Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE L'ÉDUCATION NATIONALE, DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE

éduSCOL Informer et accompagner les professionnels de l'éducation

CYCLES 2 3 4

MATHÉMATIQUES

Algorithmique et programmation

Algorithmique et programmation

SOMMAIRE

Introduction

Comme l'indique le programme, l'enseignement de l'informatique au cycle 4 n'a pas pour objectif de former des élèves experts, ni de leur fournir une connaissance exhaustive d'un langage ou d'un logiciel particulier, mais de leur apporter des clés de décryptage d'un monde numérique en évolution constante. Cet enseignement permet d'acquérir des méthodes qui construisent la pensée algorithmique et développe des compétences dans la représentation de l'information et de son traitement, la résolution de problèmes, le contrôle des résultats. Il fournit également l'occasion de mettre en place des modalités d'enseignement fondées sur une pédagogie de projet, active et collaborative.

Le logiciel Scratch offre un environnement d'édition et d'exécution des programmes. Il s'agit d'un logiciel gratuit et disponible sur toutes les plates-formes usuelles, choisi pour sa simplicité, sa fiabilité et sa robustesse dans la mise en œuvre. Il permet de travailler tous les concepts figurant au programme, en particulier la programmation événementielle et la gestion de scripts s'exécutant en parallèle. L'annexe de ce document explique comment l'installer.

Objectifs de la formation

La lettre de saisine du Conseil supérieur des programmes datée du 19 décembre 2014 précisait les objectifs et démarches d'apprentissages : « L'enseignement de l'informatique et de l'algorithmique au cycle 4 n'a pas pour objectif de former des élèves experts, mais de leur apporter des clés de décryptage d'un monde numérique en évolution constante. Il permet d'acquérir des méthodes qui construisent la pensée algorithmique et développe des compétences dans la représentation de l'information et de son traitement, la résolution de problèmes, le contrôle des résultats. Il est également l'occasion de mettre en place des modalités d'enseignement fondées sur la pédagogie de projet, active et collaborative. [...] La maîtrise des langages informatique n'est pas la finalité de l'enseignement, mais leur pratique est le moyen d'acquérir d'autres démarches d'investigation, d'autres modes de résolution de problèmes, de simulation ou de modélisation. »

Compétences développées

Cet enseignement a pour objectif de développer chez les élèves les compétences suivantes :

- **décomposition** : analyser un problème compliqué, le découper en sous-problèmes, en sous-tâches ;
- **reconnaissance de schémas** : reconnaître des schémas, des configurations, des invariants, des répétitions, mettre en évidence des interactions ;
- **généralisation et abstraction** : repérer les enchaînements logiques et les traduire en instructions conditionnelles, traduire les schémas récurrents en boucles, concevoir des méthodes liées à des objets qui traduisent le comportement attendu ;
- **conception d'algorithme** : écrire des solutions modulaires à un problème donné, réutiliser des algorithmes déjà programmés, programmer des instructions déclenchées par des événements, concevoir des algorithmes se déroulant en parallèle.

Les modalités de l'apprentissage correspondant peuvent être variées : travail en mode débranché, c'est-à-dire sans utilisation d'un dispositif informatique, individuel ou en groupe, en salle informatique ou en salle banale, sur tablette ou sur ordinateur.

L'apprentissage associe trois types essentiels de démarche :

- une démarche de **projet** active et collaborative qui suppose l'établissement d'objectifs partagés et la répartition des tâches, la communication entre élèves contributeurs d'un même projet et qui permet l'intervention de plusieurs disciplines ;
- une démarche de **création** : l'enseignement permet la réalisation de productions collectives

(programmes, applications, animations, etc.), au cours desquelles les élèves développent leur autonomie, leur créativité et leur imagination, mais aussi le sens du travail collaboratif ;

- une démarche **interdisciplinaire** qui favorise la mise en œuvre de diverses activités de création numérique, en particulier dans le cadre des enseignements complémentaires.

L'apprentissage de la programmation en Scratch peut se développer sur les trois années du cycle suivant une progression liée à des concepts de programmation plutôt qu'à des éléments de syntaxe du langage :

- la programmation **événementielle** : conception de séquences d'instructions déclenchées par un événement (appui de touche, son reçu par le microphone, motif « touché » par un personnage, etc.) ;
- l'initiation à la programmation **parallèle** : déclenchement par le même événement de deux ou plusieurs séquences d'instructions ;
- le professeur pourra envisager une initiation de ses élèves à certains concepts simples de la **programmation-objet** par la programmation des « lutins » de Scratch, c'est-à-dire les objets tels que les différents personnages, en particulier pour la scénarisation d'une activité, comme il est expliqué plus bas dans « Déroulement de l'exécution d'un programme », mais aussi tout simplement en associant des scripts distincts aux différents lutins d'une même activité, comme il est expliqué plus loin par exemple dans le jeu de Pong.

Le thème « algorithmique et programmation » du programme de mathématiques du cycle 4

Le programme définit comme attendu de fin de cycle : *« Écrire, mettre au point et exécuter un programme simple. »* Il met l'accent sur l'analyse d'un problème en vue de concevoir un algorithme, mais aussi sur la phase de programmation effective, indissociable d'une étape de mise au point. Cependant, il ne faut pas nécessairement insister sur une chronologie précise, ces trois phases avançant souvent en parallèle : la mise au point comprend la phase d'essais-erreurs où l'élève construit petit à petit un programme qui répond au problème qu'il s'est posé. De même, au moment de la programmation effective l'élève peut se retrouver confronté à une question algorithmique qu'il n'avait pas prise en compte dès le départ.

L'attendu de fin de cycle évoquant un programme **simple**, il n'est pas question de formaliser d'aucune manière la distinction algorithme/programme. En particulier, il ne s'agit pas de définir un langage de description d'algorithme (ni pseudo-langage ni organigramme) même si son utilisation peut être envisagée, ponctuellement, mais sans souci de normalisation.

En revanche, il ne faut pas négliger la phase de mise au point : le professeur peut évoquer, si la situation s'y prête, l'utilisation d'un jeu de test. Les élèves sont amenés à comparer leurs solutions algorithmiques et leurs programmes, ce qui les conduit à une première approche, informelle, de la documentation d'un programme (commentaires et spécification[1]). Le partage de programmes entre classes ou entre collèges donne l'occasion de réutiliser en tout ou en partie les programmes écrits par d'autres, illustre la notion de réseau et introduit les questions liées à son usage, qui peuvent utilement être développées en lien avec le professeur-documentaliste.

Approches pédagogiques : éléments de réflexion

La mise en activité des élèves

Une séance d'apprentissage de l'algorithmique et de la programmation ne saurait se dérouler sous forme d'un cours descendant, magistral, où les élèves resteraient passifs. Ainsi, il serait par exemple inefficace de demander à des élèves de reproduire durant un tiers de la séance un programme que le professeur aurait expliqué au tableau durant les deux premiers tiers.

