Bien commencer avec

Introduction à l'informatique

par Jeremy Scott

LIVRET DE L'ENSEIGNANT

THE ROYAL SOCIETY OF EDINBURGH

bcs The Chartered Institute for IT Academy of Computing

Préambule

Le livret « Starting from Scratch : an introduction to computing science » a été écrit par Jeremy Scott en 2012[1].
La traduction a été effectuée par Inria aux bons soins de http://www.provence-traduction.com et l'adaptation au système éducatif français par Martine Courbin-Coulaud (Inria), en octobre 2014.

Remerciements

L'élaboration du kit « Starting from Scratch : an introduction to computing science» a été en partie subventionnée par Education Scotland, agence nationale écossaise qui œuvre pour l'amélioration de la qualité de l'éducation en Écosse. Nous tenons aussi à remercier les institutions et les personnes suivantes pour leur aide et leur soutien :

Cathkin High School
Linlithgow Academy
Perth High School
George Heriot's School
Stromness Academy
CompEdNet, le forum écossais pour les enseignants d'informatique
Computing At School
Le professeur Hal Abelson, MIT
Mitchel Resnick, MIT
Le Scottish Informatics and Computer Science Alliance (SICSA)
La faculté d'informatique de l'université Napier d'Édimbourg
La faculté d'informatique de l'université de Glasgow
La faculté d'informatique et de mathématiques de l'université Heriot-Watt
La faculté d'informatique de l'université d'Édimbourg
La faculté d'informatique de l'université Robert Gordon
La faculté d'informatique de l'université de Dundee
Le département d'informatique et de mathématiques de l'université de Stirling
La faculté d'informatique de l'université de l'Écosse de l'Ouest
Le Comité International Olympique
ScotlandIS
Turespaña
Brightsolid Online Innovation
JP Morgan
Microsoft Research
Oracle
O2
Sword Ciboodle

Nous sommes également reconnaissants aux personnes suivantes pour leur contribution en qualité de membres du groupe consultatif du projet RSE/BCS :

[1] http://www.royalsoced.org.uk/1050_AnIntroductiontoComputingScience.html

Certains des outils de ce kit sont basés sur des travaux existants issus du site de la communauté ScratchEd, qui ont été reproduits ou adaptés sous licence Creative Commons. L'auteur remercie les personnes concernées d'avoir accepté que leur travail soit utilisé et adapté.

BCS est une organisation caritative reconnue d'intérêt public : N° 292786
La Royal Society of Edinburgh, Académie des Sciences et des Lettres d'Écosse et organisme caritatif écossais N° SC000470.

Sommaire

Présentation

Introduction

Le rapport sur l'enseignement de l'informatique, préparé par un groupe de travail de l'Académie des sciences, « L'enseignement de l'informatique en France : il est urgent de ne plus attendre »[2], présenté en mai 2013 indique un certain nombre de recommandations pour mettre en place un enseignement de la science informatique depuis le primaire jusqu'au lycée, orienté vers la compréhension et la maîtrise de l'informatique.[3]

Ce même rapport indique qu'en Europe, dans l'enseignement primaire et secondaire en général, la plupart des pays ont considéré jusqu'à une période récente l'informatique d'avantage comme un ensemble d'outils permettant de développer des compétences dans les autres disciplines que comme une discipline autonome. Le Royaume Uni, alors que les technologies de l'information et de la communication figurent dans les programmes officiels depuis longtemps, connait depuis quelques temps des changements importants et a mis en place un curricula spécifique en informatique. Dans ce contexte, le kit « Starting from Scratch : an introduction to computing science » a été conçu[4] dans le but d'initier les élèves à l'informatique à travers l'environnement de programmation Scratch, développé au Massachusetts Institute of Technology (MIT).Ce kit à destination de celui qui enseigne et celui qui apprend est une ressource clé en main pour y aider, que ce soit en péri-scolaire ou dans les activités d'enseignement.

[2] http://www.loria.fr/~quinson/blog/2014/0726/Informatique_en_primaire_comment_faire/

[3] En savoir + : https://site.inria.fr/pixees/?p=1728 et http://binaire.blog.lemonde.fr/2014/09/12/informatique-en-primaire-comment-faire/

[4] Ce kit est le premier d'une série de trois kits élaborés par la Royal Society of Edinburgh et la BCS Academy of Computing, qui illustrent un sous-ensemble des objectifs du Curriculum for Excellence[4] (CfE, programme national d'enseignement) concernant l'enseignement de l'informatique au secondaire)

http://www.royalsoced.org.uk/1050_AnIntroductiontoComputingScience.html

La pensée informatique

Il est reconnu que la **pensée informatique** représente un ensemble de compétences essentielles à tout élève au 21e siècle, qu'il ou elle envisage ou non une carrière dans l'informatique. Elle implique d'appréhender le monde selon l'approche employée en programmation par les développeurs de logiciels.

Cette approche peut être scindée en cinq grandes catégories :

- appréhender un problème et sa solution à différents niveaux (**abstraction**)
- réfléchir aux tâches à accomplir sous forme d'une série d'étapes (**algorithmes**)
- comprendre que pour résoudre un problème complexe il faut le décomposer en plusieurs problèmes simples (**décomposition**)
- comprendre qu'il est probable qu'un nouveau problème soit lié à d'autres problèmes déjà résolus par l'élève (**reconnaissance de formes**), et
- réaliser que la solution à un problème peut servir à résoudre tout un éventail de problèmes semblables (**généralisation**).

Par ailleurs, il existe quelques points fondamentaux à comprendre au sujet des ordinateurs :

- Les ordinateurs sont **déterministes** : ils font ce qu'on leur dit de faire. Beaucoup trouveront cela surprenant, car pour eux les ordinateurs sont de la magie pure.
- Les ordinateurs sont **précis** : ils font exactement ce qu'on leur demande de faire.
- Les ordinateurs peuvent donc être **compris** ; il s'agit simplement de machines au fonctionnement logique.

Si la pensée informatique peut être une composante de nombreuses matières, il est tout particulièrement cohérent de l'enseigner dans un cours d'informatique.

Pourquoi Scratch ?

Depuis son lancement, Scratch a reçu un accueil chaleureux en tant qu'outil idéal d'initiation à la programmation et à la pensée informatique.

Son approche basée sur l'utilisation de blocs de base élimine, pour ainsi dire, un problème majeur que posent les langages textuels traditionnels ; celui de devoir mémoriser et taper des instructions selon une syntaxe rigoureuse.

De plus, son approche inspirée des dessins animés et la place importante accordée aux multimédias – sons, éléments graphiques et animations – en font un outil d'initiation à l'informatique intéressant et divertissant ; Scratch est ainsi devenu une star des salles de classe.

Utiliser ce kit

En plus des leçons, des exercices et des exemples de réponses, ce livret suggère aussi des activités supplémentaires et des opportunités d'apprentissage interdisciplinaire.

Ici, l'intention de l'auteur n'est pas que toutes ces activités soient menées à bien. Il ne s'agit que de suggestions quant aux types d'activités grâce auxquelles les enseignants pourraient enrichir l'expérience d'apprentissage des élèves.

N'hésitez pas à utiliser ce kit comme bon vous semble.

N'ayant rien de normatif, il a été conçu pour servir de guide et offrir des suggestions quant aux types d'approches susceptibles (1) de rendre l'apprentissage plus intéressant, (2) de développer la pensée informatique chez les élèves et (3) de les aider à approfondir leur compréhension des concepts de l'informatique.

Scratch

Scratch (**http://scratch.mit.edu/**) est un outil de développement de logiciels développé par le Media Lab du MIT. Il permet à des utilisateurs ayant peu ou pas d'expérience en programmation de créer d'intéressants projets multimédias, qu'ils peuvent ensuite partager avec d'autres utilisateurs à travers le monde, grâce au site Web de Scratch.

Scratch utilise une interface graphique et, pour la créer, le MIT s'est appuyé sur les nombreux travaux de recherche en informatique éducative façonnée par les théories constructionnistes de l'apprentissage. Ces théories mettent en avant la programmation en tant que vecteur d'idées puissantes et intéressantes grâce à un apprentissage actif.

En tant que langage de programmation basé sur l'utilisation de blocs de base, Scratch élimine, pour ainsi dire, un problème majeur que les langages textuels traditionnels posent aux élèves ; celui de devoir mémoriser et taper des instructions selon une syntaxe rigoureuse.

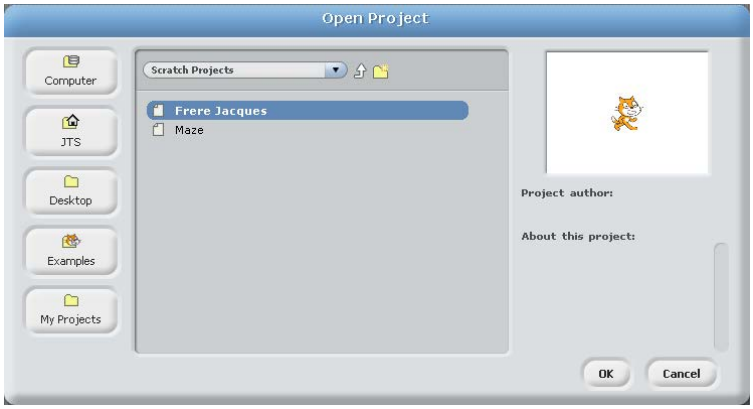
Le nom de Scratch fait référence à l'art de mélanger des sons grâce aux tables de mixage, un procédé de musique hip-hop grâce auquel les DJ réutilisent le travail d'autres artistes pour créer de nouveaux morceaux. Il est aussi à noter que, grâce à son site Web, Scratch accorde une place importante au partage et la collaboration en ligne.

Au moment de la rédaction de ce livret, la dernière version de Scratch était la version v1.4.

http://scratch.mit.edu

Problématiques connues

- Scratch est un environnement de développements multiplateformes (Windows, Macintosh & Linux).
- Pour l'ouverture et la sauvegarde de fichiers, Scratch crée ses propres boîtes de dialogue quelque peu atypiques :



- Scratch n'offre pas encore aux utilisateurs la possibilité de définir eux-mêmes des modules (procédures ou fonctions), mais il est possible de simuler cette fonctionnalité grâce à la commande « envoyer à tous ».
  Toutefois, cette fonctionnalité est présente dans un outil inspiré de Scratch et appelé BYOB (aussi connu sous le nom de Snap!). Cet outil a été développé à l'université de Californie à Berkeley et peut être téléchargé gratuitement à l'adresse suivante http://byob.berkeley.edu/.

Installation

Un peu de préparation est nécessaire avant de pouvoir utiliser Scratch en classe, mais il s'agit d'un produit bien établi qui devrait poser peu de problèmes.

Scratch peut être déployé en réseau sous Windows. Il suffit alors de l'installer sur un unique serveur. Ceci devrait faciliter la gestion du logiciel, y compris la capacité à gérer de façon centralisée le contenu des dossiers médias prédéfinis (lutins, arrière-plans, costumes, etc.).

  - Dans certaines institutions, il a été remarqué que dans ce cas de figure, double-cliquer sur un fichier de projet réalisé avec Scratch n'entraîne pas toujours l'ouverture dudit fichier dans Scratch. Pour résoudre ce problème, il suffit d'ouvrir le projet à partir de l'interface de Scratch (Fichier→Ouvrir…).

L'URL du site Web de Scratch (http://scratch.mit.edu) devrait être mise sur la liste blanche du serveur mandataire de toute école.

Par-dessus tout, il est fortement recommandé aux enseignants de tester l'installation à partir du compte utilisateur d'un élève, avant de tenter d'utiliser Scratch en classe.

Ressources utiles

Il existe de nombreux sites Web d'excellente qualité pouvant être associés à l'utilisation de Scratch. En voici quelques-uns particulièrement intéressants :

http://scratch.mit.edu, la page d'accueil de Scratch au MIT

http://info.scratch.mit.edu/Support/Reference_Guide_1.4, documentation sur Scratch

http://www.scratch.ie, le site du LERO (centre de recherche en génie logiciel irlandais) dédié à Scratch

et en français :

http://ww2.ac-poitiers.fr/rnrtechno/spip.php?article166

http://cursus.edu/institutions-formations-ressources/technologie/23814/scratch-programmation-objet-facile-pour-tous/

Leçons et approches

Les approches suivantes sont susceptibles d'intéresser les enseignants utilisant ce kit :

Captures de vidéos

En plus des outils traditionnellement présents dans un livret, ce kit invite aussi à utiliser des captures vidéos comme support de cours. Les enseignants souhaitant utiliser ces vidéos pourront les montrer à l'ensemble de la classe et/ou inviter les élèves à les regarder individuellement de façon à pouvoir arrêter et reprendre le visionnage selon leurs besoins.

L'idée derrière l'utilisation de ces captures vidéos est de favoriser les progrès des élèves grâce à des outils plus visuels et immédiatement accessibles. Les captures vidéos sont par ailleurs un outil que les élèves connaissent bien dans le monde numérique à travers des services tels que YouTube.

Il est supposé qu'en avançant dans le cours les élèves se familiariseront avec l'utilisation de Scratch, c'est pourquoi l'utilisation des captures vidéos en tant que support diminue progressivement.

Il est à noter que toutes les captures vidéos proposées dans ce kit sont en anglais.

