Activités de découverte de l'algorithmique et de Scratch

Début de cycle 4 : les différentes structures algorithmiques, comment et pourquoi les utiliser dans un projet Scratch

Certaines de ces activités sont inspirées du document Ressources où elle sont proposées sous une autre forme.

Présentation des activités

Ces 6 activités sont organisées autour des différentes structures algorithmiques : séquence, instruction conditionnelle, boucle, événement, variable, boucle conditionnelle.

Cependant, elles sont aussi l'occasion de découvrir les différents éléments de Scratch : arrière-plan, lutins, blocs d'instructions, costumes...

L'ordre dans lequel elles sont données est conseillé mais n'est pas obligatoire, à condition de s'assurer que les prérequis nécessaires à chaque activité ont été traités. Le traitement du programme d'algorithmique du collège ne saurait bien entendu se résumer à ces six séances, il faudra en plus prévoir des séances pour réinvestir les notions déjà présentées, les réactiver à plus long terme, et les intégrer dans un projet de plus grande envergure.

Toutes ces activités peuvent être facilement prolongées pour les élèves les plus rapides : en leur donnant des scénarios plus complexes à résoudre, en permettant la découverte d'une autre instruction, ou en leur faisant construire leurs propres scénarios.

Déroulement des activités :

Il s'agit pour toute la classe de manipuler Scratch et de se faire une première idée de ce qu'est un algorithme (en tant qu'ensemble structuré d'instructions amenant à une solution).

Dans un premier temps, on peut montrer et expliquer au tableau les différents éléments de Scratch :

- le fond qui est fixe mais peut avoir une influence sur l'exécution du script
- le lutin mobile
- les instructions disponibles, avec un bref aperçu des différentes catégories
- les scripts du lutin, les différentes façons de les déclencher (cliquer dessus, bouton Départ, autre événement…)
- lorsqu'un fichier de travail est fourni, on expliquera où le trouver et comment l'ouvrir

Il est ensuite conseillé de mettre les élèves par groupes de 2, tout en veillant à ce que chacun manipule le logiciel durant la séance.

Déplacement dans un labyrinthe

Pré-requis : notions de repère cartésien

Objectifs
- découvrir et manipuler les différents éléments de Scratch
- écrire un premier algorithme formé d'une seule séquence

Déroulement

Ouvrez le fichier fourni (laby.sb2) et démarrez l'activité à l'aide du drapeau vert.

Dans la partie droite de l'écran, on va pouvoir utiliser des *instructions* pour déplacer le personnage (on parle de *lutin*).

On trouve également deux blocs d'instructions plus compliqués dans cette partie de l'écran, mais il ne vous est pas demandé de les comprendre ni de les utiliser. Si vous les modifiez par erreur, rechargez le fichier de départ.

Pour l'instant, quatre instructions nous permettent de déplacer le lutin : « avancer de 100 », « avancer de 50 », « tourner à gauche de 90 degrés », « tourner à droite de 90 degrés ».

1. En cliquant successivement sur ces différentes instructions, essayez de faire sortir le lutin du labyrinthe.

   En cas d'erreur, vous pouvez recommencer au début à l'aide du drapeau vert.

   Dans quelle direction le lutin se déplace-t-il au démarrage ?

2. Une fois que vous avez bien compris comment se déplaçait le lutin, vous pouvez *programmer* son déplacement. Pour cela, empilez plusieurs blocs de mouvement en dessous du bloc « quand drapeau vert pressé ». Une fois que vous êtes satisfaits de votre programme, commencez le déplacement du lutin à l'aide du drapeau vert. S'il ne parvient pas à la sortie, modifiez votre programme, recommencez, etc.

   Pour construire le programme, vous pouvez au choix :

   - dupliquer les blocs bleus de mouvement proposés
   - ou aller en chercher autant que nécessaire dans la colonne du milieu (à condition de n'utiliser que des blocs « avancer de ... » et « tourner de ... » bien sûr)

Institutionnalisation
Bien faire la différence entre le personnage commandé (action par action) et programmé.
Mettre en évidence l'intérêt de faire des tests et de construire sa solution petit à petit (un programme écrit « d'un seul coup » a peu de chances de fonctionner et sera difficile à corriger)

Prolongements
D'autres labyrinthes plus complexes peuvent être proposés. Il ne faut pas cependant que l'écriture du programme devienne lassante, en revanche on peut en profiter pour montrer l'utilité d'une boucle pour un parcours qui serait très répétitif.