Si chaque séance doit viser des **objectifs de formation clairs et explicites**, par exemple découvrir

1. Spécifier un programme, ou un bloc de programme, c'est décrire avec précision les données attendues, les hypothèses éventuelles sur ces données, et l'action que réalise ce programme sur un tel jeu de données.

l'utilisation des variables, il convient de réserver l'essentiel du temps à une **activité autonome des élèves**.
Une séance peut commencer par quelques minutes où le professeur expose une situation-problème qui introduit la notion visée : par exemple, il propose de reprendre un jeu réalisé dans une séance précédente, en introduisant un score. Il montre comment créer une variable score[2], et comment l'incrémenter. Puis il laisse les élèves modifier leur programme, et peut proposer des défis pour aller plus loin, comme l'introduction de niveaux de difficulté, liés à la réalisation d'un score attestant chaque niveau. La séance peut se terminer par la récapitulation des instructions permettant d'utiliser les variables, éventuellement sous forme d'une fiche.

La pédagogie de projet

L'enseignement de l'algorithmique et de la programmation se prête particulièrement à la pédagogie de projet. L'objectif est de permettre à chaque élève de construire des connaissances et des compétences tout en développant son imagination et sa créativité dans le cadre d'un projet suivi sur quelques séances. Comme l'indique le programme, cet enseignement doit se traduire par la réalisation de productions collectives dans le cadre d'activités de création numérique, au cours de laquelle les élèves développent leur autonomie, mais aussi le sens du travail collaboratif. Pour autant, il ne s'agit bien sûr pas de monter des projets de trop grande envergure, qui se dérouleraient sur une année entière.

Une approche possible est la suivante : au cours d'une première séance, comme il est indiqué dans le paragraphe *« La mise en activité des élèves »*, le professeur propose une activité, dont il a fixé clairement les objectifs de formation et les concepts nouveaux qui devront être installés. Une deuxième séance peut permettre à chaque élève de développer son programme dans les directions qu'il aura lui-même choisies : la séance permet au professeur d'outiller l'élève selon ses besoins pour faire aboutir son projet. Il peut aussi aider l'élève à le redéfinir si la piste sur laquelle il s'est engagé est inadaptée. Une dernière séance peut permettre la finalisation des projets. Le professeur peut aussi proposer une mise en commun des concepts et des techniques utilisés par les uns et les autres.

Une mise en ligne de certains projets, sans distinction de niveau d'expertise, peut être envisagée afin de mettre en valeur les travaux des élèves. On peut également valoriser ces travaux par des ateliers-jeux, des expositions…

Mettre en œuvre la différenciation pédagogique

Les programmes du cycle 4 sont des programmes de cycle, et non pas des programmes annuels. L'organisation du cycle et la conception des enseignements ont pour objectif d'opérationnaliser l'acquisition par chacun des élèves des attendus du socle commun de connaissances, de compétences et de culture : il s'agit d'amener chaque élève à la meilleure maîtrise possible de tous ces attendus, dans le cadre d'un parcours de formation qui prend en compte ses acquis et ses marges de progression.

La différenciation pédagogique permet de réaliser cet objectif et ne saurait se réduire à de la remédiation pour les élèves qui ont plus de besoins. Il s'agit d'accompagner chaque élève, en permettant aux meilleurs de construire des méthodes expertes dans la résolution d'un problème algorithmique, de développer des projets personnels ambitieux, d'assimiler des concepts de programmation riches. Il s'agit en même temps de conduire les élèves les plus en difficulté à une maîtrise suffisante des attendus du programme pour valider l'acquisition du socle commun de connaissances, de compétences et de culture.

C'est ainsi qu'une séance peut comporter un « défi » permettant aux élèves qui auraient rapidement réalisé l'objectif commun de poursuivre en autonomie leur travail, pendant que le professeur se rend disponible auprès d'autres élèves.

2. Il suffit pour cela de cliquer sur le bouton « Créer une variable » dans la catégorie Données.

Par exemple, dans le cadre du *« Parcours d'un labyrinthe »* de la partie *« Exemples de premières séances avec les élèves »*, l'objectif commun peut être dépassé par certains élèves qui peuvent utiliser un bloc de répétition, ou analyser le fonctionnement des scripts cachés par le professeur, comme le bloc-utilisateur « départ ». On propose d'autres stratégies de différenciation pédagogique dans d'autres exemples de ce document.
De même, comme on l'a dit au paragraphe « La pédagogie de projet », le professeur guide l'élève dans la définition de son projet, en respectant ses choix, mais en les adaptant si besoin au degré de maîtrise effectivement atteint, que ce soit pour l'enrichir ou en réduire les objectifs.

La boucle essai-erreur en informatique

L'apprentissage de l'algorithmique et de la programmation bénéficie d'une boucle essai-erreur qui est très courte : en cliquant sur un bloc ou un script on peut observer immédiatement l'effet sur les lutins, les tracés du stylo… D'autre part, aucun jugement de valeur n'est porté sur l'élève ou ses performances, la machine se contente d'effectuer les commandes et c'est l'élève lui-même qui constate s'il s'est trompé ou pas, et qui est invité à essayer une autre solution en cas d'échec. De plus, l'obtention d'une solution ne signifie pas la fin du processus de création, qui peut se poursuivre, pour en améliorer la qualité ou la performance.

Pour le professeur, cela signifie qu'il accepte pour ses élèves la nécessité de phases de tâtonnement et d'essais-erreurs, durant lesquelles il intervient le moins possible. En particulier, il est recommandé de résister à l'envie de taper sur le clavier ou de manipuler la souris à la place de l'élève, et de se contenter de répondre oralement aux questions de l'élève, de lui rappeler éventuellement tel ou tel script déjà écrit qui pourrait l'inspirer ou encore de lui suggérer de nouvelles pistes pour résoudre la difficulté qu'il rencontre, mais sans intervenir directement sur la machine de l'élève.

La préparation d'une séance

Nous proposons ci-après quelques exemples de séquences. Il convient, comme on l'a déjà dit, de définir par avance les objectifs de formation correspondants et de prévoir une chronologie en trois phases : très courte introduction de présentation d'un concept ou d'un problème, activité autonome des élèves comportant éventuellement des défis ou d'autres éléments de différenciation, et, le cas échéant, phase finale d'institutionnalisation des nouveaux concepts, de mise au propre et par écrit des notions rencontrées.

Pour favoriser le travail autonome des élèves, il est possible de préparer la séance en proposant aux élèves, via le réseau ou une clé USB, une « trame » de programme, qu'ils seront amenés à compléter. Cette trame doit être limitée à ce que les élèves n'auraient pas le temps de préparer, ou aux objets qu'ils ne sont pas encore en mesure de programmer eux-mêmes.
Il serait par exemple malavisé de faire travailler les élèves durant toute une séance à la création d'un arrière-plan : le professeur peut demander que ce travail soit effectué à la maison, ou proposer un arrière-plan qu'il met à disposition des élèves en début de séance[3].

On verra plus loin tout l'intérêt qu'il peut y avoir que les élèves disposent d'un bloc utilisateur « reset », créé par le professeur, qui permette de réinitialiser systématiquement la configuration avant un travail sur le déplacement et les tracés dont on prévoit qu'il nécessitera de nombreuses tentatives avant d'aboutir.