Compréhension profonde

En plus des leçons, ce kit suggère des activités écrites et des idées de discussion visant à permettre aux enseignants et aux élèves d'évaluer à quel point ces derniers ont assimilé les différents concepts de l'informatique. Cette approche s'inspire de travaux récents effectués dans le cadre du programme *CS Principles* des universités de San Diego et de Glasgow. Certains aspects de cette approche sont aussi abordés dans le programme de l'université de Californie à Berkeley intitulé *The Beauty and Joy of Computing.*

Traditionnellement, les enseignants déduisaient le degré de compréhension des élèves à partir des programmes réalisés par ces derniers. Des travaux de recherche ont cependant montré que cet indicateur n'était pas toujours fiable. En conséquence, il est important pour l'enseignant d'utiliser des activités de consolidation des connaissances telles que des quiz, des discussions de groupe, des séances de questions/réponses et des exercices à faire en classe ou à la maison, afin d'évaluer de façon formative la compréhension des élèves tout au long du cours, plutôt que de simplement la déduire à partir des programmes qu'ils auront réalisés.

Programmation en binôme

L'apprentissage collaboratif est favorisé dans les méthodes pédagogiques. Aussi, ce kit favorise ce type d'apprentissage à travers la programmation en binôme. La programmation en binôme peut être définie comme étant :

« [...] une méthode Agile de développement de logiciels dans laquelle deux programmeurs travaillent ensemble sur un même poste de travail. L'un d'eux, le pilote, tape le code pendant que l'autre, le copilote, vérifie chaque ligne de code alors qu'elle est tapée. Les deux programmeurs changent fréquemment de rôle.

Tout en vérifiant le code, le ou la copilote réfléchit aussi à l'orientation stratégique du travail. Il ou elle peut proposer des idées d'amélioration et repérer les problèmes susceptibles de survenir afin d'y remédier en amont. Ceci laisse la liberté au pilote de concentrer toute son attention sur les aspects "stratégiques" de la réalisation de la tâche en question, sachant que le copilote est là et sert de filet de protection et de guide. »
Traduit de Wikipedia

La programmation en binôme peut servir à encourager la collaboration entre élèves et l'utilisation des ressources disponibles en classe.

Il est conseillé aux enseignants d'inciter les élèves à demander l'aide d'un de leurs camarades de classe avant de demander celle de l'enseignant. Toutefois, les enseignants comprendront l'importance de veiller à ce que les deux élèves soient impliqués à part égale dans le travail !

Activités suggérées

Ces notes contiennent des suggestions à l'intention des enseignants concernant la façon d'approfondir les points abordés en classe. Elles n'ont aucun caractère normatif et sont simplement des moyens possibles de pousser une activité ou un sujet. Les enseignants sont libres de sélectionner les activités comme ils l'entendent, car les faire toutes risque de rallonger considérablement le cours.

Activités suggérées

Ces activités sont signalées par la mention « **Activités suggérées** » dans la marge de gauche.

Apprentissage interdisciplinaire

Scratch est un environnement multimédia « riche » qui offre de nombreuses possibilités de faire le lien avec d'autres matières. En plus de proposer des activités

interdisciplinaires par essence, le présent kit contient aussi des suggestions d'autres activités interdisciplinaires.

AID Les opportunités d'apprentissage interdisciplinaire sont signalées par la mention « **AID** » dans la marge de gauche.

Introduction

Cette section explique brièvement ce qu'est un ordinateur. Il appartient aux enseignants de décider d'utiliser cette section comme une introduction ou d'intégrer les activités proposées à la séance d'exercices pratiques.

Qu'est-ce qu'un ordinateur ?

Souligner le caractère omniprésent des ordinateurs et le fait que leur utilisation est étroitement liée à de nombreux domaines de la vie moderne. Toutefois, les ordinateurs sont des machines qui exécutent des instructions données par des humains. Sans instructions, un ordinateur serait incapable de faire quoi que ce soit.

- Souligner les « forces » et les « faiblesses » des ordinateurs.

Activité

Écris trois tâches quotidiennes réalisées par les humains et que les ordinateurs ne peuvent effectuer (ou pour lesquelles ils ne sont pas très efficaces).

1. Avoir une conversation / raconter une blague

2. Préparer un repas

3. S'occuper d'un enfant

Activité suggérée

Permettre aux élèves d'utiliser un chatbot en ligne tel que A. L. I. C. E. (http://alice.pandorabots.com/), afin de faire ressortir les limites des capacités de conversation des ordinateurs. Ceci pourrait aussi être l'occasion de parler d'Alan Turing, du test de Turing et de l'intelligence artificielle.

Activité suggérée

Montrer aux élèves quelques clips vidéos de robots actuels, comme NAO créé par Aldebaran :
http://www.aldebaran-robotics.com/fr/

ou le robot Asimo de chez Honda, essayant de réaliser des tâches de routine – beaucoup de vidéos d'Asimo sont disponibles sur la chaîne YouTube qui lui est consacrée :
http://www.youtube.com/playlist?list=PL2FEFAF55603817E7&feature=plcp

Toutefois, certains des exemples les plus parlants sont des vidéos montrant les domaines dans lesquels les robots comme Asimo ne sont pas à la hauteur ! Il existe de nombreux exemples sur YouTube et chez d'autres fournisseurs de vidéos.

Les types d'ordinateurs

En toute probabilité, les élèves connaîtront les trois catégories d'ordinateurs individuels modernes.

Activité

Les types d'ordinateurs individuels décrits précédemment sont présentés dans l'ordre du plus ancien au plus récent.
Qu'est-ce que cela nous apprend quant au type d'ordinateurs que veulent les gens ?

Les réponses des élèves devraient être très variées. Quelques caractéristiques essentielles sont :

- **portable**
- **facile à utiliser**
- **peu encombrant**
- **connecté à Internet**

Les élèves ne connaîtront peut-être pas aussi bien les ordinateurs centraux et les serveurs. Il sera peut-être difficile pour beaucoup de comprendre qu'une console de jeux ou un smartphone est aussi un ordinateur.

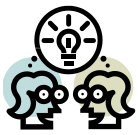
De nombreux élèves n'auront jamais entendu le terme « système embarqué ». S'attarder sur ce type d'ordinateur, et inviter les élèves à réfléchir aux endroits où il est possible de trouver des systèmes embarqués.

Activité

Nomme trois appareils présents chez toi, qui selon toi peuvent contenir un système embarqué (mis à part les exemples donnés ci-dessus).

De nombreux exemples existent : lecteur de DVD/de disque Blu-ray, imprimante, cuiseur, four à micro-ondes, lave-vaisselle, réveil, radio, récepteur satellite/câble

Activité



Nomme trois technologies que l'on retrouve dans un smartphone moderne.

S'agissant du matériel, on peut citer les éléments suivants : boussole, accéléromètre, récepteur GPS, écran tactile, radio Bluetooth, radio cellulaire, radio FM, haut-parleur, décodeur audio.

Il est aussi possible de citer une large gamme de logiciels.

Les différents éléments d'un ordinateur

Cette section est axée sur la saisie, le traitement, le stockage et la récupération en sortie d'informations plutôt que sur les appareils effectivement utilisés pour ces tâches et dont utilisation est induite par les tâches.

Activité suggérée

Inciter les élèves à réfléchir aux éléments en entrée, aux traitements et aux éléments en sortie d'appareils ne relevant pas de l'informatique.

ex. une machine à laver

éléments en entrée = linge sale, lessive, électricité

traitement = chauffer l'eau, faire tourner le tambour pour laver les vêtements

éléments en sortie = eau savonneuse sale, vêtements propres

Activité

Indique les données d'<u>entrée</u> et de <u>sortie</u> des activités suivantes réalisées sur différents types d'ordinateur. Lorsque tu auras terminé, rajoute une activité :

Activité	Donnée(s) d'entrée	Donnée(s) de sortie
Jouer à un jeu vidéo	Déplacement du contrôleur de jeu Clics sur les boutons	Le personnage se déplace Les menus sont sélectionnés
Naviguer sur le Web	**Clics de souris**	**Une autre page s'ouvre**
Téléphoner	**Bouton pressé**	**Le numéro est composé**
Regarder la télé	**Boutons de télécommande pressés**	**Changement de chaîne** **Réglage du volume**
Le choix de l'élève		

Le matériel informatique

Activité **À quelle catégorie penses-tu que les appareils ci-dessous appartiennent ? Répartis les périphériques d'entrée, de sortie et de stockage dans le tableau ci-dessous. Les trois premières cases ont été remplies pour toi.**

clavier, lecteur de disque dur, écran, haut-parleur, scanner, imprimante, souris, lecteur de DVD, microphone, clé USB, contrôleur de jeu, écran tactile de smartphone, carte mémoire

Périphérique d'entrée	Périphérique de stockage	Périphérique de sortie
clavier	lecteur de disque dur	écran
scanner	lecteur de DVD	haut-parleur
souris	clé USB	imprimante
microphone	carte mémoire	
contrôleur de jeu (boutons/ détecteur de mouvement)		contrôleur de jeu (retour de vibrations)
écran tactile de smartphone		écran tactile de smartphone

Il est probable que les élèves classent à tort le lecteur de disque dans la catégorie des périphériques d'entrée, parce qu'il fournit des informations à l'ordinateur (ou dans la catégorie des périphériques de sortie, parce qu'il reçoit les informations que nous sauvegardons).

Mettre l'accent sur le fait qu'il ne s'agit ni de l'un ni de l'autre. Un lecteur de disque est un **périphérique de stockage** : il stocke les informations lorsque l'ordinateur est éteint. Souligner le fait que les informations stockées par le lecteur y ont été **initialement** introduites grâce à un périphérique d'entrée.

Les logiciels

Mettre l'accent sur le fait que sans logiciel, le matériel informatique ne sert à rien. Par exemple :

Question : Combien de tâches différentes la plupart des machines peuvent-elles réaliser ?
Réponse : Une seule.

Bien qu'elle puisse avoir différents réglages, l'unique fonction d'une machine à laver est de laver les vêtements. Et une bouilloire ? Et une voiture ? De façon générale, toutes ces machines n'ont qu'une utilité : laver les vêtements, chauffer l'eau, transporter des personnes.

Question : Combien de tâches différentes un ordinateur peut-il exécuter ?
Réponse : Beaucoup – tout dépend du logiciel utilisé.

C'est ce qui différencie les ordinateurs des autres machines.

Activité **Note dans le tableau ci-dessous dix tâches différentes que tu peux faire sur ordinateur. Pour chacune d'elles, donne le nom d'un logiciel qui permet de la réaliser.**

Tâche	Logiciel
Naviguer sur le Web	Google Chrome
Jouer à des jeux	Angry Birds
Modifier un fichier vidéo	iMovie
Rédiger une rédaction	Microsoft Word
Rester en contact avec des amis	Facebook
Enregistrer de la musique	Steinberg Cubase
Créer une lettre d'information (newsletter)	Microsoft Publisher
Créer une présentation	Microsoft PowerPoint
Faire des calculs	Microsoft Excel
Écouter de la musique	Apple iTunes

Les langages de programmation

Rappeler que les ordinateurs sont des machines qui suivent des instructions et souligner les points suivants introduits en page 3 de ce livret :

- Les ordinateurs sont **déterministes** : ils font ce qu'on leur dit de faire.
- Les ordinateurs sont **précis** : ils font exactement ce qu'on leur demande de faire.

Les ordinateurs peuvent donc être **compris** ; il s'agit simplement de machines au fonctionnement logique. Si un ordinateur ne fonctionne pas correctement, c'est parce qu'il a reçu de mauvaises instructions (en supposant que le matériel informatique fonctionne).

Activité suggérée

Montrer aux élèves des exemples d'un même algorithme mis en œuvre dans deux ou trois langages différents, idéalement dans un langage de haut niveau (tel que Scratch), dans un langage de plus bas niveau (comme Basic ou C++) et peut-être même dans un programme en langage d'assemblage. La notion d'abstraction est un axe majeur du 2e kit de cette série d'outils pédagogiques[5].

Activité suggérée

Discuter avec les élèves de l'omniprésence et de l'importance des ordinateurs dans la société d'aujourd'hui. Du fait de cette importance, le secteur de l'informatique est l'un des plus gros employeurs à l'international et offre des carrières souvent lucratives et gratifiantes.

Aborder aussi le fait que, malgré cette omniprésence et cette importance, beaucoup de personnes voient encore l'ordinateur comme une sorte de boîte magique qu'ils comprennent moins bien que la voiture qu'ils conduisent. Pourquoi est-il important de comprendre ce qu'est un ordinateur ?

[5] http://www.royalsoced.org.uk/1053_AnIntermediateCourseinComputingScience.html

Apprentissage interdisciplinaire étendu

Citoyenneté mondiale

Comment les ordinateurs aident-ils les habitants des pays en développement ?

Citoyens responsables

Où vont tous les ordinateurs ?
Étudier la problématique des déchets électroniques et de leur élimination.

Qualité de vie

Les élèves pourraient étudier les risques pour la santé...

- Risques associés à la sédentarité
- Distraction des automobilistes ou des piétons due aux smartphones
- Perturbations des cycles du sommeil du fait de passer trop de temps devant un écran

... et les bienfaits

- avancées médicales que seule l'utilisation d'ordinateurs a rendu possibles
- dispositifs de sécurité à la maison, au travail et dans les véhicules
- utilisation des ordinateurs par la police pour arrêter les criminels et nous protéger.
- meilleur accès aux informations sur la santé grâce à Internet

associés à l'utilisation des ordinateurs (de tous types).