Construction de figures géométriques

Pré-requis : déplacements du lutin, géométrie du carré, éventuellement polygones réguliers

Objectifs
- découvrir la fonction « tortue »
- utiliser une boucle

Déroulement
1. Créez un nouveau projet Scratch. Dans le menu du milieu, choisissez la catégorie « Apparence » et exécutez l'instruction « Mettre à 30 % de la taille initiale ».

   Programmez quelques déplacements pour votre lutin (peu importe lesquels, évitez juste des distances inférieures à 50, sinon le résultat ne sera pas lisible). Ces déplacements devront être démarrés lorsqu'on cliquera sur le drapeau vert.

2. Dans le menu du milieu, choisissez la catégorie « Stylo » et ajoutez une instruction « Stylo en position d'écriture » au début de votre script. Exécutez le script à nouveau et observez.

3. Au bout de quelques essais, l'écran devrait être un peu saturé. Toujours dans la catégorie « Stylo », cherchez une instruction qui permettra de nettoyer la zone de dessin avant de commencer un nouveau tracé.

4. Programmez votre lutin pour qu'il trace un carré de côté 100.

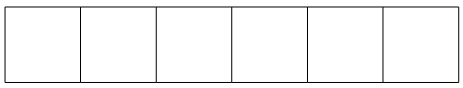
5. Programmez votre lutin pour qu'il trace un hexagone de côté 100.

6. Et si on lui demandait de tracer une figure à 20 côtés ?

   Cherchez dans la catégorie « Contrôle » un bloc d'instruction qui vous aiderait à réaliser cette tâche.

   Vérifiez que vous pouvez programmer le carré ou l'hexagone de cette façon.

7. Avez-vous remarqué ce qui se passe lorsque le lutin essaye de sortir de l'écran ?

8. Tracez 6 carrés identiques côte à côte : 

Institutionnalisation
Règle d'or : « si j'écris deux fois la même chose dans un programme, c'est que je m'y suis mal pris ». Faire remarquer que la boucle ne dispense pas de réfléchir à son écriture (ordre des instructions à l'intérieur, calcul de l'angle...)

Prolongements
Les exemples de figures et frises réalisables ne manquent pas, par exemple dans les chapitres de géométrie des manuels de mathématiques ou sur le site de Geotortue.
On pourra également donner un premier aperçu de la notion de variable en faisant demander par le lutin le nombre de côtés du polygone à tracer.

Simulation de tirage aléatoire

Pré-requis : écriture de script simple en Scratch, repère cartésien, événements équiprobables

Objectifs
- utiliser une instruction conditionnelle
- simuler une loi de probabilité simple

Déroulement
Lors de cette activité nous allons simuler plusieurs événements aléatoires, en commençant par un jeu de fléchettes.

1. Ouvrez le fichier fourni cible.sb2. Le but est de programmer la balle pour qu'elle se déplace à un endroit aléatoire du carré noir, et réagir en fonction de la zone de la cible atteinte : la balle doit afficher un message de félicitations dans les zones rouges et bleues, d'encouragement dans la zone jaune, et de défaite dans la zone blanche.
   Pour cela, voici les blocs qui pourront vous servir :
   - le bloc Mouvement « aller à x : … y : … » (déplacer la souris dans le décor et observer les nombres affichés en dessous pour voir quelles valeurs utiliser)
   - le bloc Apparence « dire … pendant 1 seconde »
   - le bloc Événement « Quand drapeau vert pressé »
   - le bloc Contrôle « Si… alors… »
   - le bloc Capteur « couleur … touchée » (pour changer la couleur, cliquer sur le petit carré de couleur puis sur une zone du décor)
   - le bloc Opérateur « nombre aléatoire entre… et… »
2. Testez votre script. Que se passe-t-il lorsque la balle arrive à cheval sur deux zones ? Utilisez le bloc « Si… alors… sinon… » pour améliorer cette situation.
3. Ouvrez un nouveau projet Scratch et simulez une pièce de monnaie avec un lutin :
   a) Dans le menu du milieu, choisissez l'onglet Costumes. Importez les deux costumes pile et face fournis, et supprimez les costumes habituels du chat.
   b) Revenez dans l'onglet Scripts, et écrivez un programme qui simule un lancer d'une pièce (équilibrée) à chaque fois qu'on clique dessus. Vous pourrez utiliser le bloc Apparence « basculer sur costume... ».
   c) Modifiez votre script pour que la probabilité de tomber sur Pile soit de 9/10.