En revanche, il n'est pas souhaitable de systématiquement proposer aux élèves un programme incomplet qu'ils seraient chargés de compléter, d'améliorer ou de corriger. En effet, la phase de mise au point qui accompagne toujours l'écriture d'un programme développe précisément ces

3. Les observations déjà conduites en école et au collège montrent que les élèves s'amusent volontiers à dessiner leurs lutins, les arrière-plans, ou à jouer avec les différents sons. Mais le risque est de limiter une séance à ces seules activités, sans plus aucun objectif de formation. En revanche, il peut être intéressant de travailler avec un professeur d'arts plastiques sur la création d'arrière-plans, avec un professeur d'histoire sur la création d'un lutin figurant un personnage historique, etc.

compétences, et il ne faut pas perdre de vue l'objectif d'autonomie des élèves. Ceux-ci doivent être amenés, le plus souvent possible, à écrire eux-mêmes leurs propres programmes en développant leur inventivité, leur imagination et leur créativité.

Considérations didactiques : premiers éléments

Le lecteur est invité à installer Scratch sur sa machine personnelle[4], et à tester au fur et à mesure de sa lecture les activités proposées.

Le vocabulaire de scratch

Comme en mathématiques, le travail sur la compétence Communiquer, sur la construction d'une expression écrite et orale précise est un objectif de formation important.
Il convient donc de s'accorder sur un vocabulaire précis. Dans l'environnement Scratch, on évoquera en particulier les points suivants.

- les lutins sont les objets manipulés par Scratch. Par défaut, il s'agit du chat, mais on peut utiliser plusieurs lutins, qui peuvent interagir en échangeant des messages, en détectant des collisions, etc. Chaque lutin a ses propres scripts. Tout lutin est muni d'un stylo, inactif (car relevé en position haute) par défaut, qui se déplace avec le lutin et qui, s'il est en position d'écriture, laisse une trace lors de ses déplacements.
- Un lutin peut porter différents costumes, ce qui permet par exemple d'animer les déplacements d'un lutin ou encore de prévoir un dé avec 6 costumes, un par face.
- On pourra par exemple consulter le tutoriel « Comment simuler un lancer de dés dans mon projet Scratch ? » du site MagicMakers ;
- les *blocs ovales* représentent des expressions, c'est-à-dire des valeurs, sur lesquelles on peut effectuer des calculs ;

- les *blocs pointus* représentent également des expressions, mais à valeurs booléennes, c'est-à-dire vrai ou faux, et sont utilisés pour réaliser des expressions conditionnelles, ou des tests ;

4. Il peut pour cela s'aider de l'annexe qui figure à la fin de ce document.

- les *blocs aimantés* représentent des instructions : en les accolant les uns aux autres on construit des blocs de script, souvent initiés par un bloc chapeau associé à un événement. À chaque lutin peuvent être associés différents scripts. C'est le principe de construction des programmes en Scratch ;

- les scripts sont construits en accolant des blocs aimantés. Chaque lutin peut être muni de plusieurs scripts.

La notion de variable

La notion de variable en informatique diffère de la notion mathématique.

En mathématiques, la variable apparaît dans des formules comme celle du périmètre du cercle $2\pi R$ ou dans l'expression symbolique des fonctions, comme dans $y = f(x)$. On distingue : les indéterminées, comme dans une identité remarquable $(a+b)^2 = a^2+2ab+b^2$, qui indique que l'égalité est vraie pour toutes les valeurs données à a et à b ; les inconnues, comme dans l'équation $2x+3 = 4x-7$, où cette fois il s'agit de déterminer pour quelles valeurs de la variable x l'égalité est vraie ; les paramètres, qui conservent une valeur fixe, mais de portée générale, comme a qui désigne le coefficient de la fonction linéaire $x \mapsto ax$.

Dans de nombreux langages informatiques, on est amené à écrire des instructions comme $x \leftarrow x+1$, souvent même sous la forme $x = x+1$, qui sont d'une nature totalement différente : il ne s'agit pas là d'une égalité, ni d'une équation, mais **d'une instruction d'affectation**, qui va modifier le contenu de la variable x et qui devrait être lue « x reçoit la valeur $x+1$ », et surtout pas « x égale $x+1$ ».

Il existe une façon à la fois claire et rigoureuse de présenter les variables informatiques.
On dira qu'une variable x est une **étiquette** collée sur une **boîte** qui peut **contenir** différentes valeurs. Il est important de faire comprendre que le contenu de chaque boîte varie au cours de l'exécution d'un programme, ce qui n'est pas le cas de la variable mathématique (la variable x pour une fonction ne varie pas au cours du calcul, ou encore la lettre x associée à une grandeur dans un problème reste attachée à cette grandeur pendant les calculs, cf la ressource thématique *Comprendre et utiliser la notion de fonction*).

Quand l'ordinateur évalue une expression dans laquelle figure une variable, comme

l'expression x + 4, le résultat de l'évaluation est la somme du nombre contenu dans la boîte étiquetée par x et de 4.

Une **instruction d'affectation** comme y = x + 4 signifie donc :

- évaluer l'expression x + 4, en ajoutant 4 à la valeur contenue dans la boîte étiquetée par x ;
- jeter le contenu de la boîte étiquetée par y, et le remplacer par la valeur calculée à l'étape précédente.

L'instruction x = x + 1 s'explique alors exactement de la même façon.

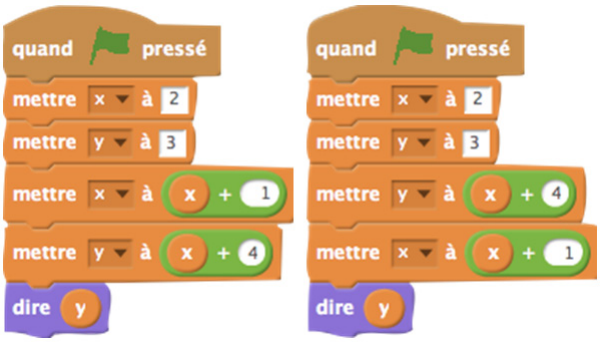
En Scratch, le risque de confusion est moindre puisqu'on dispose de blocs ovales comme 

pour les expressions, et l'instruction d'affectation est le bloc aimanté ci-dessous, qui ne ressemble en rien à une équation mathématique, ni à une égalité[5].

Pour autant, il est important d'expliquer correctement la notion de variable, en illustrant le propos par exemple avec des boîtes dessinées au tableau. On fait évoluer le contenu en simulant l'exécution d'un programme. On fait ainsi explicitement la différence avec l'utilisation des variables en mathématiques.

Déroulement de l'exécution d'un programme

Quand on accole différents blocs aimantés, leur ordre est tout à fait essentiel. Ainsi, si l'on s'intéresse aux deux scripts suivants, en s'appuyant sur l'interprétation des variables que nous venons de développer, on observe des effets totalement différents :



Le script de gauche fera dire « 7 » au chat, le script de droite lui fera dire « 6 ». Pour expliquer le fonctionnement d'un programme, il est donc nécessaire de parler d'états ou de configurations successifs, la description d'une configuration précisant en particulier les étiquettes et les contenus de chaque boîte de mémoire. Ainsi est introduite une notion de temporalité, par la suite des états (ou configurations) successifs.

Il n'est évidemment pas question, au niveau du collège, d'entrer dans des détails trop précis sur le temps d'exécution de tel ou tel bloc, de tel ou tel script. Néanmoins, la notion d'états (ou configurations) successifs peut être introduite, pour expliquer le comportement d'un programme.