1 : Les bases de Scratch

Les concepts introduits

- L'environnement de Scratch, avec
    - Les lutins et la scène
    - Les propriétés
        - Les scripts
        - Les costumes et les arrière-plans
        - Les sons
- La création d'un programme avec sons et animations
- La rédaction de séquences d'instructions
- Les boucles « répéter »

Les commandes de Scratch introduites

- Mouvement
    - avancer de <n> pas
    - tourner de <n> degrés
- Contrôle
    - quand *drapeau* pressé
    - attendre <n> secondes
    - répéter <n> fois
- Sons
    - jouer note <ton> [pour n temps]
    - jouer le son <nom du son> [complètement]
- Apparence
    - modifier l'effet <nom de l'effet> par <quantité>
    - costume suivant

La pensée informatique et ses thématiques

- L'abstraction
    - ce qu'il se passe, ex. un son est joué, le lutin bouge
    - position représentée grâce aux coordonnées x et y
    - le ton représenté par le son
- La reconnaissance de formes
    - utilisation d'une boucle « répéter » pour répéter du code
- La décomposition
    - utilisation de scripts distincts pour la résolution de sous-problèmes distincts
- Les algorithmes
    - séquence
    - action déclenchant un événement

Les objectifs

L'élève devrait être capable :

- d'identifier les principaux éléments de l'environnement de Scratch
- de comprendre comment les lutins et les blocs fonctionnent et interagissent
- de comprendre qu'un programme informatique est un ensemble d'instructions
- de travailler avec des animations et des sons simples
- de comprendre la différence entre exécuter des instructions en parallèle ou de façon séquentielle

Les ressources

- Projets réalisés avec Scratch : **Premiers pas, Frère Jacques**
- Captures vidéos : **Catwalk, FrereJacques**

Introduction

- Les enseignants pourraient montrer aux élèves une vidéo d'introduction à Scratch ou les laisser la regarder individuellement.
- Présenter une sélection de projets issus du site Web de Scratch (http://scratch.mit.edu), afin de montrer aux élèves ce qu'il est possible de faire.

Tâche nº 1 : Premiers pas

- Les enseignants pourront laisser les élèves regarder la capture vidéo « **Catwalk** » ou faire eux-mêmes une visite guidée de l'environnement de Scratch, en mettant l'accent sur (1) les lutins et la scène, et (2) les scripts, les costumes, les arrière-plans et les sons.
- Une fois la visite guidée achevée, les élèves pourront tenter de créer un programme simple afin de faire danser un lutin en utilisant les blocs de mouvement et les blocs **costume suivant**.

Tâche nº 2 : Frère Jacques

- Après avoir réalisé la première tâche, les élèves pourront regarder la capture vidéo « **FrereJacques** », qui les guidera dans la création d'un simple programme de musique. **Il est à noter que cette activité permet d'introduire le mode pas-à-pas (menu d'édition). Il faudra que les élèves désactivent ce mode pour exécuter les scripts normalement.**
- Souligner la différence entre **jouer le son** et **jouer le son complètement**.

Tâche nº 3 : Mes mélodies

- Les élèves devraient ensuite créer leur propre petite mélodie. Les comptines sont parfaites pour cette activité, car elles sont caractérisées par la répétition et l'utilisation de mélodies simples.

Activité supplémentaire nº 1 : Faut que ça danse !

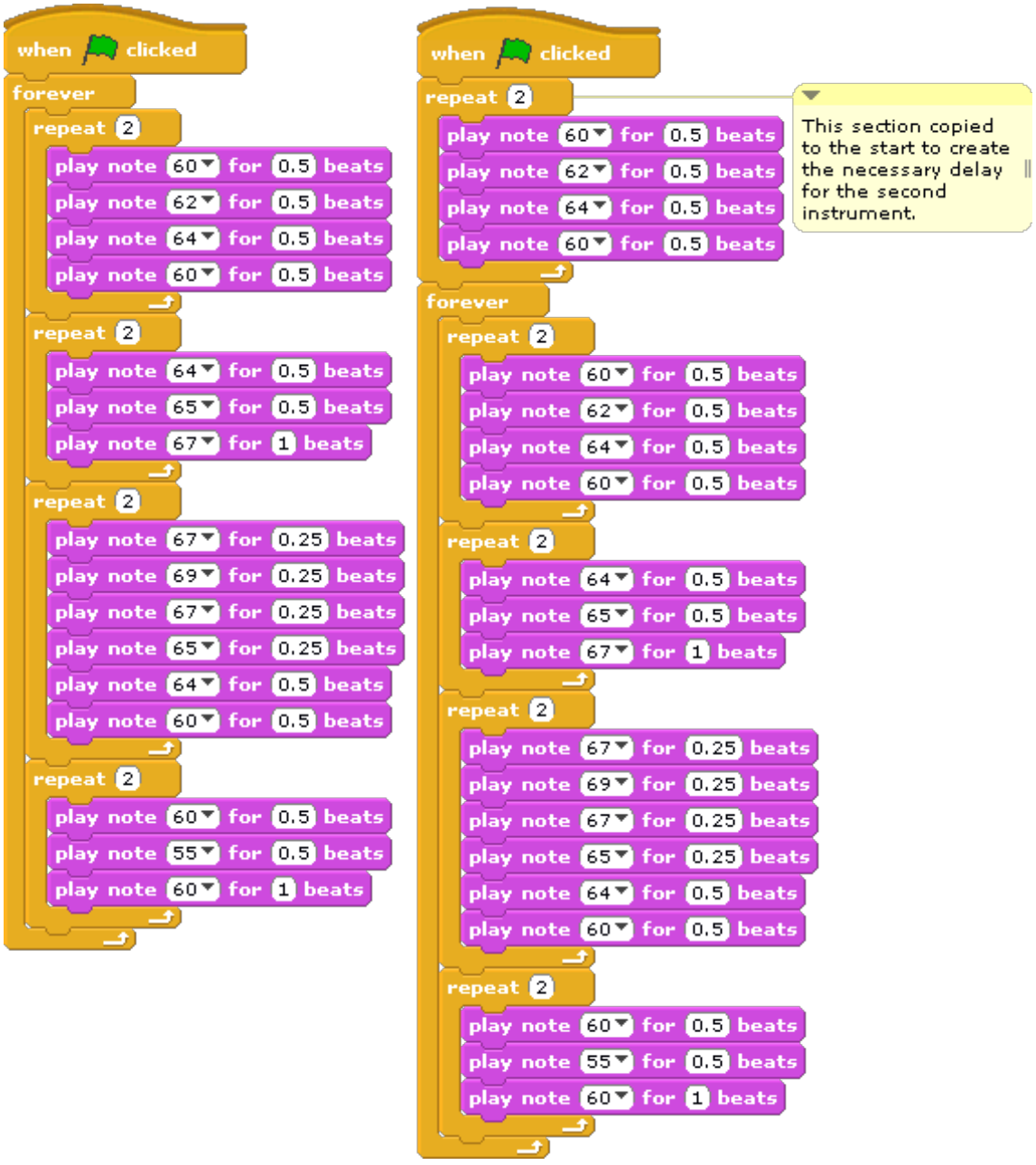
- Créer un script qui fera danser un lutin au son de la mélodie créée par les élèves pour la tâche n° 3 (ci-dessus).
- S'ils le souhaitent, les enseignants pourront inviter les élèves à comparer deux méthodes permettant de faire danser le lutin
    - imbriquer des blocs de mouvement dans le script permettant de jouer une mélodie
    - utiliser un script distinct exécuté en parallèle.
- La notion de parallélisme pourra être davantage explorée à travers d'utilisation de plusieurs scripts pour jouer un accord, plutôt que des notes distinctes, à partir du clic sur le drapeau vert. Par exemple,

Activité suggérée

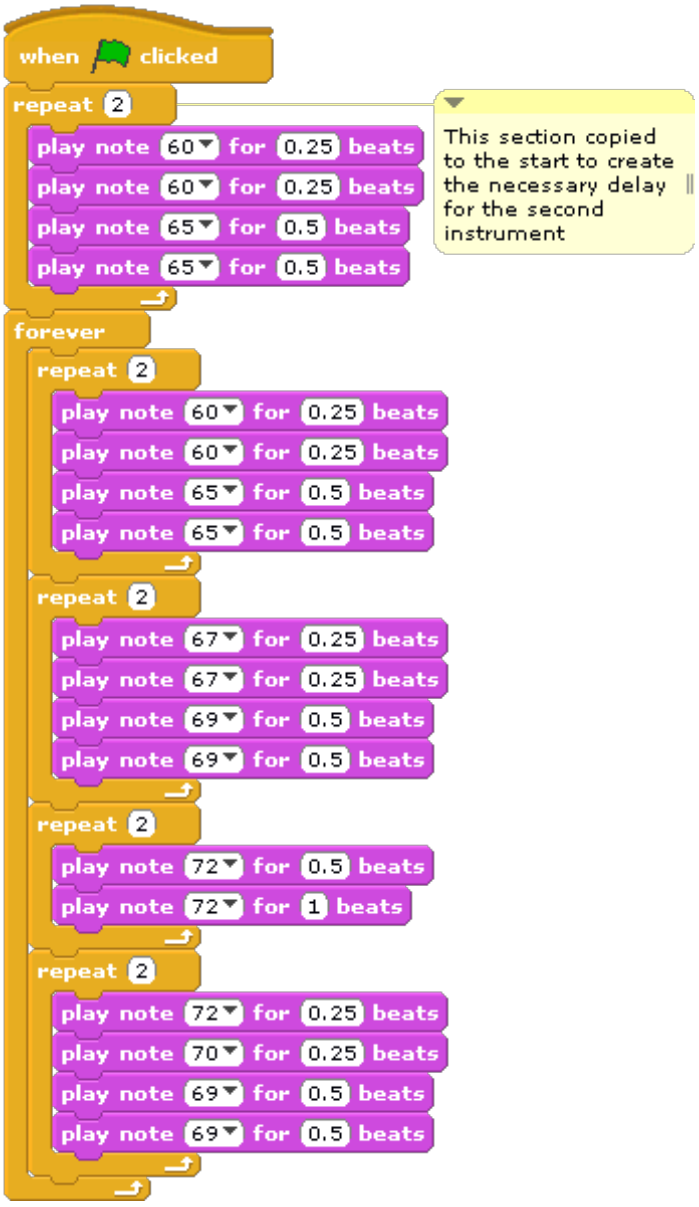
Frère Jacques se chante souvent *en canon*. Dans un canon, deux voix ou plus (ou des instruments) chantent ou jouent le même morceau de musique en démarrant à des instants différents et chaque voix reprend la chanson à son début.

Cette activité peut être vue non seulement comme une opportunité d'apprentissage interdisciplinaire, mais aussi comme une occasion d'approfondir la notion de parallélisme. Il est à noter que plusieurs niveaux de parallélisme peuvent être associés pour créer un bel effet musical.

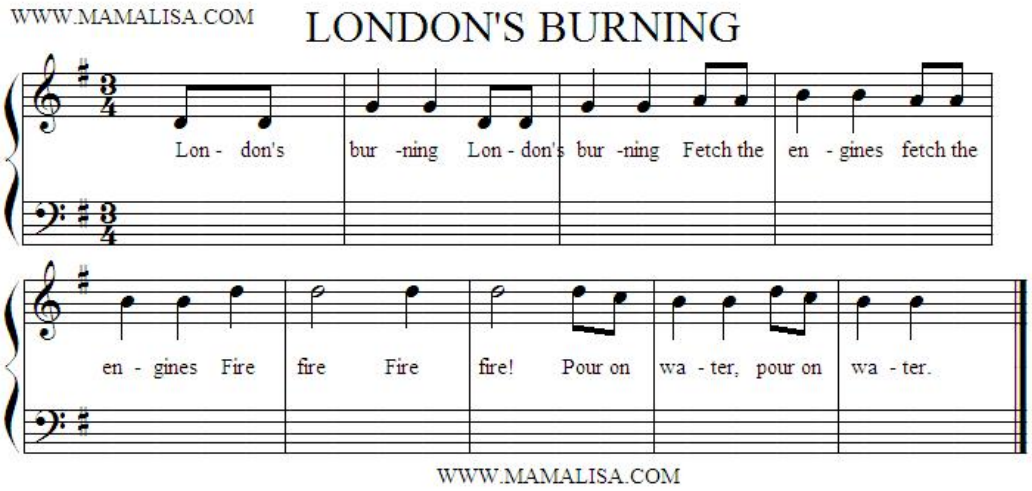
Un exemple de scripts à exécuter en parallèle pour cette activité est donné ci-dessous. Dans cet exemple, une boucle infinie est utilisée pour créer un canon.



Un autre canon bien connu est *London's Burning*[6]. Un exemple de scripts à exécuter en parallèle pour ce canon est donné ci-dessous. Dans cet exemple, une boucle infinie est utilisée pour créer un canon.



[6] https://www.youtube.com/watch?v=dRJBegR-SgM



©www.MAMALISA.COM

AID Faire traduire les paroles de la chanson par les élèves.

London's burning, London's burning.
Fetch the engines, fetch the engines.
Fire fire, fire fire!
Pour on water, pour on water.

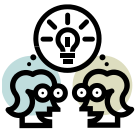
Londres flambe, Londres flambe
Aux machines, aux machines
Au feu, au feu
Versez de l'eau, versez de l'eau.