Institutionnalisation
Le bloc pointu représente une *condition* : quelque chose qui, à un instant donné, peut être vrai ou faux. Le « sinon » permet de mieux contrôler ce qui se passe qu'avec un simple bloc « Si alors ».

Prolongements
Faire calculer un score pour le jeu de fléchettes (notion de variable).
Sur le même principe, construire un lutin qui simule un lancer de dé à l'aide des 6 costumes fournis.

Télécran

Pré-requis : tracés à l'aide d'un lutin, boucle, condition, repère cartésien

Objectifs
- découvrir les différents événements disponibles en Scratch
- programmer un projet constitué de plusieurs scripts pour un même lutin

Déroulement
Le but de ce projet est de simuler un jouet pour enfant (télécran) où on peut déplacer un curseur laissant une trace sur son passage sur un écran. Le curseur est déplacé à l'aide de deux molettes, l'une pour les déplacements horizontaux et l'autre pour les déplacements verticaux. En combinant astucieusement les déplacements, on peut même tracer des diagonales ou des dessins plus élaborés.

1. Choisissez un lutin assez discret (par exemple un de ceux nommés Button), et réduisez sa taille.
2. Combinez le bloc Événement « quand flèche droite est pressé » avec le bloc Mouvement « ajouter 10 à x ». Testez votre script pour voir son effet.
3. Construisez de même plusieurs autres scripts pour ce même lutin, de sorte que :
   - il soit possible de le déplacer dans les 4 directions
   - on puisse baisser le crayon à la demande, ou encore effacer l'écran
   - un clic sur le curseur change la couleur du stylo
4. Que pensez-vous du déplacement de ce lutin, en particulier lorsqu'on appuie longtemps dans la même direction ou sur deux touches simultanément ?

   Il est possible de faire mieux en remplaçant l'événement « quand … est pressé » par une combinaison de 4 blocs : l'événement « quand drapeau vert pressé », le capteur « touche … pressée ? », et les contrôles « répéter indéfiniment » et « si … alors … ».

Institutionnalisation
Un programme peut être constitué de plusieurs scripts, ceux-ci (ou en tous cas une partie d'entre eux) s'exécutent alors en parallèle.
Certains scripts sont déclenchés par une action extérieure (appui sur une touche, collision...).
En réalité, il y a une « boucle cachée » qui vérifie en permanence si un de ces événements se produit et quels scripts déclencher le cas échéant.

Prolongements
Jeu de Tron (effet persistant d'un événement)
En utilisant plusieurs lutins : bonus permettant de traverser les murs, jeu à deux, jeu en équipes...

Score et chronomètre

Pré-requis : boucle, condition, scripts s'exécutant en parallèle

Objectifs
- modifier un projet existant
- découvrir et utiliser les variables, le chronomètre
- utiliser des opérateurs

Déroulement
Un jeu vous est fourni mais il manque un peu d'intérêt. Vous allez le modifier pour l'améliorer, notamment en ajoutant un système de score.

1. Ouvrez le projet chat_souris.sb2 fourni, essayez d'en comprendre le fonctionnement (en particulier l'interaction des deux lutins) et testez-le. On démarre une « partie » à l'aide du drapeau vert mais celles-ci ne se terminent jamais, il faut pour cela cliquer le bouton Stop.
2. Dans la catégorie Données, créez une nouvelle variable Score (pour tous les lutins) et cochez la case voisine pour faire apparaître cette variable sur le terrain de jeu.