Si les deux scripts ci-dessus sont exécutés en même temps, que se passe-t-il ? On introduit ici la notion de parallélisme, c'est-à-dire d'exécution simultanée de deux scripts différents. Le terme *« parallélisme »* n'est d'ailleurs pas une connaissance attendue des élèves, c'est l'utilisation de

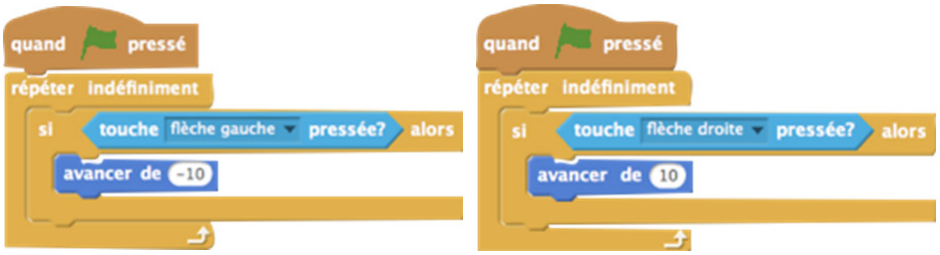
5. Il est sans doute préférable de ne pas recourir au bloc aimanté « ajouter à y 1 » dont la traduction n'est pas heureuse (« ajouter 1 à y » aurait assurément été plus clair).

scripts parallèles qui figure seule parmi les objectifs de formation. Bien sûr, tout développement théorique sur le parallélisme est à proscrire.

En l'occurrence, le résultat est assez imprévisible, car les deux scripts devraient se dérouler à peu près à la même vitesse, et on ne peut deviner ce qu'annoncera le chat…

Il est sans doute préférable de n'évoquer le parallélisme que dans des situations où les différents scripts amenés à se dérouler en parallèle sont déclenchés par des événements différents.

Par exemple :



On pourrait bien sûr n'écrire qu'un script, la boucle « répéter indéfiniment » comprenant les deux conditionnelles de façon consécutive : ce serait cependant un peu cacher le fonctionnement de l'algorithme, en compliquer la lecture et la compréhension. C'est ainsi un exemple typique d'utilisation pertinente de la notion de parallélisme.

Une activité prévoyant un scénario particulier, par exemple un dialogue entre plusieurs personnages, posera la question des « rendez-vous » : le personnage A commence, puis passe la main au personnage B qui réalise d'autres actions. Ceci peut être programmé à l'aide d'un message envoyé par A à B, qui lui indique que c'est à son tour d'agir ; pour ce faire, Scratch, dans la catégorie « Événements » propose deux blocs 

et 

. Cette utilisation, très modeste, de la programmation-objet permet un travail intéressant de scénarisation d'une activité.

Les déplacements et les dessins au stylo

Les lutins de Scratch sont tous munis d'un stylo[6]. Le stylo se déplace avec le lutin, mais il faut le mettre en position d'écriture pour que le tracé s'effectue lors des déplacements du lutin, ou le relever en position haute pour déplacer le lutin sans rien dessiner.

Il y a deux façons de se déplacer :

- en déplacement relatif. Les commandes 

et 

, 

déplacent le lutin dans la direction qu'il regarde, depuis la position courante, ou le font tourner, par rapport à la direction courante ;
- en déplacement absolu. Les commandes

et

indiquent la nouvelle position ou la nouvelle direction du lutin.

Il est utile de faire le lien entre ces commandes et les notions mathématiques correspondantes, mais on peut également rapprocher la différence relatif/absolu des modes d'adressage d'une cellule dans un tableur.

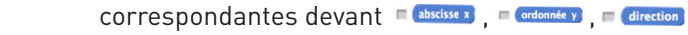
Les mouvements relatifs correspondent bien à des instructions de déplacement données à un être humain : elles peuvent être simulées physiquement, par exemple sur un quadrillage dessiné dans la cour du collège.

6. La position précise du stylo par rapport au lutin est au centre de celui-ci : en éditant le costume et en cliquant sur l'icône en forme de croix  tout en haut à droite, on voit apparaître avec précision cet emplacement sur le dessin du lutin.

Il est sans doute utile, dans les premiers temps, de laisser apparent le lutin associé au stylo afin de comprendre la notion de direction courante : c'est la direction du regard du chat. Plus tard, on pourra cacher le lutin avant de dessiner.

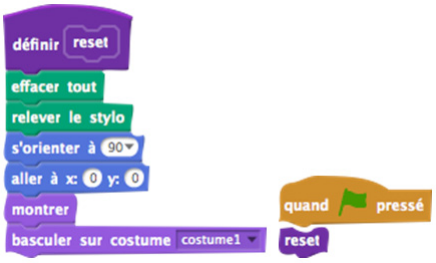
Les mouvements absolus permettent de travailler plus directement les notions de géométrie du programme.

Il convient de préciser les conventions de repérage de Scratch : l'origine est au centre de l'écran de travail, ses bords gauche et droit sont d'abscisses -240 et +240, les bords bas et haut d'ordonnées -180 et +180. Les angles sont mesurés dans le sens des aiguilles d'une montre, la direction 0° visant le haut de l'écran, la direction 90° la droite, etc.

On peut faire afficher les coordonnées absolues d'un lutin en cochant les cases correspondantes devant 

(dans la catégorie « Mouvement »).
A contrario, les coordonnées affichées sous le coin droit de la zone de dessin sont celles de la souris…

Une difficulté se rencontre fréquemment lors de la mise au point d'un programme de dessin : quand on clique sur le drapeau vert pour lancer un script, aucune initialisation particulière n'est effectuée par Scratch[7], qui reprend le lutin dans l'état courant (position et direction). Ce peut être une bonne pratique que le professeur crée un nouveau bloc-utilisateur « reset » que les élèves pourront systématiquement insérer en tête de leurs scripts, même sans entrer dans les détails de la conception d'un tel bloc lors de ses premières utilisations.



Blocs crées par les utilisateurs

La création d'un bloc-utilisateur permet d'aborder la notion de typage des arguments, puisque Scratch autorise des blocs à plusieurs arguments, chacun d'entre eux étant d'un type bien défini, à choisir parmi nombre, chaîne de caractères ou booléen. Une attention particulière sera portée au nommage d'un bloc-utilisateur et de ses arguments, Scratch permettant d'introduire du texte entre plusieurs arguments. Par exemple, on pourra choisir comme entête d'un bloc-utilisateur :

La création d'un bloc-utilisateur peut être envisagée dans le cas de la réutilisation fréquente d'un script. Elle permet donc également d'aborder les notions de réutilisation du code, de partage de programme et de spécification : pour partager un programme, il est important de le documenter. Cela passe par un nommage pertinent et parlant et le typage explicite des arguments, comme on l'a déjà dit, mais aussi par la description (en français) de l'action réalisée par le bloc-utilisateur et la déclaration des hypothèses qu'on fait éventuellement sur les arguments. Là encore, l'objectif n'est pas de formaliser des protocoles de documentation,

7. Il y a une exception : le chronomètre est remis à zéro à chaque clic sur le drapeau.

mais de sensibiliser à son importance, à l'occasion d'un partage effectif de « morceaux de code » entre élèves.