Activité supplémentaire nº 2 :

- Les élèves pourront s'amuser à ajouter des couleurs et autres effets à leurs lutins pendant qu'ils dansent. Pour ce faire, ils devront utiliser le bloc **modifier l'effet <nom de l'effet> par <quantité>** de la catégorie **Apparence**.

As-tu compris ?

Les activités « As-tu compris ? » sont l'un des principaux outils utilisés pour stimuler la pensée informatique chez les élèves. Les enseignants pourront par exemple envisager d'utiliser ces activités :



- **pour des discussions de groupe, menées en binôme ou avec l'ensemble de la classe**
- **comme exercices à faire en cours ou la maison**
- **pour favoriser l'éducation par les pairs**

1.1 Étudie le bout de code ci-contre qui contrôle un lutin. Que crois-tu que l'utilisateur verra en cliquant sur le drapeau vert ?

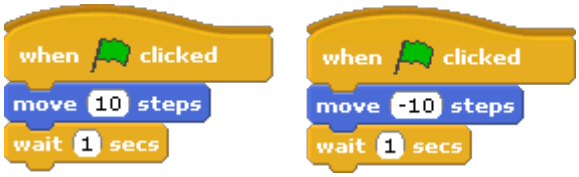
Il semble que le lutin n'a pas bougé.

Pourquoi ? L'ordinateur a exécuté les instructions trop vite pour qu'on puisse voir un quelconque mouvement.

1.2 Maintenant, ajoute un bloc « **attendre 1 seconde** » entre les deux blocs de mouvement. Décris ce qui se passe.

On a vu le lutin bouger.
Souligner le fait qu'un ordinateur travaille très vite. Dans le premier exemple, le lutin a bel et bien bougé dans un sens puis dans l'autre, mais ça s'est passé si vite qu'on n'a pas pu le voir !

1.3 Étudie le bout de code ci-dessous qui contrôle un lutin.



Que crois-tu que l'utilisateur verra en cliquant sur le drapeau vert ?

Le lutin n'a pas bougé.

Pourquoi ? **Les deux scripts s'exécutent en même temps – en parallèle – et donc s'annulent l'un l'autre.**

1.4 Avec la pile de blocs ci-dessous, combien de fois le lutin avancera-t-il de 10 pas ?

Huit fois.

1.5 Un programmeur veut que le chat danse au son d'une musique. Toutefois, le chat ne commence à danser **que lorsqu'il n'y a plus** de musique !

L'ordinateur joue entièrement le son « hip-hop » avant de passer à l'instruction suivante. Le programmeur aurait pu choisir une des deux méthodes suivantes :

- **utiliser un bloc jouer le son hip-hop**
- **intégrer le bloc jouer le son complètement dans un autre script de type « quand drapeau pressé ».**

1.6 Dans l'exemple ci-dessous, un programmeur a choisi un morceau de musique (le son « Xylo1 ») pour la musique d'un jeu. Cependant, lorsque l'utilisateur clique sur le drapeau vert, l'ordinateur ne joue que la première note du morceau... en boucle !

Quelle erreur le programmeur a-t-il faite ?

Le programme est bloqué dans la boucle infinie, qui relance sans cesse le bloc jouer le son sans lui donner le temps d'aller jusqu'au bout. Le programmeur aurait dû opter pour un bloc jouer le son complètement.

1.7 Dans l'**activité supplémentaire n° 1 « Faut que ça danse ! »**, tu as fait danser un lutin au son d'une mélodie que tu as créée. Il y avait **deux** méthodes pour y arriver :

- en créant un **unique script** où les blocs de mouvement du lutin se trouvent parmi les blocs permettant de jouer une note.
- en créant des **scripts distincts** pour un même lutin – un script joue la mélodie pendant qu'un autre fait danser le lutin.

À ton avis, pourquoi des programmeurs expérimentés utiliseraient-ils des **scripts distincts** ?

Cela permet de faire la distinction entre les différentes activités, ce qui permet aux programmeurs de se concentrer sur une chose à la fois.
Souligner le fait qu'il s'agit là d'une importante notion en informatique ; décomposer un problème en problèmes plus simples et résoudre chaque petit problème individuellement. Nous rencontrerons souvent cette thématique de la pensée informatique.

1.8 Prépare une autre question du même genre que les questions 1.1 à 1.5 et pose-la à ton voisin.

Encourager les élèves à anticiper les erreurs de compréhension ou autres susceptibles de survenir (qu'eux-mêmes ont peut-être déjà rencontrées).

Paresseux ou astucieux ?

Les programmeurs sont toujours à la recherche de raccourcis pour faciliter leur existence.

Un bon exemple est la façon dont nous avons utilisé le bloc **répéter** dans l'activité « Frère Jacques », pour répéter la même ligne de musique plutôt que d'avoir deux piles de blocs identiques. En plus du fait que la présentation est plus soignée, cela élimine aussi le risque de faire une erreur au cours de la création d'un second groupe de blocs.

D'après toi, cela veut-il dire que les programmeurs sont paresseux ou astucieux ?

(**Indice** : La réponse est « astucieux » !)

Toi aussi tu peux te faciliter la vie en repérant de tels raccourcis.

Activité suggérée

Discuter de l'encart ci-dessus avec les élèves. Souligner l'importance des thématiques de la pensée informatique telles que la reconnaissance de formes, et mettre l'accent sur la satisfaction de trouver une solution succincte et élégante à un problème.

Conclusion

- Réviser l'environnement de Scratch.
- Demander aux élèves de résumer ce qu'ils ont vu et appris.
- Mettre l'accent sur l'importance fondamentale de l'**ordre d'exécution** en programmation.
  Mettre l'accent sur le fait qu'un ordinateur ne fera que **ce qu'on lui demande de faire/ce pour quoi il a été programmé.**
  Si un programme ne fonctionne pas comme prévu, c'est qu'une erreur a été commise !

Autres activités supplémentaires

- Laisser les élèves faire des tests avec différents sons/instruments pour leurs mélodies.
- Créer un programme pour que plusieurs lutins animés dansent tous ensemble sur scène.
- Photographier les élèves prenant différentes poses de danse (pour que les photos servent de costumes aux lutins comme les costumes de Cassy déjà disponibles dans Scratch). Importer les photos pour servir de costumes à un lutin, afin que les élèves puissent se faire danser.
    - Créer une fête virtuelle dans Scratch, avec des groupes entiers d'élèves dansant tous ensemble.

2 : C'est l'heure d'une histoire

Les concepts introduits

- Consolidation des connaissances concernant l'environnement de Scratch, y compris
    - Les lutins et la scène
    - Les propriétés
        - Les scripts
        - Les costumes et les arrière-plans
        - Les sons
- La rédaction de séquences d'instructions
- L'exécution du code de façon séquentielle et en parallèle
- Les erreurs de programmation et le débogage
- La programmation événementielle (y compris avec des événements définis par le programmeur)

Les commandes de Scratch introduites

- Contrôle
    - envoyer à tous <message>
    - quand je reçois <message>
- Sons
    - jouer le son <nom du son> [complètement]
- Apparence
    - basculer sur le costume <nom du costume>
    - dire <chaîne de caractères> pendant <n> secondes

La pensée informatique et ses thématiques

- L'abstraction
    - ce qu'il se passe, ex. un son est joué, le lutin bouge
- La décomposition
    - utilisation de scripts distincts pour la résolution de sous-problèmes distincts
- Les algorithmes
    - séquence
    - action déclenchant un événement

Les objectifs

L'élève devrait être capable :

- de créer des histoires et des pièces de théâtre
- d'écrire des séquences d'instructions
- de créer ses propres événements en utilisant la commande « envoyer à tous »
- de développer un projet étape par étape

Les ressources

- Les projets : **Une mauvaise blague**
- Captures vidéos : **BadJoke**

Introduction

Montrer aux élèves la capture vidéo « **BadJoke** » ou leur permettre de la regarder seul(e).

Tâche nº 1 : Une mauvaise blague

Faire une démonstration ou permettre aux élèves de regarder la capture vidéo **BadJoke**. Elle montre comment utiliser Scratch pour raconter une blague ou monter une pièce avec deux personnages.

La fille	Le garçon
Dire « Hey, je connais une blague ! » pendant 3 secondes	Attendre 3 secondes
Attendre 3 secondes	Dire « Vas-y, raconte ! » pendant 3 secondes
Dire « Mon chien n'a pas de museau » pendant 3 secondes	Attendre 3 secondes
Attendre 3 secondes	Basculer sur le costume du garçon haussant les épaules Dire « Comment sent-il ? » pendant 3 secondes
Dire « Mauvais » pendant 2 secondes	Attendre 2 secondes
	Basculer sur le costume du garçon qui rit Dire « <Soupir> » pendant 3 secondes

2.1 Fais une liste de tous les problèmes rencontrés et des solutions que tu as trouvées pour les résoudre.

Il est probable que la plupart des problèmes rencontrés par les élèves seront liés aux points suivants :

- **s'assurer que chaque lutin parle au bon moment**
- **maintenir une conversation synchronisée si des durées différentes sont utilisées pour la commande « dire »**
- **se souvenir à quel lutin correspond chaque script**

Tâche nº 2 : Une petite pièce de théâtre

Mettre l'accent sur les points suivants auprès des élèves :

- Ils doivent privilégier la simplicité en n'utilisant que deux ou trois acteurs (lutins).
- Ils doivent écrire un script sur papier ligné, en rédigeant les lignes de chaque acteur côte à côte, comme dans l'exemple précédent.

Faire une démonstration de l'utilisation du bloc « envoyer à tous ». Il appartient à l'enseignant de décider s'il est préférable d'introduire ici la différence entre les blocs « envoyer à tous » et « envoyer à tous et attendre » ou s'il vaut mieux attendre.

Pour cette tâche, les élèves pourront éventuellement s'inspirer d'une capture vidéo du MIT intitulée « **Haunted Scratch** ». Elle est disponible à l'adresse suivante : http://info.scratch.mit.edu/node/165

Activité supplémentaire nº 1 : Marcher c'est bien

Cette activité introduit les notions d'initialisation et de réinitialisation de l'état d'un programme chaque fois qu'il est exécuté. Mettre l'accent sur le fait que chaque petit détail doit être programmé : la position de départ, la direction, etc.

Les erreurs de programmation

Une **erreur de programmation** (ou « **bug** ») est une erreur qui empêche ton code de fonctionner comme prévu. Il existe **deux** grandes catégories d'erreurs de programmation :

- **Les erreurs de syntaxe**
  Ce genre d'erreur survient si l'on enfreint les règles du langage de programmation, par exemple en faisant une faute d'orthographe dans une commande. Généralement, les erreurs de syntaxe empêchent l'exécution du code. Certains langages comme Scratch fournissent du code sous forme de blocs déjà rédigés, ce qui élimine le risque d'erreur de syntaxe.
- **Les erreurs de raisonnement**
  Ceci signifie que le code est exécuté, mais ne donne pas le résultat prévu. Malheureusement, il est possible de faire des erreurs de raisonnement avec Scratch !

Identifier et résoudre ces erreurs de programmation est un processus appelé « **débogage** ».

Activité suggérée

Discuter de l'encart ci-dessus avec les élèves. Introduire la notion d'erreurs de programmation. Souligner le fait que les ordinateurs sont déterministes. Par conséquent, si le code ne fonctionne pas comme prévu, c'est que le programmeur a fait une erreur.

Dans le cadre de cette discussion, poser la question suivante :

Q : Comment pourrait-on trouver les erreurs dans nos programmes ?
R : En les testant. C'est là tout l'intérêt de la phase de tests : identifier les erreurs éventuelles et les corriger.

Introduire ensuite les notions de conception et d'algorithmes :

Q : Que pourrions-nous faire pour réduire le risque de voir apparaître des erreurs dans notre code ?
R : Le planifier, tout comme nous avons planifié notre pièce.

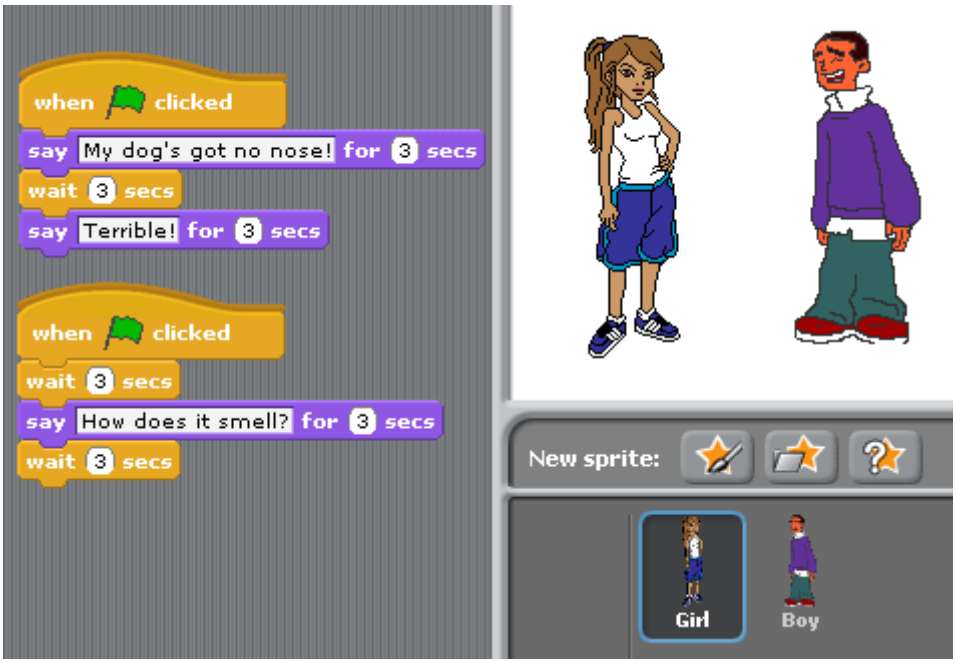
As-tu compris ?

