature de cette conséquence et faire émerger que l'élève ne peut pas se fabriquer ses propres propriétés dans un raisonne-

ment mathématique tandis qu'en programmation, l'élève doit concevoir des instructions conditionnelles en fonction des actions qu'il veut déclencher. Le cycle 4 est également le lieu de l'introduction et de l'apprentissage de la lettre dans ses différents statuts mathématiques, statuts étroitement liés à ceux de l'égalité. Introduire la variable informatique prématurément et sans précautions ne permet pas de *revisiter* cette notion de variable mathématique : l'enseignant devra introduire progressivement les différents statuts des lettres en mathématiques et en informatique en distinguant leurs fonctionnements sans chercher à s'appuyer systématiquement sur l'un pour travailler l'autre. Cet enseignement ne permet pas, à notre avis, de *renouveler* l'introduction de l'algèbre au collège.

Dans le même ordre d'idée, l'utilisation de l'opérateur « nombre aléatoire entre ... et ... » pour simuler le lancer d'une pièce ou d'un dé à six faces permet-elle de *revisiter* la notion de probabilités ? Ou bien les probabilités permettent-elles d'*éclairer* cet opérateur utilisé en programmation ? L'utilisation de l'opérateur « nombre aléatoire entre ... et ... » pour réaliser des simulations et le travail autour des probabilités sont indissociables. Habituellement, les collégiens ne s'interrogent pas sur le fonctionnement de cet opérateur. Bien évidemment, l'enseignant n'explique pas aux élèves de collèges que cet opérateur s'appuie sur des notions comme les variables aléatoires et les lois de probabilités. Il réalise des expériences réelles avec des lancers de pièces et de dés mais les élèves ne font pas le lien entre ces expériences, les simulations informatiques et les probabilités. Si l'enseignant introduit l'opérateur « nombre aléatoire entre ... et ... » comme l'objet informatique qui fonctionne à l'image du lancer d'une pièce ou d'un dé, alors il peut program-

mer des simulations sous Scratch visuellement semblables à l'expérience réelle²³. Cette démarche permet de passer, sans perte de sens, d'une expérience de taille réduite à une simulation d'une taille suffisante pour une approche fréquentiste des probabilités. Il sera toujours temps ensuite d'utiliser le tableur ou un logiciel comme Algebox pour réaliser des simulations moins visuelles, moins proches de l'expérience réelle et donc plus abstraites.

La programmation au collège devrait s'orienter davantage vers des activités intra-mathématiques à l'instar de celles sur le thème des probabilités. L'aspect visuel de Scratch est un atout qui doit permettre d'*éclairer* d'autres champs des mathématiques comme les constructions géométriques, le repérage dans le plan, les programmes de calcul... Au lycée ou en post-bac, l'algorithmique et la programmation sont principalement en relation avec les autres domaines des mathématiques.

Les logiciels utilisés après le cycle 4 sont majoritairement des langages dits de bas niveau de programmation qui s'appuient sur des éléments d'algorithmiques : ils ne sont pas utilisés pour réaliser de la programmation orientée objet (POO). Quels éléments des activités de programmation avec Scratch vont être réellement transférables pour travailler l'algorithmique et programmer au lycée puis en post-bac ? Quelles connaissances informatiques et mathématiques sont alors réellement construites et acquises ? Le choix d'un logiciel de programmation orientée objet ne complexifie-t-il pas l'entrée dans la variable informatique avec la cohabitation entre attributs et méthodes, propriétés et méthodes statiques, variables déclarées et affectation ? Qu'en est-il alors de *revisiter* la variable mathématique avec le logiciel Scratch ?²⁴

²³ Voir les activités proposées sur ce sujet sur le site de l'Irem de Rouen.

²⁴ Voir annexe.

ANNEXE

Variables et attributs d'objets, attributs de classe d'objets...