Malheureusement, Scratch n'offre pas la possibilité d'écrire des fonctions, c'est-à-dire des blocs-utilisateurs qui renvoient un résultat. Les blocs-utilisateurs de Scratch opèrent par « effet de bord », c'est-à-dire en modifiant l'état de la mémoire, les objets ou la fenêtre de tracé : on peut modifier la valeur d'une variable, déplacer les lutins ou les faire changer de costume, dessiner à l'écran, etc. Mais on ne peut renvoyer un résultat, par exemple numérique, qui serait réutilisé dans une expression. Dans la philosophie de Scratch, on a créé un nouveau bloc, à l'image des blocs usuels, qu'on peut empiler dans un script complexe, mais dont le contour comporte les encoches aimantées habituelles, ce qui montre bien qu'il ne peut s'agir d'une expression.

Exemples de premieres séances avec les élèves

Parcours d'un labyrinthe

Le lecteur est invité à ouvrir et expérimenter à partir du document joint PremierLabyrinthe.sb2 en parallèle de sa lecture.

Cet exemple de première séance a comme objectif principal la prise en main par les élèves du logiciel.

Il s'agit :
- de commencer à utiliser le vocabulaire de Scratch pour savoir nommer les objets de l'interface ;
- de comprendre le fonctionnement des blocs, leur « aimantation », leur duplication par un clic droit dans la fenêtre de script ;
- de savoir exécuter et mettre au point un script, de déclencher son exécution par un clic sur le drapeau vert ;
- d'enregistrer son programme.

On ne cherche surtout pas ici à découvrir l'intégralité des commandes disponibles !

On peut par exemple proposer d'ouvrir un fichier préparé par le professeur, avec un arrière-plan déjà dessiné, un bloc utilisateur « départ », un script détectant le succès.

L'arrière-plan peut être celui-ci[8] :
Le bloc « départ » permet de positionner le « chat » à sa bonne place et à la bonne taille.
Le défi proposé aux élèves est de combiner des déplacements (« avancer de » et « tourner ») pour amener le chat à la cible jaune.

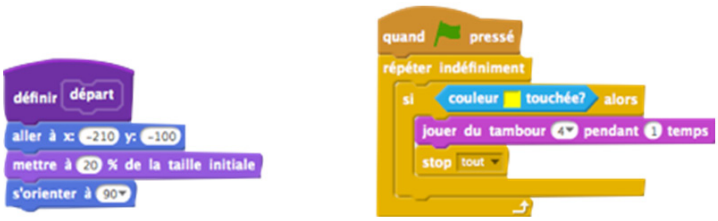
8. Spécifier un programme, ou un bloc de programme, c'est décrire avec précision les données attendues, les hypothèses éventuelles sur ces données, et l'action que réalise ce programme sur un tel jeu de données.

On peut présenter dans la fenêtre de script le squelette du script visé : déclenchement par le drapeau vert, utilisation du bloc départ, et les trois blocs aimantés qu'il faut dupliquer et « attacher » pour obtenir le script final.

En ayant déjà placé un bloc « avancer de 100 », l'élève peut tester que le chat arrive au bout du premier trajet horizontal, ce qui lui donne une première idée de l'échelle en pixels.

Il est d'ailleurs possible de proposer à l'élève, pour l'aider, d'afficher les coordonnées du chat en cochant les cases devant abscisse x et ordonnée y de la catégorie « Mouvement ».

Le professeur a préparé ces éléments de script, qu'il n'est pas nécessaire de présenter aux élèves lors d'une première séance :



Aux élèves qui réussiraient très vite à terminer l'exercice, on peut expliquer le fonctionnement de ces deux blocs. Une autre idée consiste à simplifier leur script à l'aide d'un bloc de répétition :

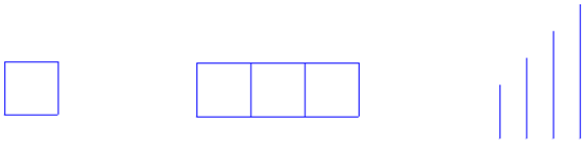
Cette façon de procéder, donner un même objectif à tous et proposer selon l'avancement de chacun des compléments de recherche, peut être utilisée régulièrement, tout au long du cycle.

Programmation d'un dessin a l'écran

Le lecteur est invité à ouvrir et expérimenter à partir du document joint PremierDessins.sb2 en parallèle de sa lecture.

Il s'agit là aussi d'une activité qui peut être menée dès le début du cycle, et qui a pour objectifs de faire découvrir les commandes de déplacement et de tracé.

On demande de réaliser des tracés très simples, comme, par exemple :

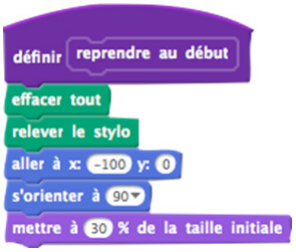


Comme il est indiqué dans le paragraphe « Les déplacements et les dessins au stylo », on peut commencer par une activité débranchée, où les élèves peuvent se répartir par paire : un élève écrit, à la main, un jeu d'instructions avancer, tourner, relever le stylo ou le descendre en position d'écriture, l'autre exécute en respectant scrupuleusement le « programme » de son camarade. On échange les rôles pour d'autres dessins.

On passe ensuite à la programmation en Scratch, le professeur montre rapidement quels sont les quelques blocs aimantés à utiliser (trois pour les mouvements, deux pour le stylo, et le bloc effacer tout), et propose de reprendre les tracés précédents, et éventuellement de nouvelles figures simples.

Il n'est pas ici besoin de travailler longuement sur les unités : on peut par exemple proposer que 1 carreau = 50 pixels, sans préciser une taille en millimètres. De même, l'objectif n'est pas que les élèves utilisent nécessairement un bloc de répétition avant qu'ils en ressentent eux-mêmes l'utilité : la différenciation pédagogique joue ici pleinement son rôle, et le professeur propose ces blocs aux seuls élèves qui en font la demande, et ne procède à leur institutionnalisation que dans une séance ultérieure.

Le professeur peut avoir préparé un bloc utilisateur « reprendre au début », pour effacer l'écran, positionner le lutin à un endroit de départ précis, dans une orientation précise.

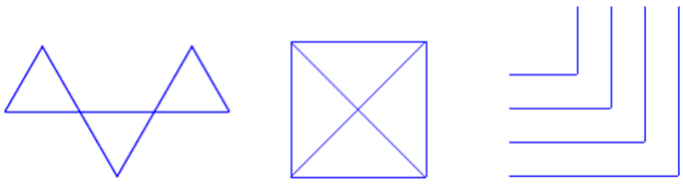


Afin d'aider les élèves à gérer les rotations, il est probablement préférable dans un premier temps de laisser le chat apparent, la direction de son regard pouvant aider à la mise au point des programmes.

Chacune des figures proposées ci-dessus peut être obtenue de nombreuses façons différentes. Par exemple, pour la troisième, on pourra simplement utiliser le script ci-dessous.

Des élèves peuvent chercher à utiliser des répétitions pour simplifier leur script, se confrontant à une difficulté : la taille variable des 4 bâtons à dessiner.

On peut dans une séance suivante, introduire la notion de variable et proposer ainsi de reprendre la figure des 4 bâtons, et en proposer d'autres, comme les suivantes : là encore, chaque élève doit pouvoir avancer à son rythme, et ne devrait pas être contraint à utiliser trop tôt des procédures expertes avant d'avoir maîtrisé les commandes élémentaires de mouvement et de tracé.



