SCIENCE NORTH SCIENCE NORD
SUDBURY, ONTARIO, CANADA

Pousser avec du code	3e et 4e année

Plan de leçon

Outil de programmation	Scratch (ou ScratchJr)
Durée	Deux périodes

Liens au curriculum de mathématiques

3e et 4e année

Algèbre : Programmation

C3. Résoudre des problèmes et créer des représentations computationnelles de situations mathématiques au moyen de concepts et d'aptitudes de programmation

Attentes précises

C3.1. Résoudre des problèmes et créer des représentations computationnelles de situations mathématiques en composant et en exécutant du code efficient, y compris du code qui comporte des événements séquentiels, concurrents et répétés

C3.2. Lire et modifier du code existant, y compris du code qui comporte des événements séquentiels, concurrents et répétés, et décrire la façon dont la modification du code influence les résultats

Liens au curriculum de sciences

3e année

Systèmes de la vie : Croissance et changements dans les plantes

3. Démontrer une compréhension que les plantes poussent et changent et ont des caractéristiques distinctes

Attentes précises

3.1. Décrire les besoins fondamentaux des plantes, y compris l'air, l'eau, la lumière, la chaleur et l'espace

3.4. Décrire la façon dont la plupart des plantes obtiennent de l'énergie pour vivre directement du soleil

4e année

Systèmes de la vie : Habitats et communautés

3. Démontrer une compréhension des habitats et des communautés et des relations entre les plantes et les animaux qui y vivent

Attentes précises

3.1. Démontrer une compréhension des habitats comme régions qui fournissent aux plantes et aux animaux les nécessités de la vie (par exemple, la nourriture, l'eau, l'air, l'espace et la lumière)

3.3. Identifier les facteurs (par exemple, la disponibilité de l'eau ou de la nourriture, la quantité de lumière, le type de conditions météorologiques) qui influencent la capacité des plantes et des animaux à survivre dans un habitat en particulier

Description

De quoi les plantes ont-elles besoin pour pousser? Au moyen d'activités pratiques combinées à des activités de programmation à l'écran, les étudiants exploreront les besoins des plantes et les représenteront au moyen de concepts de programmation. Les étudiants s'exerceront à utiliser les boucles pour développer un programme avec Scratch et explorer les motifs répétés et les événements concurrents dans le contexte de faire pousser des plantes.

Critères de réussite

À la fin de cette leçon, les étudiants devraient être en mesure de décrire et de répartir les événements comme étant soit séquentiels, soit concurrents. Ils devraient être en mesure d'utiliser des contrôles de programmation comme des boucles pour décrire des événements qui se répètent et expliquer, au moyen d'algorithmes de programmation, des programmes qui comprennent des événements séquentiels, concurrents, imbriqués et répétés.

Matériel et média

- Ordinateur ou iPad avec accès à Scratch (ou ScratchJr si l'on utilise la variante de l'activité de programmation)
- Document Pousser avec du code
- Guide de programmation Pousser avec du code ou Variante de Pousser avec du code – Guide de programmation ScratchJr pour débutants

Compétences en pensée computationnelle

Cette leçon utilise une activité pratique et un exercice de programmation par blocs en ligne pour renforcer les concepts de la pensée algorithmique et des boucles.

Dans l'activité pratique, les étudiants exploreront la logique des événements séquentiels, concurrents et imbriqués pour voir comment réorganiser l'ensemble d'instructions dans le code (l'algorithme) influence les résultats. Ils s'exerceront également à utiliser des boucles pour répéter des événements et rendre le code plus efficient. Le document **Pousser avec du code** offre des exercices qui permettent aux étudiants de s'exercer à appliquer ces concepts aux instructions associées aux plantes et à la croissance des plantes.

Dans l'activité en ligne, les étudiants seront en mesure de combiner des concepts de programmation à des exemples de facteurs qui influencent la croissance des plantes dans l'environnement. Ils construiront et modifieront du code au moyen d'événements séquentiels, concurrents et imbriqués, ainsi que des structures de contrôle comme des boucles et des instructions conditionnelles, afin de décrire la façon dont l'apparence d'un sprite de plante (résultat) change lorsqu'il interagit avec d'autres éléments graphiques du programme.

Le document **Guide de programmation Pousser avec du code** pour cette leçon comprend des procédures étape par étape détaillées pour utiliser Scratch.

Pour une version simplifiée de cette activité destinée aux programmeurs débutants, il y a également le **Guide de programmation pour débutants Pousser avec du code** étape par étape, lequel utilise ScratchJr, une application de programmation par blocs gratuite avec des blocs illustrés plutôt que du texte.