2.1 Le programme ci-dessous montre les scripts permettant à deux lutins de se raconter une blague. Pourquoi ce programme ne peut-il fonctionner ?

La fille	Le garçon
when clicked say Knock knock! for 3 secs say Doris. for 3 secs say Doris locked.That's why I'm knocking! for 3 secs	when clicked say Who's there? for 3 secs say Doris who? for 3 secs say GROAN! for 3 secs

Il n'y a pas de pause, ce qui implique que les personnages parlent tous en même temps. Il faut insérer des commandes attendre appropriées.

2.2 Étudie le programme ci-dessous qui permet de raconter une blague. Mis à part le fait que la blague est très mauvaise, qu'est-ce qui ne va pas dans ce programme ?



Les deux scripts/parties de la blague sont stockés pour le même lutin (la fille). La fille se raconte la blague à elle-même.

2.3 Le programme ci-dessous montre les scripts permettant à deux lutins de se raconter une blague. Pourquoi ce programme ne peut-il fonctionner ?

La fille	Le garçon
	

Les temps ne sont pas adéquats. Il faut ajuster les temps accordés aux commandes attendre et dire pour que les lignes des personnages soient correctement synchronisées.

2.4 À présent, prépare une autre question de débogage et pose-la à ton voisin.

La programmation événementielle

Une fois lancés, certains programmes informatiques tournent seuls sans données d'entrée provenant de l'utilisateur. C'est notamment le cas du programme créé pour jouer une mélodie.

Toutefois, de nombreux programmes réagissent à des **événements** (des choses qui se produisent), tels que :

- un clic de souris ou une touche pressée ;
- un basculement de contrôleur de jeu ;
- un effleurement d'écran de smartphone ;
- un mouvement corporel détecté par un contrôleur de jeu à détecteur de mouvement tel que Kinect

Dans Scratch, les blocs d'événement possèdent un sommet bombé (parfois appelé « chapeau ») :

Réagit lorsque l'utilisateur clique sur le drapeau vert. Souvent utilisé pour lancer un programme.

Réagit lorsque l'on appui sur une touche. Clique sur le petit triangle noir pour sélectionner la touche que tu veux détecter. Utile pour contrôler un lutin ou déclencher une action.

Réagit lorsque l'utilisateur clique sur un lutin. Utile pour contrôler les personnages d'un programme.

Tu peux aussi créer tes propres événements dans Scratch en utilisant la commande « **envoyer à tous** ».

2.4 Étudie l'environnement de Scratch et repère d'autres **événements** ou **conditions** auxquels les programmes en Scratch peuvent réagir. Note-les ci-dessous.
Astuce : les blocs des catégories **Contrôle** et **Capteurs** sont un bon début.

- **Toucher le bord de l'écran (rebondir si le bord est atteint)**
- **envoyer à tous un événement que l'utilisateur peut définir**
- **<propriété> de <objet>, par exemple :**
  - **position x du lutin**

    - **direction du lutin**
    - **arrière-plan de la scène**
- **lutin/bord touché**
- **couleur touchée**

3 : Le labyrinthe

Les concepts introduits

- La création d'un jeu
- La détection de collisions
- Les boucles
- Les instructions conditionnelles

Les commandes de Scratch introduites

- Mouvement
    - Rebondir si le bord est atteint

Seule une nouvelle commande est introduite dans cette leçon, car le but est de consolider la maîtrise des commandes introduites précédemment en les utilisant dans le nouveau contexte de la création d'un jeu.

La pensée informatique et ses thématiques

- L'abstraction
    - la répétition
    - la position : les coordonnées x & y
- Les algorithmes
- La décomposition
    - décomposition du jeu en ses principaux éléments
- La reconnaissance de formes
    - utilisation d'un code similaire pour les déplacements dans différentes directions

Les objectifs

L'élève devrait être capable :

- de comprendre l'utilité d'un algorithme pour concevoir une solution à un problème
- de passer de l'algorithme au code avec plus de facilité
- d'utiliser des instructions conditionnelles

Les ressources

- Captures vidéos : **Maze**

Introduction

Montrer aux élèves la capture vidéo « **Maze** » ou leur permettre de la regarder seul(e).

Une autre approche pourrait être de faire une démonstration du jeu aux élèves, afin qu'ils essaient ensuite de concevoir un algorithme adapté.

Tâche nº 1 : Planter le décor

Demander aux élèves de créer un nouveau projet et d'importer un arrière-plan de type Labyrinthe (**Maze**). Ils peuvent aussi créer leur propre labyrinthe en guise d'arrière-plan en utilisant les outils graphiques de Scratch.

Note : Si la deuxième option est choisie, veiller à ce que les labyrinthes des élèves soient suffisamment larges pour accueillir des lutins, et à ce qu'il soit effectivement possible d'y déambuler !

De l'importance de la conception

Avant de réaliser quoi que ce soit – une maison, une robe ou un programme informatique –, il faut commencer par sa **conception**. La plupart des programmes étant constitués de deux principaux éléments – l'**interface** et le **code** –, ces deux parties sont conçues séparément.

- Le plus simple pour concevoir une **interface** est d'en faire un croquis sur une feuille de papier.
- La façon la plus habituelle de concevoir le **code** est d'écrire **en français** la liste des étapes à exécuter. C'est ce que l'on appelle un **algorithme**.

Écrire un algorithme est la clé d'une programmation réussie. En fait, c'est l'essence même de la programmation. Il s'agit de résoudre des problèmes, et non de taper des commandes sur ordinateur.

Tout bon programmeur conçoit des algorithmes avant de commencer à coder.

- Discuter de l'encart ci-dessus et introduire le concept de l'**algorithme** – une liste d'étapes visant à résoudre un problème (habituellement rédigé en français).
- Faire prendre conscience aux élèves qu'ils sont en train d'apprendre à « penser comme un ordinateur ».

Tâche nº 2 : Concevoir la solution

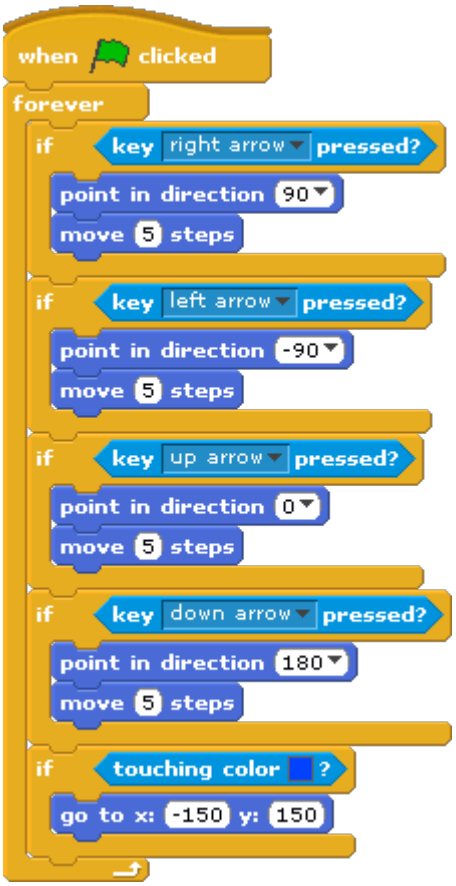
Souligner le fait qu'il y a deux principaux éléments que nous devons coder pour notre jeu :

1. faire avancer l'explorateur
2. atteindre le centre du labyrinthe (et sauver l'ami de l'explorateur)

En décomposant le problème, il est possible de résoudre chaque partie séparément. Il s'agit là d'une bonne occasion d'expliquer pourquoi il est utile et important de décomposer un problème en éléments plus simples :

- la résolution d'un problème complexe est facilitée par sa décomposition en problèmes plus simples que l'on peut résoudre individuellement.
- les programmeurs professionnels travaillent souvent en équipe sur la réalisation des programmes. Décomposer un problème permet aux différents membres de l'équipe de travailler sur différentes parties du programme. Ces parties peuvent ensuite être combinées pour créer le programme final.

S'attarder sur le tableau montrant un **algorithme** permettant de déplacer l'explorateur et le **code** correspondant en Scratch. Souligner le fait que l'algorithme et le code sont deux représentations d'une seule et même chose, mais à différents niveaux (**abstraction**).

Algorithme pour faire avancer l'explorateur	Code
lorsque l'utilisateur clique sur le drapeau répéter indéfiniment si l'utilisateur appuie sur la touche de déplacement vers la droite pointer vers la droite avancer de 5 pas si l'utilisateur appuie sur la touche de déplacement vers la gauche pointer vers la gauche avancer de 5 pas si l'utilisateur appuie sur la touche de déplacement vers le haut pointer vers le haut avancer de 5 pas si l'utilisateur appuie sur la touche de déplacement vers le bas pointer vers le bas avancer de 5 pas si l'explorateur touche la même couleur que celle du mur du labyrinthe revenir au point de départ	

Souligner la présence d'indentations dans les algorithmes, utilisées pour faire ressortir la structure.

Activité supplémentaire nº 1 : Quand la musique est bonne

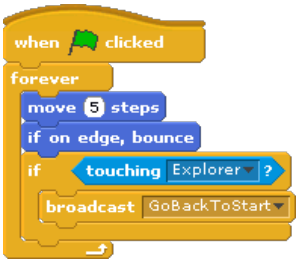
Quel serait le meilleur endroit pour stocker ce son, dans la mesure où il s'applique à l'ensemble du jeu ? **La scène.**

Comment vas-tu faire pour que la musique joue sans interruption ? **Il faut placer le son dans une boucle** répéter indéfiniment.

Faudrait-il utiliser un bloc jouer le son ou jouer le son complètement pour jouer le morceau ? **Il faut utiliser le bloc** jouer le son complètement**, pour que l'intégralité du morceau de musique soit répétée plutôt qu'uniquement la première note.**

Activité supplémentaire nº 2 : Ajouter un ennemi

Le code pour cet ajout est relativement simple si la collision avec l'ennemi apparaît dans le script de l'explorateur.



Si les élèves le placent dans le script de l'ennemi, ils devront envoyer un événement au lutin explorateur pour réinitialiser sa position (et inclure l'événement quand je reçois correspondant dans le script de l'explorateur pour que ce dernier revienne à sa position de départ).

Ce genre de modularité peut être justifiée et pourra faire l'objet d'une discussion avec les élèves.

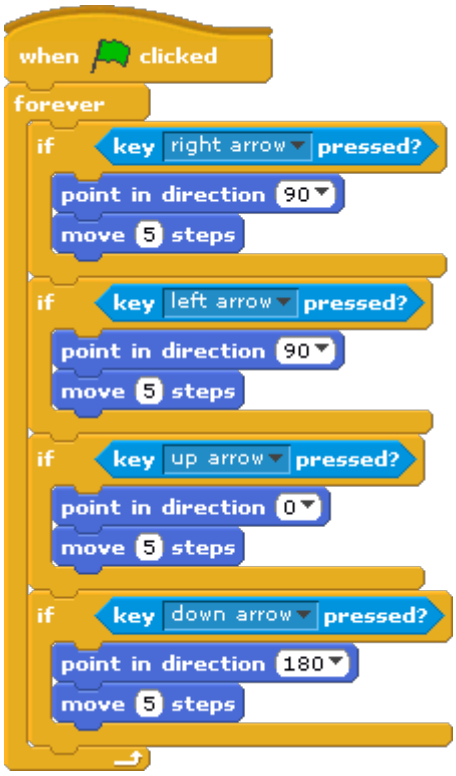
Activité supplémentaire nº 3 : Un jeu à deux joueurs

Bien que cette activité ne soit pas incluse dans le livret de l'élève, les enseignants seront peut-être intéressés par l'idée de proposer aux élèves de créer une version à deux joueurs du jeu.

Dans un tel jeu, différents groupes de touches aux extrémités du clavier pourraient être utilisés pour contrôler deux lutins distincts faisant la course vers le centre depuis deux extrémités opposées du labyrinthe (qui aura idéalement été créé par l'élève dans ce but).

As-tu compris ?

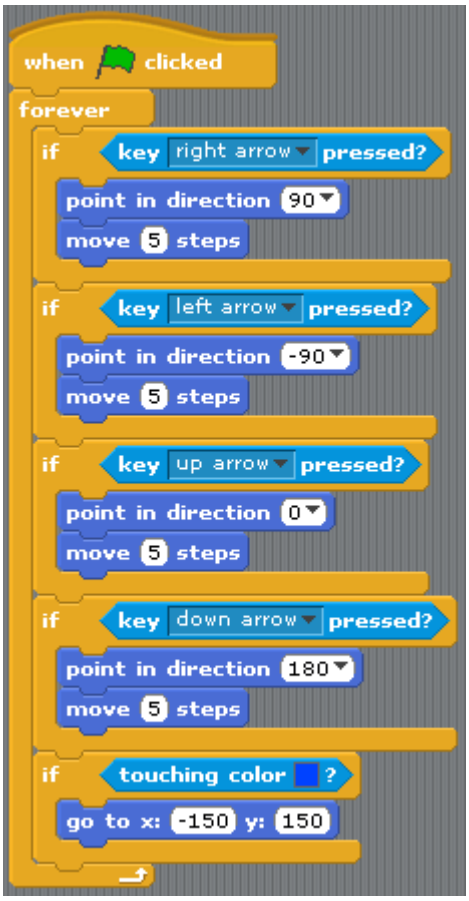
3.1 Un programmeur crée un jeu de labyrinthe semblable à celui que tu viens de créer. Malheureusement, son personnage ne se déplace pas comme prévu.