Il ne faut pas exclure que des élèves souhaitent eux-mêmes expérimenter sur des figures qu'ils auront choisies sans l'intervention du professeur. On veillera toutefois à éviter deux écueils : le choix de figures trop complexes, et une situation où un élève s'amuserait à jouer avec la couleur, l'épaisseur du trait, ou d'autres détails « cosmétiques » alors que l'objectif est de mettre en place des éléments de programmation et des compétences de repérage dans le plan[9].

Quelques exemples d'activités

Ces exemples d'activités ne prétendent pas recouvrir exhaustivement le programme.
Ils ne sont en rien prescriptifs, mais fournissent simplement des pistes de travail qui pourront inspirer les professeurs.

Un jeu à trois personnages

Le lecteur est invité à ouvrir et expérimenter à partir du document joint PoissonsChatChiens.sb2 en parallèle de sa lecture.

Cette activité se prête facilement à un découpage, en ajoutant au fur et à mesure un personnage et des interactions, ce qui permet une pédagogie différenciée.

Elle permet de revoir la gestion des déplacements et introduit l'utilisation conjointe de plusieurs personnages, ainsi que leur interaction. On rencontre également naturellement la notion de variable, et l'utilisation du chronomètre intégré à Scratch.

On peut commencer avec le seul lutin habituel (le chat). On demande d'abord aux élèves de gérer son déplacement à l'aide des quatre flèches du clavier, ce qui relève de la programmation événementielle.

On ajoute un nouveau lutin, un chien, qui doit effecteur indéfiniment des allers-retours du bord gauche au bord droit de l'écran, ce qui est facilement réalisé à l'aide d'une répétition infinie et des blocs  et  . Un défi supplémentaire, pour les élèves les plus à l'aise, consiste à demander de programmer un parcours aléatoire du chien. Pour l'intérêt du jeu, il convient de prévoir un déplacement pas trop rapide.

On souhaite alors que chaque lutin réagisse au moment où il rencontre l'autre : le chien pourra simplement dire bonjour et le chat exprimer sa frayeur. La « rencontre » est détectée par le capteur  et il faudra la tester au sein d'une répétition infinie. Il est probable que la nécessité de cette boucle n'apparaisse pas immédiatement aux élèves et que certains se contentent d'effectuer le test une seule fois. Il peut être intéressant de comparer la gestion directe d'un événement comme l'appui d'une touche avec celle du contact détecté par la valeur d'un capteur. La distinction entre les scripts de chaque lutin est également un élément qui mérite d'être souligné.

On peut d'ores et déjà ajouter une variable globale qui permet de gérer un score, chaque rencontre du chat et du chien faisant baisser le score.

On propose alors d'ajouter un troisième personnage, un poisson. Il s'agit de programmer un déplacement aléatoire dans le plan de ce lutin.

Une fois cette étape franchie, on gère l'interaction chat-poisson : à chaque contact, on incrémente le score, et on souhaite faire disparaître le poisson (le chat l'a mangé) et le faire réapparaître à un endroit aléatoire après un délai fixé.

Il s'agit donc de simuler le fait que le chat mange les poissons qu'il réussit à attraper et qu'à chaque fois un nouveau poisson apparaît, même si c'est un lutin unique qui gère les poissons.
La difficulté tient au délai fixé. Cela permet d'introduire le chronomètre de Scratch, mais également une variable qui conservera l'heure où le poisson a été attrapé : le script du poisson étant le seul qui ait besoin de cette variable, on la définira comme variable privée du lutin, à la différence du score que plusieurs lutins ont besoin de modifier ou de consulter, qui doit

9. Le fichier joint *PremiersDessins.sb2* montre un exemple d'activité. L'utilisation d'un message a permis de regrouper les tracés de deux figures différentes dans le même script : ce n'est évidemment pas recommandé pour des premières séances.

donc être une variable publique (ou globale). Cette distinction est une première introduction aux concepts de la programmation-objet qui n'a évidemment pas à être formalisée.

La commande qui permet d'attendre le délai fixé (ici à 2 secondes) ne va pas de soi et mérite sans doute un moment collectif de réflexion et d'explication :

Une dernière étape consiste à ajouter un quatrième lutin, l'arbitre, qui apparaît quand le score devient négatif (après des rencontres du chat et du chien) ou dépasse une valeur fixée (le chat a réussi à manger suffisamment de poissons). Là encore, on met en évidence la nécessité d'une boucle infinie pour scruter la valeur d'une variable globale, le score, et il peut être utile de réfléchir au nombre de scripts qui se déroulent en parallèle à un moment donné.

On peut imaginer différents prolongements : par exemple, un étang où nage le poisson, le chat ne pouvant l'attraper, et le manger, qu'au bord de l'étang. Ou encore la présence de plusieurs poissons…

Un exemple de production est proposé dans le fichier joint PoissonChatChien.sb2.

Un jeu de pong

Le lecteur est invité à ouvrir et expérimenter à partir du document joint Pong.sb2 en parallèle de sa lecture.

Le professeur peut choisir ce type d'activité avec différents objectifs : travailler la notion de direction angulaire relative ou absolue, découvrir et utiliser les constructions conditionnelles, découvrir et utiliser des variables, programmer une gestion d'événements.

Selon le(s) choix qu'il effectue, il développera plus tel ou tel aspect de l'activité, proposera des défis différents aux élèves, prolongera l'activité suivant diverses directions.

On peut commencer par une activité concernant le rebond sur les bords d'une balle. Il vaut mieux choisir pour commencer un lutin dont la direction est immédiatement visible (le chat, une flèche), et reporter à plus tard l'utilisation d'un lutin en forme de balle. De la même façon, il est sans doute utile d'afficher la direction du lutin (en cochant la case appropriée tout en bas de la catégorie « Mouvement »).

Une répétition infinie de 

, 

, permet alors l'observation de ce qui se passe aux bords : un objectif est que les élèves conjecturent les règles de changement de direction.

On peut alors proposer un défi, qui est de réaliser le même processus, mais sans utiliser le bloc 

. Il faut toutefois prendre garde à repérer l'approche des bords, en ajoutant des répétitions infinies de test sur l'abscisse (inférieure à -215 ou supérieure à 215) et sur l'ordonnée. Ce défi n'est pas si simple pour des élèves inexpérimentés.

On ajoute deux raquettes (que les élèves peuvent dessiner, un simple rectangle suffit). On pensera à les nommer de façon explicite, raquette gauche et raquette droite, par exemple. Il n'est pas très difficile d'associer à chaque raquette un script qui permet à l'utilisateur de la faire monter ou descendre, à l'appui d'une touche ou d'une autre. Il faudra donc choisir quatre touches différentes du clavier, pour le joueur gauche et le joueur droit.

Il s'agit là d'une programmation événementielle très simple à mettre en œuvre.

Pour l'instant, la balle et les raquettes agissent sans aucune interaction.

Il convient maintenant de gérer le rebond de la balle sur chaque raquette.

Pour cela on crée une boucle infinie qui teste le contact à l'aide du premier bloc de la catégorie « Capteurs ». Si la raquette gauche est touchée, il suffit de changer la direction de la balle en l'opposé de la direction actuelle. Remarquons que puisqu'il s'agit de modifier la direction de la

balle, c'est bien sur le script de la balle qu'il faut travailler, et non sur celui de la raquette !