   À l'aide des blocs qui apparaissent dans cette catégorie, faites en sorte que le score se remette à zéro à chaque début de partie, et qu'il augmente à chaque fois qu'on attrape la souris.
3. Ajoutez une condition de fin : lorsque le score atteint la valeur 5, afficher un message et terminer la partie (à l'aide du bloc « Stop tout »).
4. Affichez la variable Chronomètre (dans la catégorie Capteurs) pour voir comment elle se comporte lors de l'exécution d'un script. Utilisez cette variable pour écrire dans votre message de fin de partie le temps total mis par le joueur. Vous pouvez utiliser l'opérateur « regroupe … … » pour construire une phrase.
5. Pour rendre le jeu plus difficile, on peut faire en sorte que la souris se déplace à nouveau si elle n'a pas été attrapée au bout de 2 secondes, ou qu'elle bénéficie d'une immunité d'une seconde avant de pouvoir être attrapée.

Institutionnalisation
Une variable permet de mémoriser une valeur.
Comme son nom l'indique, on peut changer sa valeur au cours de l'exécution du programme, soit en prenant une valeur complètement nouvelle, soit en lui ajoutant quelque chose (ou en fait, n'importe quelle opération avec les variables existantes).

Prolongements
Stocker les temps successifs dans une liste, calculer le meilleur temps.
Modifier le déplacement du chat pour qu'il « suive » la souris à vitesse constante, à vitesse variable en fonction du score.

Marche aléatoire

Pré-requis : boucle, condition, variables, simulation d'expérience aléatoire simple

Objectifs
- utiliser une boucle conditionnelle
- simuler une situation complexe
- construire un script par raffinements successifs

Déroulement
Dans cette activité, on simule le déplacement aléatoire pas à pas d'un personnage dans un environnement. On commencera pour cela par programmer un pas de déplacement, puis plusieurs, et enfin on ajoutera des compteurs pour acquérir des informations sur le déplacement effectué.

1. Mise en place : ouvrez un nouveau projet et remplacez l'arrière-plan par le fichier

   rochers.png fourni. Positionnez votre lutin sur le 2è rocher en partant de la gauche, et réduisez sa taille si nécessaire pour qu'elle corresponde à celle d'un rocher. Prévoyez un script pour réinitialiser la situation.

2. Programmez un pas de déplacement : on tire au sort si le lutin ira à gauche ou à droite, puis on effectue le déplacement. Les rochers sont distants de 50 pixels, et on considère que les deux directions sont équiprobables.

3. Enchaînez plusieurs pas de déplacements : le lutin continue à se déplacer jusqu'à ce qu'il soit revenu au rocher le plus à gauche. On peut utiliser la valeur « Abscisse x » de la catégorie mouvement pour faire ce test, ainsi qu'un bloc Contrôle bien choisi.

4. Que se passe-t-il si le lutin essaye d'aller à droite du dernier rocher ? Corriger le script si nécessaire pour qu'un lutin sur ce dernier rocher aille forcément à gauche.

5. Ajoutez un compteur pour déterminer automatiquement le nombre de pas effectués par le lutin au cours de son déplacement.

6. Mémorisez également l'abscisse la plus à droite atteinte par le lutin au cours de sa marche.

7. Modifiez les probabilités d'aller à gauche ou à droite et observez l'influence sur le nombre de pas dans une marche.

Institutionnalisation
On utilise une boucle « Répéter jusqu'à… » quand on ne sait pas à l'avance combien de fois elle devra s'exécuter ; souvent, c'est même ce nombre qui nous intéresse en sortie de boucle.
La condition d'arrêt peut être complexe.

Prolongements
Arrêter le lutin s'il n'a pas terminé sa marche au bout d'un nombre de pas fixé à l'avance, ou choisi par l'utilisateur. Animer les déplacements, programmer une marche aléatoire dans un environnement 2D.