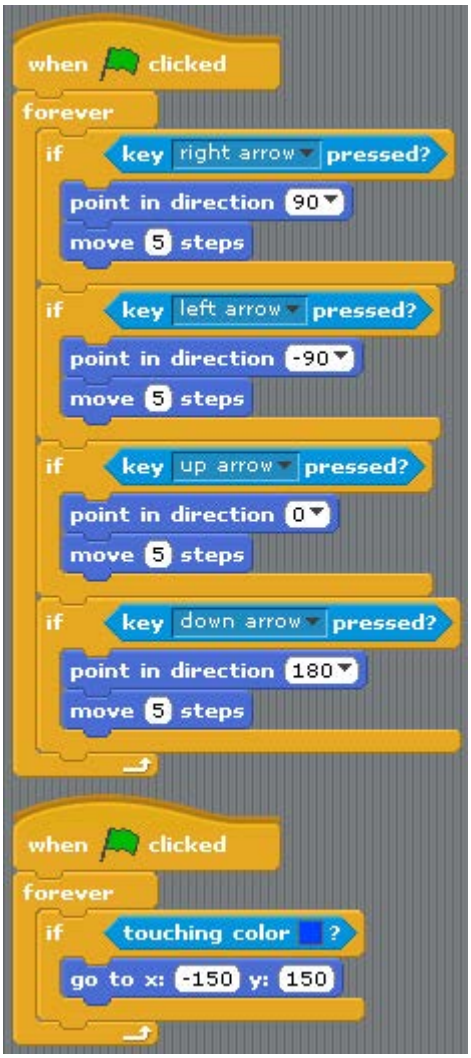


Quelle erreur a été commise ?

Les flèches de gauche et de droite pointent toutes deux vers la droite (90). La flèche de gauche devrait pointer vers la gauche (-90).

3.2 Étudie les exemples de code ci-dessous.





Accomplissent-ils la même tâche ? **Oui.**

Explique ta réponse.

Les piles de blocs liées aux instructions répéter indéfiniment et si couleur touchée ? sont exécutées tout au long du programme, même si les piles sont séparées dans l'exemple de droite.

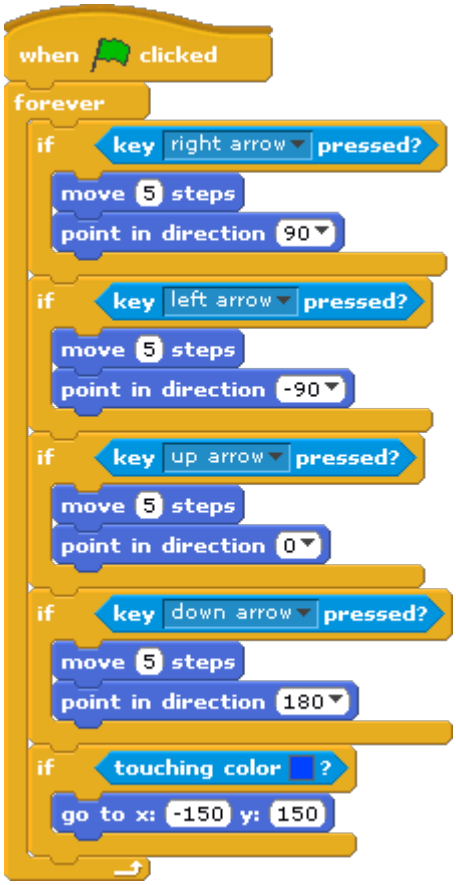
Cela pourrait offrir une bonne occasion de reparler des concepts de parallélisme et de modularité :

Q : Pourquoi pourrait-on penser que le script de droite est mieux que celui de gauche ?
R : Il sépare le mouvement et les collisions en deux procédures distinctes. Or, il

s'agit de deux problèmes distincts qu'il serait logique de résoudre séparément. Cela rendra le code plus simple à maintenir si de nouvelles fonctionnalités sont ajoutées en lien avec l'une ou l'autre de ces problématiques.

3.3 Le code ci-dessous contrôle un lutin se déplaçant dans le labyrinthe. Si le lutin touche le mur du labyrinthe (de couleur bleue), il retourne au point de départ en -150, 150.

Malheureusement, le lutin touche parfois les murs du labyrinthe et retourne au point de départ alors que le joueur ne s'y attend pas.

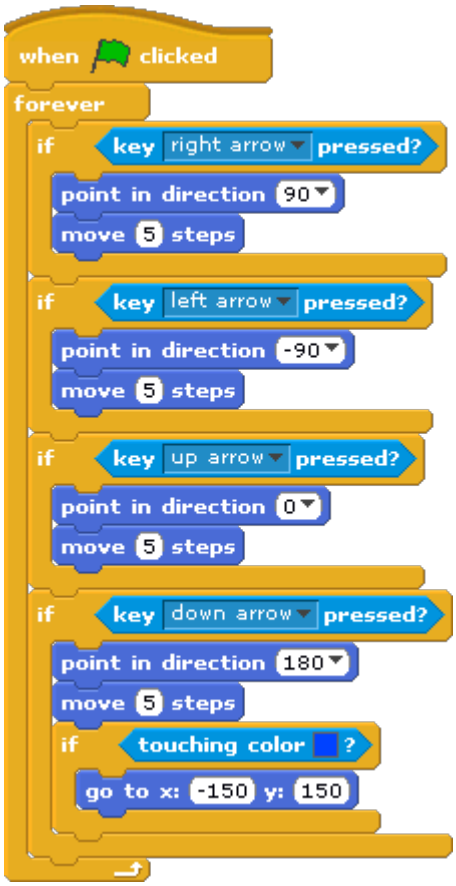


Quelle erreur le programmeur a-t-il fait ?

Le lutin bouge avant que la bonne direction soit indiquée.

Cela va le conduire à avancer de 5 pas dans la direction précédemment sélectionnée et à toucher la couleur bleue si celle-ci se trouve à proximité.

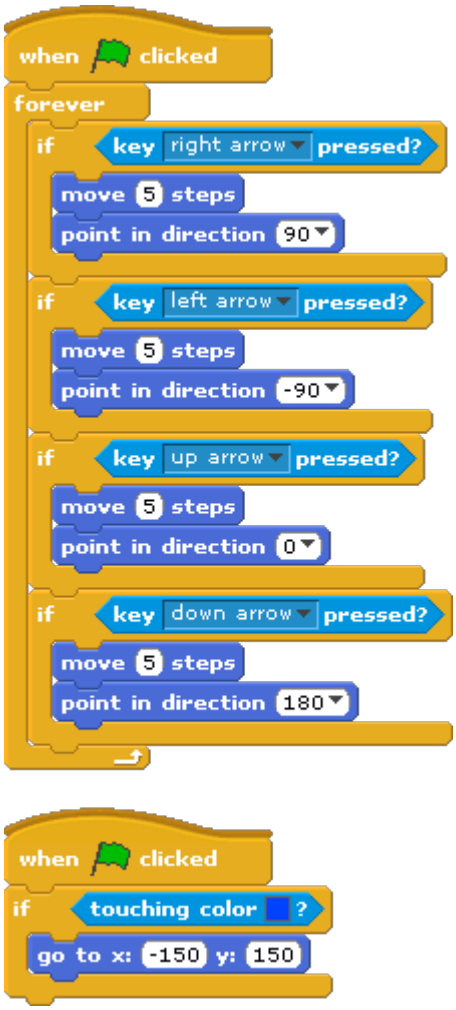
3.4 Dans cet exemple, le lutin est censé retourner à l'entrée du labyrinthe lorsqu'il en touche les murs (de couleur bleue). Toutefois, ce n'est pas toujours ce qui se produit.



Quelle erreur le programmeur a-t-il fait ?

Le bloc couleur touchée ? est à l'intérieur de la dernière instruction « si », si bien qu'il ne pourra être exécuté que si l'utilisateur appuie sur la touche de déplacement vers le bas.

3.5 Dans cet exemple, le lutin ne revient **jamais** au point de départ, même s'il touche les murs du labyrinthe (de couleur bleue).



Quelle erreur le programmeur a-t-il fait ?

Le bloc si couleur touchée ? n'est exécuté qu'au tout début du jeu, c'est à dire au moment où l'on clique sur le drapeau (une erreur courante).

Il doit donc être placé à l'intérieur d'une boucle infinie.

3.6 À présent, prépare une autre question de débogage et pose-la à ton voisin.

4 : Une question d'image…

Les concepts introduits

- La décomposition d'un problème et la modularisation à l'aide de procédures
- L'élément chronomètre/horloge

Les commandes de Scratch introduites

- Mouvement
  - aller à <emplacement ou objet>
  - pointer en direction <n>
- Contrôle
  - envoyer à tous <message> et attendre
- Stylo
  - effacer tout
  - relever le stylo, abaisser le stylo
  - mettre la couleur du stylo à <couleur>
  - modifier la couleur du stylo par <n>
  - mettre la taille du stylo à <n>
  - modifier la taille du stylo par <n>

La pensée informatique et ses thématiques

- L'abstraction
  - la répétition, y compris avec boucle « répéter » imbriquée
  - la position : les coordonnées x & y
- Les algorithmes
- La décomposition
  - la création de formes complexes à partir de formes simples
- La reconnaissance de formes
  - la règle de la rotation
  - l'utilisation de l'itération pour la réalisation de dessins complexes
- Généralisation
  - la règle de la rotation

Les objectifs

L'élève devrait être capable :

- de comprendre l'utilité d'un algorithme pour concevoir une solution à un problème
- de passer de l'algorithme au code avec plus de facilité
- de manipuler les variables plus facilement
- d'utiliser des procédures afin de créer un programme plus modulaire

Les ressources

- Capture vidéo : **Graphics**

Tâche nº 1 : Remise en forme

- Montrer aux élèves la capture vidéo « **Graphics** » ou leur permettre de la regarder seul(e).

  Encourager les élèves à poser sur papier les étapes nécessaires à la création de l'heptagone et du triangle. Il pourrait être utile pour les élèves de travailler en binôme ; un élève pourrait donner des instructions à son coéquipier pour qu'il dessine les formes sur papier, ce qui les encouragerait à « **penser comme un ordinateur** ».

AID **Les mathématiques : certains élèves n'arriveront pas à faire le triangle, car ils appliqueront une rotation de 60° (angle interne) plutôt qu'une rotation de 120° (angle externe).**

- Avant de passer à la suite, demander aux élèves d'essayer de trouver la règle de la rotation en se basant sur ce qu'ils ont déjà fait.

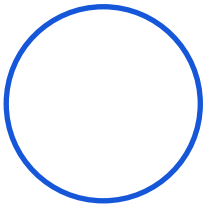
Tâche nº 2 : Une étoile est née !

- La règle de la rotation fonctionnera ici, mais les élèves devront réaliser qu'ils font **deux fois** un tour complet, si bien qu'il faut tourner de 144° à chaque fois.

  Les faire travailler en binôme avec un élève indiquant à l'autre ce qu'il doit dessiner (ou bien concevoir l'algorithme avec l'ensemble de la classe) pourrait les aider à bien comprendre ceci.

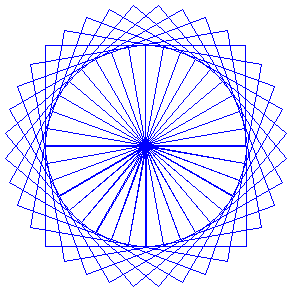
Tâche nº 3 : Le cercle

- Les élèves réaliseront peut-être qu'il ne s'agit pas d'un véritable cercle, mais d'un polygone à 36 côtés[7].
- Il est à noter qu'un algorithme/pseudo-code est proposé ici pour la première fois.



Tâche nº 4 : Motif circulaire

- Encourager les élèves à faire des tests au cours de cette tâche, en modifiant la forme à reproduire et la couleur du stylo.
- Cette activité rappellera peut-être à certains élèves le jeu du spirographe™ qui permet de tracer des formes géométriques.



[7] Un triacontakaihexagone.

AID **Il est évident que les mathématiques sont très présentes dans cette activité. Il pourrait être intéressant pour les enseignants de faire le lien avec les mathématiques déjà vues par les élèves, ou d'en parler à l'avance avec leur professeur de mathématiques.**

Imbrication

Dans la tâche n° 4, nous avons utilisé une boucle « **répéter** » à l'intérieur d'une autre ; c'est ce qu'on appelle une boucle **imbriquée**.

En effet, le programme lance la boucle « **répéter** » externe, puis entre dans la boucle de répétition interne, qui s'exécute jusqu'au bout. Puis, l'exécution de la boucle « **répéter** » externe se poursuit et ainsi de suite.