Introduction

Introduction à la pensée algorithmique

En programmation, un algorithme est une série d'étapes (ou d'instructions) qui indiquent à un programme la façon de réaliser une tâche. Utiliser de bons algorithmes, soit composer de bonnes instructions, vous permet de créer des programmes intéressants et importants.

Un programme informatique ne peut pas interpréter des instructions et prendre ses propres décisions comme le font les humains. Il ne peut que suivre les instructions exactement comme elles sont écrites. Il ne peut pas ajouter d'étapes ou changer leur ordre. Il est donc important que, lorsque vous planifiez votre code, vous organisiez vos instructions d'une façon qui est claire et précise et qui suit la bonne séquence.

Les **événements séquentiels** sont des étapes qui se produisent l'une après l'autre. Un programme informatique commencera par la première étape dans la séquence, continuera avec la deuxième étape, puis la troisième et ainsi de suite jusqu'à ce qu'il atteigne l'étape finale.

Les **événements concurrents** sont des étapes qui ont lieu en même temps ou pendant des périodes qui se chevauchent. Dans cette leçon, nous décrirons le code concurrent dans l'activité pratique et dans l'activité de programmation Scratch en créant des programmes séparés qui démarrent en même temps.

Les **événements répétés** sont décrits dans le code au moyen de boucles. L'ensemble d'instructions contenu à l'intérieur d'une boucle se répétera jusqu'à ce qu'une condition soit satisfaite pour dire au programme informatique de quitter la boucle et de passer aux prochaines étapes dans l'algorithme. La condition de la boucle pourrait être de toujours se répéter, de se répéter pour un certain nombre de cycles ou de se répéter jusqu'à ce qu'un événement se produise (par exemple, un jeu pourrait demeurer dans une boucle jusqu'à ce que toutes les vies du joueur soient épuisées, moment auquel le programme quittera la boucle pour déclencher l'écran « Partie terminée »).

Les **événements imbriqués** sont des structures de contrôle, comme les boucles et les instructions conditionnelles, qui sont placées à l'intérieur d'autres structures de contrôle. Un exemple serait une boucle imbriquée à l'intérieur d'une autre boucle. Il est utile d'imbriquer pour simplifier un programme et rendre le code plus efficient. Dans cette leçon, les étudiants exploreront les boucles imbriquées à l'intérieur d'autres boucles au cours de l'activité pratique et les instructions conditionnelles imbriquées dans des boucles dans l'activité de programmation en ligne.

Le pseudocode est le processus de composer du code hors ligne afin d'aider à planifier ce qui sera saisi avec l'ordinateur plus tard. Utiliser le pseudocode aide à rendre le code plus efficient puisqu'il élimine les erreurs potentielles avant d'investir trop de temps dans le langage de programmation. Les étudiants pratiqueront la planification de leur code en composant des instructions, en intégrant des structures de contrôle comme des boucles pour décrire les événements répétés et en indiquant clairement les ensembles concurrents d'instructions.

Introduction à la croissance et aux changements dans les plantes

Nous pouvons décrire la croissance des plantes comme une séquence ou une série d'étapes avec un ordre particulier. Par exemple :

1. Une graine est plantée et arrosée d'eau.
2. La graine germe et devient un semis avec des racines et une pousse.
3. Le semis grandit pour devenir une plante adulte avec une tige et des feuilles.
4. La plante adulte produit des fleurs.
5. Les fleurs sont pollinisées par des insectes et des animaux et avec l'aide du vent.
6. Les fleurs pollinisées produisent des fruits.

Nous pouvons également décrire la croissance des plantes au moyen de boucles, puisque les étapes associées à la croissance des plantes se répètent pour former un cycle. Par exemple :

1. Une graine est plantée et arrosée d'eau.
2. La graine germe et devient un semis avec des racines et une pousse.
3. Le semis grandit pour devenir une plante adulte avec une tige et des feuilles.
4. La plante adulte produit des fleurs.
5. Les fleurs sont pollinisées par des insectes et des animaux et avec l'aide du vent.

6. Les fleurs pollinisées produisent des fruits.
7. Le fruit contient des graines qui sont répandues par des animaux et les fruits qui tombent.
8. Retourner à la première étape.

Dans cette leçon, les étudiants s'exerceront à développer et à modifier des algorithmes et ils appliqueront cet apprentissage pour décrire les étapes associées à la croissance des plantes.

Action

Activité pratique

Le but de cette activité pratique est de présenter les concepts d'événements séquentiels plutôt que concurrents et l'utilisation des boucles pour les événements répétés. L'activité utilisera une forme de pseudocode pour décrire le code comme des ensembles d'instructions.