C'est un objectif important que les élèves comprennent bien quelles commandes sont à écrire dans le script de la balle et quelles autres dans celui de chaque raquette : on introduit ici des concepts liés à la programmation-objet, sans avoir pour autant besoin du moindre formalisme.

Une autre remarque s'impose ici : on peut bien sûr créer un unique script long et compliqué pour la balle, mais il est sans doute plus simple de créer plusieurs scripts associés chacun à un comportement particulier : le déplacement perpétuel sur le terrain de jeu, la gestion du rebond sur les bords (si on ne veut pas utiliser la commande 

), le contact avec la raquette gauche, le contact avec la raquette droite. Cela introduit à la fois la notion de programmation d'actions en parallèle et permet une mise au point beaucoup plus facile des programmes.

Plusieurs prolongements sont envisageables.

En premier lieu, on peut ajouter un score pour chacun des deux joueurs, en créant deux variables score gauche et score droit (toujours utiliser des identifiants explicites et non abrégés), qui sont incrémentées à chaque rebond réussi sur la raquette correspondante.

Les élèves peuvent décider si la balle rebondit sur les côtés gauche et droit au cas où les raquettes l'auraient ratée, ou si dans ce cas la partie continue avec la balle au centre et une direction tirée au hasard. On peut également ajouter des pénalités au score adéquat dans ce cas de balle ratée.

Un prolongement plus ardu consisterait à gérer des niveaux de difficulté, à chaque fois qu'un score dépasse une dizaine (par exemple) : ce pourra correspondre à une accélération de la balle, à des raquettes moins longues, etc.

Un exemple de production est proposé dans le fichier joint Pong.sb2.

Un jeu avec des pièces

Le lecteur est invité à ouvrir et expérimenter à partir du document joint Francs.sb2 en parallèle de sa lecture.

L'activité est reprise d'un exercice du rallye mathématique de l'académie de Lyon.

On dispose de quatre pièces, disposées initialement respectivement du côté face, pile, face, pile.

Chaque clic sur une pièce la retourne, ainsi que son éventuelle voisine de droite.

Le joueur gagne quand les quatre pièces sont du même côté.

Bien entendu, on peut varier le nombre de pièces, ainsi que la disposition initiale — mais attention, certaines dispositions ne sont pas résolubles !

L'objectif de cette activité est de travailler les notions suivantes :
- instructions conditionnelles ;
- utilisation d'une variable ;
- utilisation simple d'une liste ;
- gestion d'événements déclenchés par la souris ;
- déclenchement de plusieurs actions en parallèle ;
- échange de messages entre objets.

Chaque pièce est représentée par un lutin, avec deux costumes, pour le côté pile et le côté face.

On propose, dans le fichier joint Francs.sb2 , un exemple de production autour de cette activité.

Dans cet exemple, on a également créé deux costumes pour l'arrière-plan, avec un costume pour signaler la victoire et un autre pour préciser qu'on aurait pu gagner en moins de clics. En effet, avec la configuration initiale choisie, il existe des solutions en seulement 2 clics.

Une difficulté de l'activité tient à ce qu'un clic sur une pièce doit entraîner une action sur sa voisine : c'est une occasion naturelle d'introduire la notion de message entre lutins. Les actions déclenchées par un clic sur la pièce et la réception d'un message sont très similaires, mais différentes : c'est là aussi une difficulté.

Dans un objectif de différenciation pédagogique, on peut apporter de l'aide en fournissant le script de la pièce 3, par exemple, et demander d'écrire les scripts des pièces 1, 2 et 4, ce qui demande une bonne analyse des actions, et correspond bien aux compétences « mettre au point un programme simple » et « reconnaître des schémas ».

L'initialisation de la variable qui compte le nombre de clics et de la liste qui retient les faces courantes des pièces est réalisée, à l'appui du drapeau vert, par un script lié à l'arrière-plan. Cette méthode permet de distinguer ce qui relève du comportement global de l'application et ce qui est spécifique de chaque pièce. On notera qu'on a choisi également de tester la victoire dans le même script, lié à l'arrière-plan.

L'utilisation d'une liste paraît indispensable, car chaque lutin est le seul à avoir accès au numéro de son costume courant : aucun lutin ne peut donc tester l'égalité des côtés des quatre pièces en utilisant seulement le bloc 

.

Cette activité ne demande qu'une utilisation très élémentaire d'une liste, et peut donc servir d'introduction à cette notion.

Le test de victoire peut être réalisé de plusieurs façons : par une imbrication de plusieurs conditionnelles, comme dans la production jointe, mais aussi en utilisant les connecteurs logiques. Ce peut être intéressant de ne pas guider les élèves sur telle ou telle voie, mais de les laisser choisir. On peut s'attendre à ce que les élèves cherchent un bloc pour tester des égalités simultanées, et expliquer qu'on ne peut tester que l'égalité de deux quantités.
On a choisi de représenter les valeurs pile et face par les nombres 1 et 0, ce qui permet d'utiliser

qui devra naturellement faire l'objet d'un travail spécifique : une fois qu'on aura trouvé comment modifier un élément d'une liste, il faudra encore trouver comment une seule expression arithmétique simple permet de passer de pile à face et vice versa.

L'utilisation d'images trouvées sur Internet pour les lutins figurant les pièces sera l'occasion d'évoquer la question de licence liée à l'utilisation de ces images : sont-elles du domaine public ? Est-ce le cas pour les pièces de 1 €, par exemple ?

Un jeu de tic-tac-toe

Le lecteur est invité à ouvrir et expérimenter à partir du document joint TicTacToe.sb2 en parallèle de sa lecture.

Le lecteur est invité à expérimenter à partir du fichier joint TicTacToe.sb2 , et à regarder en détail les scripts proposés en parallèle à sa lecture. Cette activité est plus ambitieuse, et le fichier joint n'est qu'un exemple de production qui demande un niveau de maîtrise important. Mais on pourra s'en inspirer, car elle se découpe aisément en sous-problèmes dont la résolution permet d'approcher de nombreuses notions différentes.

On se propose de faire jouer tour à tour le joueur (l'utilisateur du programme) et la machine. C'est le joueur qui joue le premier. Les marques du joueur sont des croix vertes, celles de la machine des disques rouges. Ainsi chacune des 9 cases du jeu peut-elle se retrouver dans 3 états différents.

Chaque case est simulée par un lutin, qui porte 3 costumes : un carré blanc, une croix ou un disque. Jouer revient donc à changer le costume d'une case blanche. On introduit ainsi le concept d'attribut d'un objet : il s'agit ici du costume. Ces attributs sont privés : seul un lutin peut savoir quel est son costume actuel. Or l'arbitre, pour décider de la victoire du joueur ou de la machine, doit connaître l'état de chaque case du jeu.