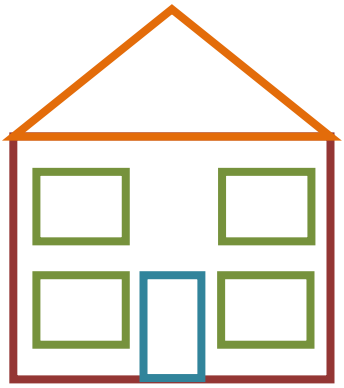
Démarre le mode **pas-à-pas** (menu d'édition) pour voir ce qu'il se passe au ralenti. Pense à désactiver le mode pas-à-pas lorsque tu auras terminé.

- Les enseignants voudront peut-être discuter de l'encart ci-dessus avec les élèves en faisant la démonstration de la tâche n° 4 en mode pas-à-pas (**Edition→Démarrer le pas-à-pas)**. Dans ce cas, rappeler aux élèves de désactiver le mode pas-à-pas lorsqu'ils n'en auront plus besoin.

Activité supplémentaire nº 1 : L'événement principal

Si la **tâche n° 4 « Motif circulaire »** est codée en utilisant le bloc **envoyer à tous** plutôt que le bloc **envoyer à tous et attendre**, on n'obtiendra pas le résultat voulu, car la boucle externe commencera l'itération suivante avant que la boucle interne ait terminé. Ceci est un bon moyen de montrer la différence entre ces deux concepts. C'est aussi la base de la question Q4.4 de l'activité « As-tu compris ? » plus loin dans cette leçon.

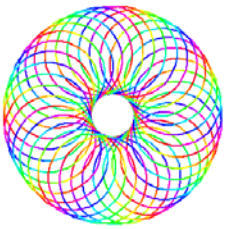
Activité supplémentaire nº 2 : Notre maison



Il sera avantageux pour les élèves d'utiliser du papier quadrillé pour dessiner le plan de leur maison. L'utilisation d'un triangle équilatéral pour le toit est aussi recommandée.

Encourager les élèves à créer des fenêtres à l'aide des blocs **envoyer à tous** et **quand je reçois**.

Activité supplémentaire nº 3 : Oh, les beaux anneaux !



Les élèves avanceront probablement par tâtonnement, car selon la taille choisie pour le cercle de base il est possible que le stylo touche le bord de l'écran, ce qui entraînera une déformation de l'image.

Activité supplémentaire nº 4 : Les anneaux olympiques[8]

Il s'agit d'un exercice difficile à réaliser même pour les élèves les plus à l'aise ! Voici quelques points sur lesquels ils devraient se concentrer lors de la conception de l'algorithme :

- Planifier sur papier quadriller et noter la distance entre les centres des cercles.
- Définir la taille et la couleur du stylo au tout début
- Utiliser les blocs **relever le stylo** et **abaisser le stylo**
- Modifier la couleur du stylo entre chaque cercle

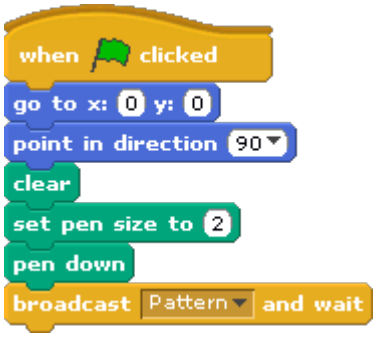
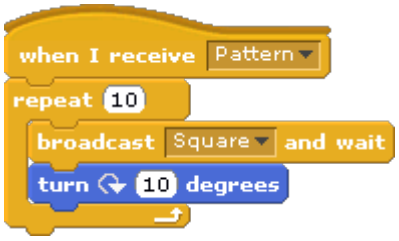
Préciser aux élèves qu'il n'est pas nécessaire que les anneaux se chevauchent comme c'est le cas dans le logo officiel.

[8] Le symbole olympique aux cinq anneaux est reproduit avec l'aimable autorisation du Comité International Olympique (CIO). Les anneaux olympiques sont la propriété exclusive du CIO. À ce titre, ils sont protégés à l'international par des marques ou des législations nationales et ne peuvent être utilisés sans accord écrit préalable du CIO.

As-tu compris ?

4.1 Étudie le programme ci-dessous.

Indique l'ordre dans lequel les procédures sont exécutées une fois que l'utilisateur a cliqué sur le drapeau vert (numérote-les de 1 à 3).

Numéro	Procédure
3	
1	
2	

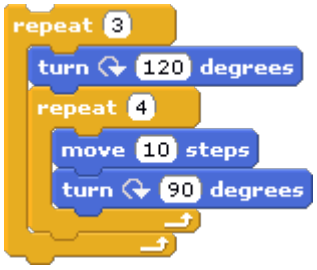
Maintenant, décris ce que va faire le code.

Procédure n° 1 : Définit le point de départ du lutin, libère la scène, définit la taille du stylo et abaisse ce dernier. Envoie ensuite à tous l'événement « Motif » et attend la fin des tâches.

Procédure n° 2 : Envoie à tous l'événement « Carré » et attends que les tâches soient terminées. Il tourne ensuite de 10 degrés. Ce processus est réitéré 36 fois.

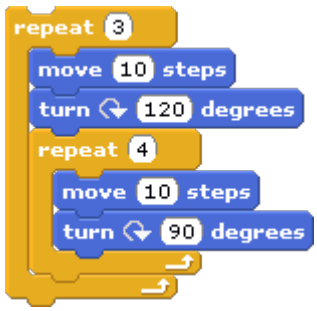
Procédure n° 3 : Dessine un carré dont chaque côté fait 100 pas de long.

4.2 Étudie les exemples de code ci-dessous.



a) Combien de fois le lutin avancera-t-il de 10 pas ?
12

Pourquoi ? **La boucle externe fait que les 4 passages de la boucle interne sont exécutés 3 fois, donc 3 x 4 = 12.**



b) Combien de fois le lutin avancera-t-il de 10 pas ?
15

Pourquoi ? **12 fois dans la boucle interne + 3 fois dans la boucle externe.**

4.3 Considère les exemples suivants issus du quotidien. Écris un « algorithme » pour chacun d'eux ! _____

a) Se préparer pour l'école

Se réveiller, prendre son petit déjeuner, se laver, s'habiller, ramasser ses affaires, quitter la maison

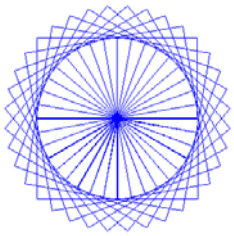
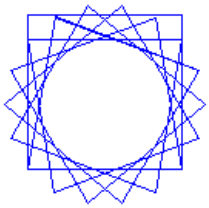
b) Préparer le petit déjeuner

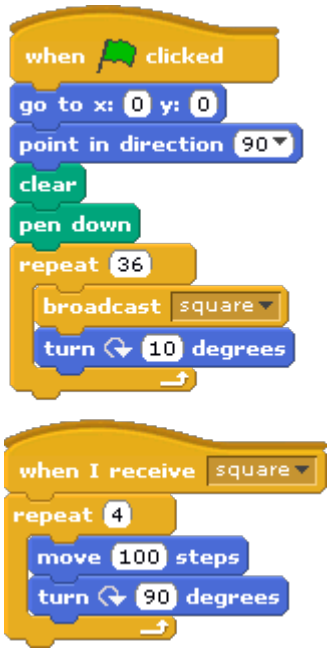
Prendre un bol, prendre le lait, verser les céréales dans le bol ;
Mettre la bouilloire en route, mettre du pain dans le grille-pain, mettre un sachet de thé dans la théière, etc.

Il est à noter qu'il s'agit là d'une des étapes de la partie a), ce qui montre qu'une décomposition successive est possible. Il est donc possible de décomposer une procédure en plusieurs procédures, etc.

Il serait même possible d'utiliser le cas du thé et du pain grillé pour aborder la question du traitement en parallèle. Par exemple, pendant que l'on attend que le grille-pain éjecte le pain ou que l'eau bout dans la bouilloire, nous allons chercher les autres ingrédients. Ainsi, nous pourrions avoir dans Scratch des procédures séparées pour ces étapes exécutées simultanément.

4.4 Un programmeur tente de créer un motif circulaire à partir de carrés, comme celui marqué « Exact » ci-dessous. Malheureusement, au lieu de fournir le résultat voulu, le programme affiche le motif marqué « Inexact ».

Exact	Inexact
	



Étudie le code du programmeur ci-contre. Quelle erreur a-t-il commis ?
Astuce : la réponse est en lien avec la vitesse d'exécution de l'ordinateur.

Il s'agit là d'une question difficile qui posera sans doute problème aux élèves.

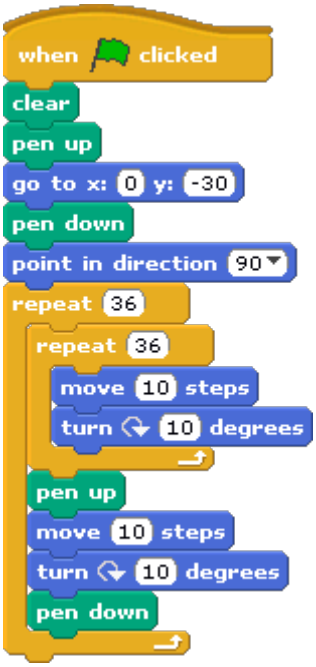
Pour dessiner le carré, le programmeur aurait dû utiliser un bloc envoyer à tous et attendre plutôt qu'un simple envoyer à tous.

La procédure du haut passe à l'instruction tourner de 10 degrés avant que l'instruction quand je reçois « carré » n'ait été entièrement exécutée.

4.5 À présent, prépare une autre question de débogage et pose-la à ton voisin.

As-tu compris ? (ne concerne que l'activité supplémentaire n° 3)

4.6 Un programmeur essaie de dessiner un anneau tel que celui de l'activité supplémentaire n° 3. Malheureusement, son programme ne fait que dessiner des cercles les uns par-dessus les autres.



Quelle erreur a-t-il commis ?

En avançant de 10 pas et en tournant de 10 degrés à la fin de chaque cercle, il ne fait qu'avancer un peu le long du cercle avant de redessiner un autre cercle à partir de là où il s'arrête.

- **Il faut qu'il modifie son programme pour à chaque fois faire 10 pas dans le sens inverse des aiguilles d'une montre dans la boucle externe, afin d'être sûr que le nouveau cercle démarre à un point de départ différent ou**
- **avancer de plus de dix pas entre chaque cercle.**

Il s'agit d'une question très difficile. Les élèves trouveront peut-être utile de reproduire le code dans Scratch et de l'exécuter en mode pas-à-pas pour voir par eux-mêmes ce qu'il se passe.

5 : Tir à l'arc en forêt

Les concepts introduits

- La création d'un jeu
- La détection de collisions
- Les boucles
- Les instructions conditionnelles
- Les variables
- Les nombres aléatoires

La pensée informatique et ses thématiques

- L'abstraction
  - la répétition
  - la position : les coordonnées x & y
- Les algorithmes
- La décomposition
  - décomposition du jeu en ses principaux éléments
- La généralisation
  - utilisation d'une variable pour stocker le score et le temps

Les commandes de Scratch introduites

- Mouvement
  - glisser en <n> secondes à x: <n> y:<n>
  - pointer en direction <n>
- Contrôle
  - quand lutin <nom du lutin> pressé
  - arrêter tout
- Capteurs
  - souris x, souris y
  - <nom du lutin> touché
- Variables
  - à <nom de variable> attribuer <n>
  - à <nom de variable> ajouter <n>

Les objectifs

L'élève devrait être capable :

- de comprendre l'utilité d'un algorithme pour concevoir une solution à un problème
- de passer de l'algorithme au code avec plus de facilité
- d'utiliser des instructions conditionnelles

Les ressources

- Capture vidéo : **ForestArchery**

Introduction

Montrer aux élèves la capture vidéo « **ForestArchery** » ou leur permettre de la regarder seul(e).

Une autre approche pourrait être de faire une démonstration du jeu aux élèves, afin qu'ils essaient ensuite de concevoir un algorithme adapté.

Tâche nº 1 : Concevoir la solution

Encourager les élèves à créer leur code à partir des algorithmes ci-après, plutôt qu'en regardant la capture vidéo une nouvelle fois.

Indiquer aux élèves qu'il faut à nouveau utiliser une **imbrication** – mais cette fois en plaçant une instruction si à l'intérieur d'une autre, plutôt qu'une boucle « répéter ».

Dans le cas présent, la condition **si le lutin a touché le lutin cible**

n'est exécutée que **si l'utilisateur a cliqué avec la souris**.

Tâche nº 2 : Toucher et manquer

Cette activité introduit l'instruction conditionnelle *si...sinon*. C'est à dire que si quelque chose est vrai, faire *ceci*... sinon, faire *cela*.

Inviter les élèves à trouver d'autres exemples de *si...sinon* dans la vraie vie.
Ex. des feux de signalisation – si vert, avancer, sinon s'arrêter.

Tâche nº 3 : Contre la montre

Cette activité introduit la **variable** chronomètre dans les programmes des élèves. Il faut que la variable apparaisse à l'écran. Elle doit donc avoir un **nom qui a du sens** et être **cochée** dans la zone des blocs.

Discuter avec les élèves des différentes approches possibles, comme l'utilisation de la boucle **répéter jusqu'à** *chronomètre = 0*, ou l'utilisation d'un bloc **répéter indéfiniment** dont le désavantage est que le chronomètre finit par prendre une valeur négative.

Tâche nº 4 : En plein dans le mille !