Première partie : Étapes séquentielles
En programmation, un algorithme est une série d'instructions ou d'étapes. Les programmes peuvent lire et exécuter votre code dans l'ordre que vous l'écrivez; votre ordre, ou votre séquence, est donc important!

Voyons un simple ensemble d'instructions.

Écrivez les instructions suivantes sur le tableau et dessinez une boîte autour des étapes :

1. Lève-toi
2. Tape des mains
3. Tape des mains
4. Tape des mains
5. Assieds-toi

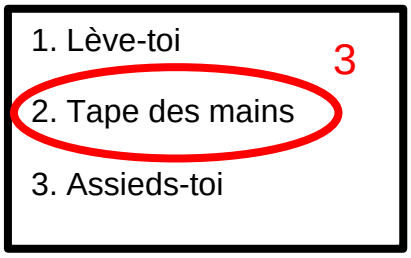
Qu'est-ce que ces instructions nous disent de faire? Comment suivriez-vous ces instructions?

Démontrez la première séquence (décrite dans l'exemple ci-dessus) en écrivant les étapes sur le tableau et en demandant aux étudiants d'exécuter les mouvements en ordre. Répétez cette activité avec les étudiants en choisissant de nouveaux mouvements et leur séquence.

Deuxième partie : Ajouter des boucles (étapes répétées)
En programmation, nous utilisons les boucles comme une instruction pour répéter des étapes. Cela rend le code plus efficient et plus facile à comprendre que si nous écrivions nos étapes encore et encore (comparez la différence entre dire « avance d'un pas trois fois » et « avance d'un pas, puis avance d'un pas, puis avance d'un pas »).

Par exemple, nous pouvons rendre la séquence de notre premier exercice plus efficiente en ajoutant une boucle. Donc, plutôt que d'écrire l'étape « tape des mains » trois fois, nous n'avons qu'à l'écrire une fois et utiliser une boucle pour communiquer que nous allons répéter cette étape trois fois.

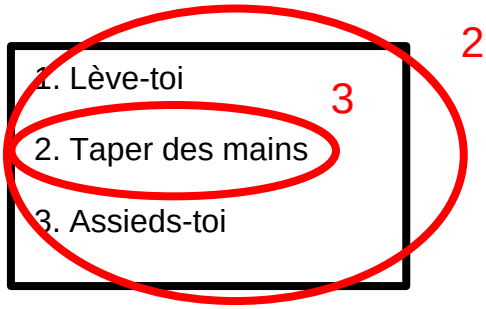
Nous représenterons notre boucle avec un cercle dessiné sur le tableau. Chaque étape contenue dans le cercle sera répétée. Nous indiquerons le nombre de fois que nous allons répéter les étapes en écrivant un nombre à côté du cercle de la boucle. Cela ressemblera à ceci :



Demandez aux étudiants d'exécuter les mouvements dans ces instructions. Le résultat devrait être le même que pour notre première séquence, mais avec moins d'étapes. Répétez cette activité avec les étudiants en choisissant de nouveaux mouvements et leur séquence. Faites-les expérimenter avec le nombre d'étapes qu'ils placent dans une boucle et le nombre de répétitions qu'ils attribuent à la boucle.

Qu'arrive-t-il s'ils placent une boucle autour de l'ensemble complet d'instructions? (Réponse : L'ensemble complet d'instructions se répète.)

Qu'arrive-t-il si vous imbriquez des boucles (par exemple, placez des boucles à l'intérieur de boucles)? Si nous plaçons une boucle autour de l'exemple ci-dessus :



Maintenant, les étudiants devront répéter la séquence (se lever, taper des mains trois fois, s'asseoir) deux fois (se lever, taper des mains trois fois, s'asseoir, se lever, taper des mains trois fois, s'asseoir). Notez que cette boucle traite la séquence entière comme une seule unité (par exemple, la nouvelle boucle ne dit pas de taper vos mains six fois avant de passer à l'étape suivante).

Troisième partie : Étapes concurrentes

Dans nos deux exercices précédents, nous exécutons nos étapes une à la fois, en ordre. Mais que faites-vous lorsque vous voulez dire à votre programme de faire deux choses à la fois?

Nous indiquerons les programmes concurrents en dessinant les étapes que nous voulons être exécutées en même temps à l'intérieur de boîtes côte à côte.

Par exemple, supposons que nous voulons dire à notre programme de compter jusqu'à dix à haute voix en même temps qu'il exécute notre programme original de se lever, de taper des mains et de s'asseoir :

Événement 1

1. Lève-toi
2. Tape des mains
3. Assieds-toi

3

Événement 2

1. Compte jusqu'à 10 à haute voix

Remarque : Les étudiants peuvent suggérer de décrire le programme pour l'événement 2 comme une série d'étapes (par exemple, 1. Dire « un » à haute voix; 2. Dire « deux » à haute voix; etc.), ce qui est aussi acceptable.