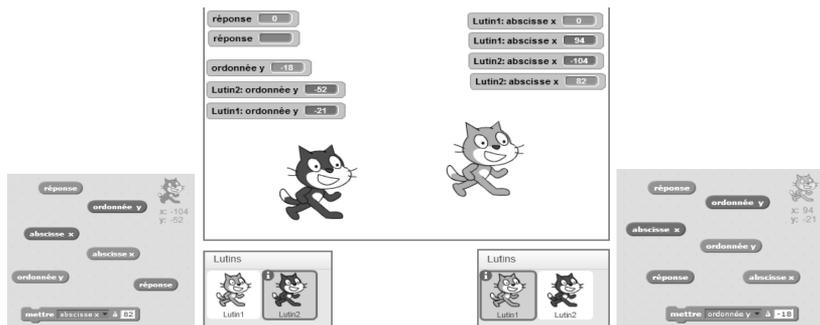
Le logiciel Scratch n'utilise pas que des variables déclarées : il convient donc de faire ici un point sur la nature et le fonctionnement des différents éléments variables. Ces informations sont à prendre en considération pour espérer *revisiter* la notion de variable mathématique par le biais de ce logiciel.

Scratch présente sur une scène un ou plusieurs lutins qui sont des **objets** ayant des **attributs**, c'est-à-dire des caractéristiques propres à chaque lutin : par exemple, « abscisse x », « ordonnée y », « costume # », « taille », etc. Lorsqu'on affiche un de ces attributs dans la scène, il est précédé du nom du lutin dont il dépend. On rencontre aussi des attributs communs à tous les lutins : ce sont des **attributs de classe**, encore appelés propriétés statiques. Un attribut de classe fréquemment rencontré est « réponse » ; on a aussi « souris x », « souris y », « chronomètre », etc.

On associe, à chaque attribut, une ou plusieurs **méthodes** : ce sont des instructions qui permettent de modifier la valeur d'un ou plusieurs attributs. Par exemple, « donner la valeur ... à x » est une méthode qui modifie l'attribut « abscisse x » ; « aller à x : ... y : ... » est une méthode qui modifie les attributs « abscisse x » et « ordonnée y » ; ces méthodes permettent d'affecter ces attributs... On rencontre des **méthodes de classe** d'objets (méthodes statiques) qui permettent de modifier une propriété statique : ainsi « demander... et attendre » est une méthode statique de l'attribut de classe « réponse ». Déplacer la souris modifie les attributs de classe « souris x » et « souris y » : c'est donc une méthode statique sans instruction pour ces deux attributs de classe... On peut aussi créer (déclarer) des **variables** dans l'onglet « Données » qui peuvent être *pour tous les lutins* (variables **publiques**) ou *pour ce lutin uniquement* (variables **privées**) : dans le cas d'une variable privée, l'affichage dans la scène de cette variable est précédé du nom du lutin dont elle dépend et sa valeur ne peut être utilisée que dans la zone de script de ce lutin. Une variable publique n'est précédée d'aucun nom de lutin et sa valeur peut être utilisée dans tous les scripts. On peut affecter une variable déclarée à l'aide d'une instruction d'affectation comme « mettre ... à ... ».

Une variable peut aussi être déclarée lors de la création d'un « bloc » (onglet « Ajouter blocs »), sorte de procédure (sous-programme) : cette variable est alors **locale** à ce bloc, c'est-à-dire que sa valeur est inaccessible par le nom de cette variable depuis n'importe quel lutin (elle n'est ni privée, ni publique) ; elle n'est pas affichable dans la scène (aucune case à cocher) et elle n'est affectée d'une valeur que lorsqu'on appelle le bloc dans un script. On peut alors dire qu'une variable déclarée est **globale** (soit globale et privée, soit globale et publique).

Voici les zones de script de chacun des deux lutins et la scène dans lesquelles des variables et des attributs cohabitent :



Quelles sont les variables globales publiques déclarées ? Que contient « ordonnée y » ?²⁵ Quelle différence y a-t-il entre « réponse », variable globale publique et « réponse », attribut de classe ? La couleur ? On peut souligner le fait qu'on ne peut affecter directement l'attribut de classe « réponse ». Quelle différence entre « abscisse x », variable déclarée globale privée et l'attribut « abscisse x » de chacun des lutins ? Les variables déclarées, bien que privées, n'influencent pas la position des lutins (tout comme la variable globale publique « ordonnée y »).

²⁵ Les variables globales publiques sont de couleur orange : « réponse » qui contient 0 et « ordonnée y » qui contient -18. Les variables globales privées sont aussi de couleur orange : « abscisse x » pour chaque lutin, l'une contient 0 (lutin 1) et l'autre 82 (lutin 2). En bleu foncé, on distingue les attributs pour chacun des lutins : « abscisse x » et « ordonnée y » qui contiennent les coordonnées des positions de chaque lutin. En bleu clair, l'attribut de classe « réponse » est vide.