C'est pourquoi on introduit une liste des valeurs des différentes cases : cette liste étant une variable globale, les lutins pourront mettre à jour la valeur qui leur correspond, et l'arbitre examiner les valeurs de toutes les cases qui l'intéressent. On peut commencer en choisissant les valeurs 0, 1, 2 pour coder les trois états. La victoire est déclarée lors de l'alignement de 3 croix ou de 3 disques, et se teste donc en faisant par exemple la somme des trois valeurs

correspondantes 1+1+1 = 3 ou 2+2+2 = 6. En jouant on s'aperçoit cependant que l'arbitre rend des décisions inexactes. Il est sans doute intéressant de laisser les élèves trouver sans aide du professeur l'erreur commise : en effet, 0+1+2 = 3 donc l'alignement blanc / croix / disque provoque, à tort, la victoire. Une façon simple de corriger le programme consiste à choisir les valeurs 0, 1 et 4 pour coder les trois états : il suffit de vérifier que la seule décomposition de 3, respectivement de 12, comme somme de trois de ces valeurs est 3 = 1+1+1, respectivement 12 = 4+4+4.

Le joueur joue en cliquant une case blanche, chaque lutin a donc un petit script qui gère l'événement clic de souris. Il faut tester, par une conditionnelle, que la case cliquée était blanche, ce que le lutin peut faire en examinant son costume actuel (lui en a bien le droit) ou en consultant la liste globale des valeurs des cases. Après avoir joué, on passe la main à l'arbitre, pour qu'il puisse éventuellement constater une victoire, en lui envoyant un message, puis on cède la main à la machine, par un nouveau message « À la machine ! » que reçoit en fait l'arbitre.

L'arbitre choisit alors au hasard une case jusqu'à trouver une blanche (avec une boucle répéter tant que) qu'il conserve dans une variable globale case avant d'envoyer à son tour un message « coup machine ». Chaque lutin gère la réception de ce message, en testant s'il concerne bien sa case à l'aide de la variable de même nom. Si tel est le cas, il poursuit de façon analogue à ce qu'il sait déjà faire pour répondre à un clic.

Dans le cadre d'un projet courant sur plusieurs séances, on peut distribuer le travail de conception des scripts à différents élèves, en distinguant par exemple, pour un lutin simulant une case, le script à utiliser en réponse à un clic ou en réponse au message « coup machine », et, pour l'arbitre, le test de victoire et la commande du déroulé du jeu, avec la gestion des messages.

Annexe : présentation et installation de Scratch

SCRATCH est un langage de programmation visuel inventé par le laboratoire Lifelong Kindergarten Group du Massachusetts Institute of Technology à vocation ludique et éducative. Son interface ludique et la simplicité de sa prise en main le rendent accessible à tous.
Le logiciel est libre, gratuit et multiplateforme. Le petit chat orange est le « lutin » par défaut.

Site officiel

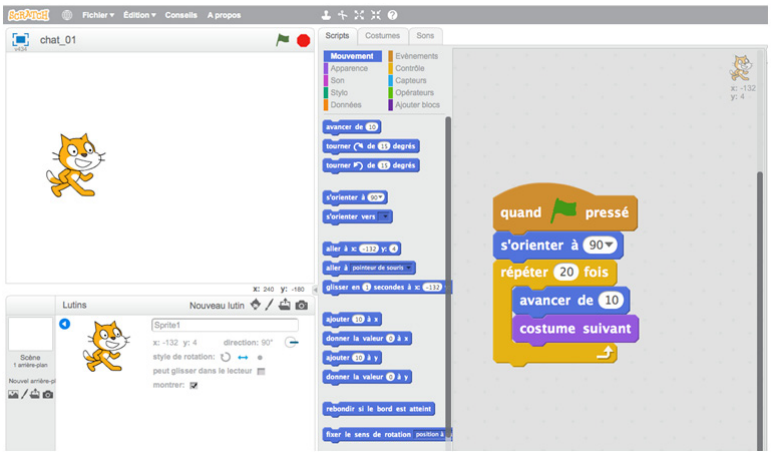
Installer Scratch

Il existe deux méthodes pour utiliser ce logiciel.

- Utilisation hors connexion
  Il faut suivre les indications qu'on trouve en suivant le lien.
  Télécharger (et exécuter si cela ne se fait pas automatiquement) :
  – la dernière version du logiciel Adobe Air ;
  – l'éditeur Scratch hors ligne.
- Utilisation en ligne
  Il faut :
  – disposer d'une version à jour du logiciel Adobe FlashPlayer ;
  – aller sur le site et cliquer sur l'onglet Créer.

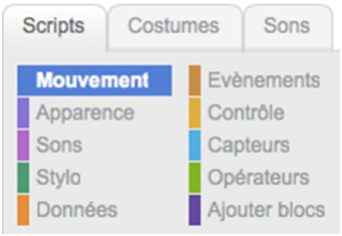
Utiliser Scratch

La langue d'utilisation est par défaut l'anglais ; pour la changer :  en haut à gauche de l'écran dans la barre de menu.



Principe général : pour écrire un programme, on encastre des blocs dans la fenêtre de script, tout à droite. Pour le lancer, on double-clique sur l'assemblage d'instructions ou, si l'on a fait commencer la série par le bloc événement avec le drapeau vert – comme dans l'exemple ci-dessus –, il suffit de cliquer sur le drapeau vert situé au-dessus de la fenêtre d'action, à gauche ; l'octogone rouge situé à sa droite est un bouton d'arrêt.

Les commandes de Scratch

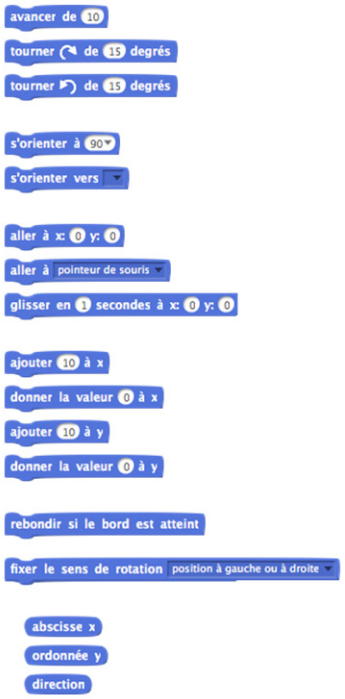


Les différentes commandes sont accessibles, sous l'onglet Scripts, dans la zone centrale. Elles sont regroupées par fonctions, explicitées dans le rectangle représenté ci-dessus. Un code couleur est utilisé pour permettre de distinguer ensuite les fonctions des commandes utilisées.

On récapitule dans le tableau ci-dessous la liste des blocs disponibles, regroupés par catégorie.

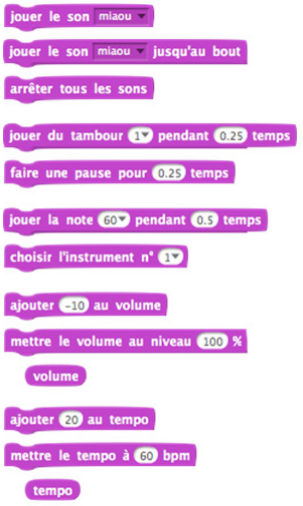
Remarque : les blocs ovales (comme taille ou tempo) représentent des valeurs qui peuvent être utilisées dans les autres blocs. Elles sont précédées d'une case à cocher : si on coche la case, la valeur courante est affichée à l'écran, dans la fenêtre où évolue le lutin (par défaut il s'agit du chat).

MOUVEMENT



APPARENCE

SONS



STYLO

DONNÉES

ÉVÉNEMENTS

Retrouvez Éduscol sur

CONTRÔLE	CAPTEURS	OPÉRATEURS
		

Retrouvez Éduscol sur