Pour les élèves, la façon la plus simple de coder ceci est d'utiliser beaucoup d'instructions « **si** » (non imbriquées). Bien que cette approche ne soit pas particulièrement efficace – il faut analyser chaque instruction « **si** » même si on en a déjà exécuté une avec succès –, à ce stade, c'est une approche plus facile à comprendre pour les élèves, comparée à l'utilisation de nombreuses instructions « **si** » imbriquées.

Pour l'instant, Scratch n'offre pas la possibilité d'utiliser une instruction **de cas**.

Tâche nº 5 : Restons positifs !

Pour cette tâche, il suffit que les élèves ajoutent – au bon endroit – une condition qui vérifie **si score > 0** lorsque l'on retranche un point pour une cible manquée.

Activité suggérée

Faire découvrir Archery Champion beta 1.0 aux élèves. Il s'agit d'un jeu de tir à l'arc élégant et de qualité professionnelle créé dans Scratch : http://scratch.mit.edu/projects/Shanesta/9710.

Les variables

L'utilisation de variables est un concept fondamental de la pensée informatique (abstraction/généralisation) qu'il est important que les élèves comprennent.

On peut considérer qu'une variable est une partie de la mémoire d'un ordinateur dans laquelle on stocke une valeur. On lui donne ensuite un nom, comme on mettrait une étiquette sur une boîte, pour se rappeler ce qui y est stocké.

Activité suggérée

Distribuer des enveloppes sur lesquelles ont été écrits des noms tels que « âge », « couleur préférée », « animal de compagnie », etc. Chaque enveloppe doit contenir un morceau de papier sur lequel une valeur a été écrite. Remplacer le morceau de papier présent dans chaque enveloppe par un autre comportant une autre valeur pour illustrer l'attribution d'une nouvelle valeur à la variable.
Utiliser cette activité pour aborder l'importance de l'attribution à chaque variable d'un nom qui a du sens pour pouvoir les reconnaître facilement dans le programme.
Question à poser aux élèves :

Q : Pourquoi stocker des informations dans des variables ? Est-ce que ça ne complique pas les choses ?

R : Non ! Grâce aux variables, il devient **plus simple** pour nous d'utiliser et de modifier les informations d'un programme. Expliquer comment les variables nous permettent de **généraliser** notre code.

Par exemple :

- Prenons le cas d'un jeu affichant un score qui change au cours du jeu.
  Devrait-on avoir une ligne de programme séparée pour chaque score possible ? Non. Nous aurions une seule ligne de programme utilisant une variable. Utiliser une variable est la seule véritable façon de stocker cette information.

Activité supplémentaire nº 1 : Top chrono !

Si les élèves n'ont plus leur propre programme de labyrinthe, ils pourront utiliser le modèle **Maze1**.
Demander aux élèves où il serait préférable de placer ce script (la scène) et pourquoi (il affecte l'intégralité du programme).

Activité supplémentaire nº 2 :

Créer leur propre labyrinthe en utilisant les outils graphiques de Scratch devrait amuser les élèves.

Il est toutefois important de veiller à ce qu'il soit facile de parcourir le labyrinthe. Il est donc possible que les élèves doivent réduire la taille de leur lutin explorateur.

Pour la création des lutins bonus, le plus simple est que chaque bonus soit programmé de façon à détecter le fait de toucher l'explorateur (plutôt que d'avoir un script complexe pour l'explorateur). Les scripts pourraient inclure des instructions servant à faire apparaître et disparaître les bonus de façon aléatoire.

Activité supplémentaire nº 3 : Et ma récompense ?

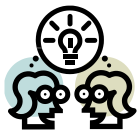
Rappeler aux élèves que pour faire apparaître une variable à l'écran, il suffit de cocher la case à côté de la variable. Cliquer sur la variable affichée sur la scène fera apparaître soit sa **valeur**, soit sa **valeur et son nom**.

Activité supplémentaire nº 4 : C'est tout vu…

Il s'agit là d'une amélioration relativement simple impliquant l'utilisation de blocs montrer/cacher. Une amélioration supplémentaire pourrait être de faire réapparaître des lutins bonus en différents endroits. Il faudrait alors vérifier que le lutin se trouve bien dans le labyrinthe en utilisant l'instruction si couleur touchée.

As-tu compris ?

5.1 Étudie le script ci-dessous qui doit permettre à une variable chronomètre de compter à rebours de 30 à 0.



Va-t-il fonctionner ? **Non**.

Explique ta réponse. **On rajoute +1 à la variable *temps* chaque fois que l'on complète la boucle (la variable augmente donc). Or, il faut retrancher 1 de la variable (c.-à-d. ajouter -1).**

5.2 À présent, prépare une autre question de débogage et pose-la à ton voisin.

Un projet avec Scratch

Analyse

- Encourager les élèves à réfléchir à la façon dont leur projet pourrait être mis en rapport avec d'autres matières qu'ils étudient. Afin qu'ils restent concentrés, ne pas accorder plus de 15 minutes à la visite de la galerie de Scratch.

Maintenant, discute de tes idées avec ton professeur.
Une fois que vous vous êtes mis d'accord sur ton projet, décris ci-dessous ce qu'il fera.

- L'essentiel ici est de veiller à ce que le projet soit réalisable par les élèves concernés.

Conception (interface)

- Pour cette étape, un ou deux simples croquis annotés devraient suffire. Le but est de forcer les élèves à bien réfléchir à leur projet et à la façon dont fonctionnera le programme.

Conception (code)

- Rappeler aux élèves qu'ils doivent prendre le temps de bien rédiger leurs algorithmes avant de coder. Se souvenir du vieux proverbe : *quelques heures de codage pour t'épargner quelques minutes de conception !*

Mise en œuvre

- Rappeler aux élèves l'importance d'utiliser des **identifiants qui ont du sens** pour les noms des lutins, des costumes, des sons et des variables.
- Insister sur le fait qu'il faut que les élèves travaillent à partir d'algorithmes.

Tests

- Les élèves devront tester leurs propres projets et les faire tester par leurs camarades de classe. Demander aux élèves de décrire les erreurs de programmation trouvées et d'expliquer comment elles ont été résolues (ou non).

Documentation

- Les élèves devront montrer qu'ils ont réfléchi à la rédaction d'une description brève et percutante de leur projet et de ses principales fonctionnalités.

Évaluation

- Encourager les élèves à discuter de leur travail et à l'évaluer en toute honnêteté.
- Encourager les élèves à revoir leur code pour tenter de le rendre plus élégant (se reporter à l'encart « Paresseux ou astucieux ? »)

Maintenance

- Suggérer aux élèves la maxime suivante :
  « Un programme informatique n'est jamais fini – nous cessons simplement de l'améliorer. »

Félicitations !

Encourager les élèves à continuer à programmer chez eux avec Scratch.

Souligner le fait qu'ils n'ont fait que goûter aux possibilités qu'offre Scratch et que le logiciel propose bien d'autres fonctions.

Annexes

Annexe A : Fiche de suivi de l'élève

Nom : _____ Classe : _____

Étape	Progrès * (A, C ou I)	Date de fin	Commentaire
Introduction			
Leçons			
1 : Les bases de Scratch			
2 : C'est l'heure d'une histoire			
3 : Le labyrinthe			
4 : Une question d'image...			
5 : Tir à l'arc en forêt			
Projet			
Analyse			
Conception			
Mise en œuvre			
Tests			
Documentation			
Évaluation			
Maintenance			

PROGRÈS *

Acquisition L'élève travaille à l'acquisition de compétences ou de connaissances.

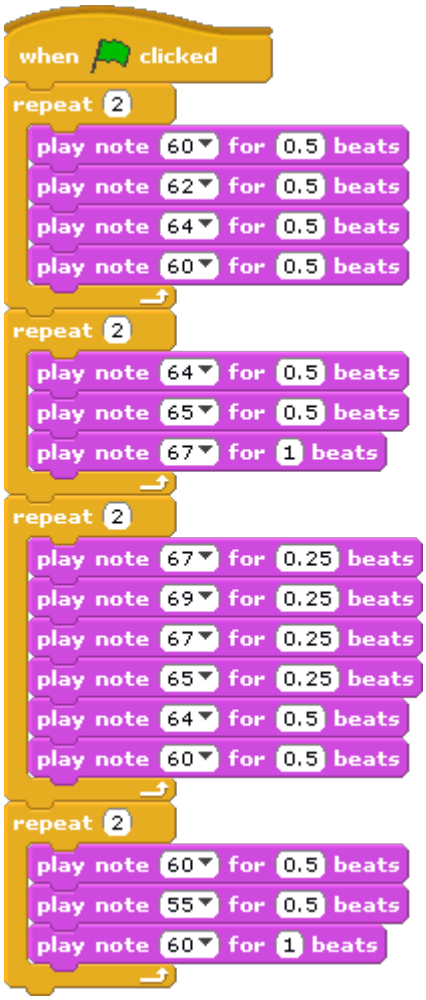
Consolidation L'élève gagne en maîtrise et en confiance quant à l'utilisation de ses compétences ou connaissances.

Intégré L'élève est en mesure d'appliquer avec assurance ses compétences ou connaissances dans des situations plus complexes ou nouvelles.

Annexe B : Échantillons de code

Leçon nº 1 : Premiers pas

Leçon nº 1 : Frère Jacques



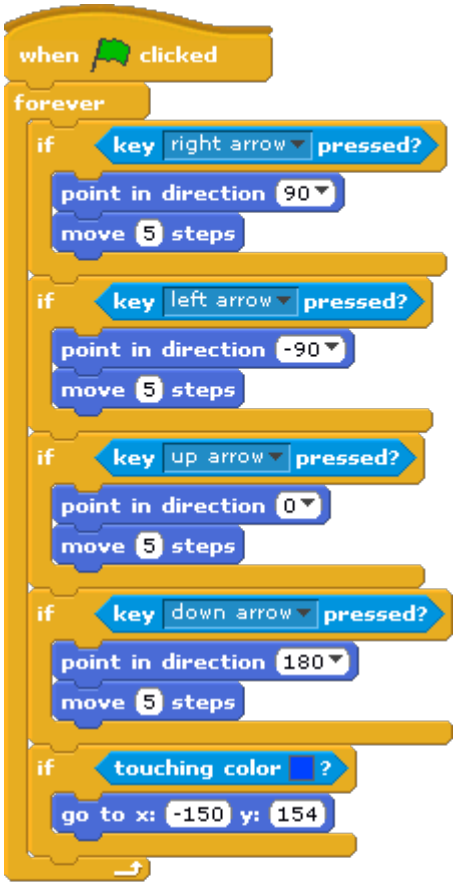
Leçon nº 2 : Une mauvaise blague

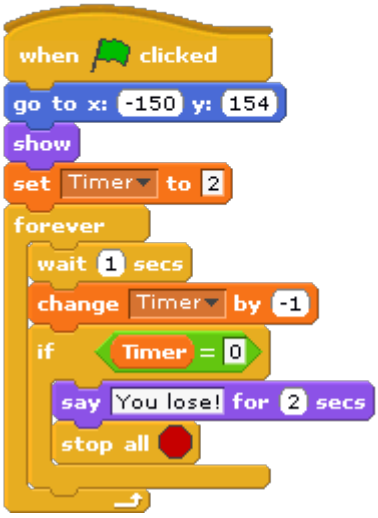
Lutin fille

Lutin garçon

Leçon nº 3 : Le labyrinthe (et les activités supplémentaires associées)

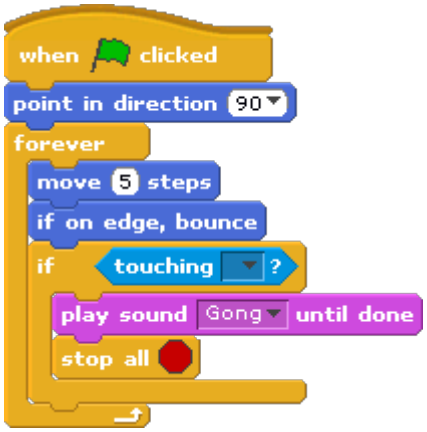
Lutin explorateur (chat)





Lutin ami (2e chat)

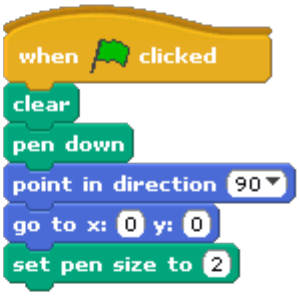
Lutin ennemi (dragon)

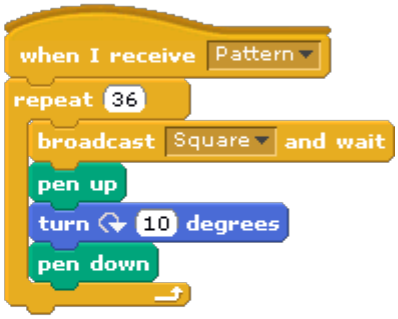


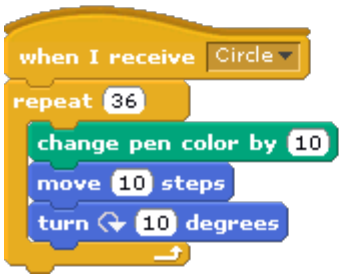
Scène

Leçon nº 4 : Une question d'image…

Échantillons de scripts utilisés pour dessiner des formes dans cette leçon :







Leçon nº 5 : Tir à l'arc en forêt

Scène

Viseur

Cible