Faites la démonstration de l'exemple ci-dessus en écrivant les séquences concurrentes sur le tableau et en demandant aux étudiants d'exécuter les instructions de l'événement 1 et de l'événement 2 en même temps. Répétez cette activité avec les étudiants en choisissant les mouvements et leur séquence.

Quatrième partie : Mettre tout ensemble

Maintenant que les étudiants se sont exercés à utiliser les événements séquentiels et concurrents, les boucles et les boucles imbriquées, encouragez-les à appliquer cet apprentissage pour élaborer une danse (considérez : comment décririez-vous les instructions pour des danses comme le boogie-woogie?). Cela peut être fait en petits groupes qui présenteront alors le code de leur danse à la classe que tout le monde tentera de suivre.

Si les instructions de danse d'un groupe d'étudiants ne sont pas claires ou donnent une danse différente de celle prévue par les étudiants, il s'agit alors d'une belle opportunité pour la classe de « déboguer » les mouvements de danse. Habituellement, cela signifie simplifier des instructions (par exemple, en s'assurant que chaque étape décrit une seule action) et séparer les instructions compliquées en des événements concurrents plus courts.

Par exemple, si une danse comprend une étape comme « lève ton bras droit et remue-le », le processus de débogage pourra ressembler à ceci :

- Clarifier : est-ce que tu lèves ton bras droit **ET ENSUITE** le remues? Si c'est le cas, cette étape doit être séparée en deux étapes dans la même séquence (1. Lève ton bras droit; 2. Remue ton bras droit).
- Est-ce que tu lèves ton bras droit **EN MÊME TEMPS** que tu le remues? Si c'est le cas, alors cette étape doit être divisée en des événements concurrents.

Événement 1

1. Lève ton bras droit

Événement 2

1. Remue ton bras droit

Rappelez aux étudiants que lorsqu'ils planifient leur code, ils peuvent écrier leurs algorithmes dans ce format pour les aider à schématiser leurs instructions sur papier avant de passer au logiciel de programmation.

Le document **Pousser avec du code** offre d'autres exercices pour les étudiants pour décrire les instructions au moyen d'événements séquentiels, concurrents et répétés.

Activité mathématique de programmation

Le but de l'activité de programmation est d'utiliser l'application de programmation par blocs Scratch pour créer un sprite de plante qui « poussera » (augmentera en taille) lorsqu'il touche des facteurs (rayon de soleil, eau) qui soutiennent la croissance des plantes. Ce code permettra aux étudiants de s'exercer à utiliser des boucles et des événements imbriqués.

Un guide détaillé étape par étape pour construire ce programme dans Scratch, y compris les possibilités d'approfondissement, est fourni dans le **Guide de programmation Pousser avec du code.**

Pour les étudiants qui apprennent toujours à lire ou qui sont des débutants absolus en programmation, faites appel au guide de programmation simplifié : **Variante Pousser avec du code – Guide de programmation ScratchJr pour débutants.** Dans ce guide de programmation simplifié, nous utiliserons ScratchJr pour créer une scène avec un arrière-plan, un soleil qui bouge quand on clique dessus, un nuage de pluie qui se déplace lorsque l'on clique dessus, une plante qui pousse lorsque l'on clique dessus et des fleurs qui apparaissent.

La leçon de programmation simplifiée utilise ScratchJr plutôt que Scratch. ScratchJr utilise des pictogrammes sur ses blocs de programmation plutôt que du texte pour illustrer les fonctions de ces blocs. Ce programme sera utile pour les étudiants dont les aptitudes en littératie ne sont pas encore à un niveau où ils peuvent facilement comprendre le langage écrit sur les blocs Scratch ou pour familiariser les étudiants avec la logique de la programmation par blocs et le développement d'algorithmes de base.

Conclusion et évaluation

À la fin de cette leçon, les étudiants devraient être en mesure de décrire les éléments de programmation comme séquentiels, concurrents ou imbriqués et de décrire la façon dont ces différentes dispositions d'instructions influencent la façon dont les instructions (algorithmes) sont exécutées. Les étudiants devraient être en mesure de décrire les boucles et la façon dont les instructions contenues dans les boucles sont répétées.

Pour l'évaluation, recueilliez le document **Pousser avec du code** des étudiants. Examinez leur travail afin de vous assurer qu'ils ont compris les concepts d'événements séquentiels, d'événements concurrents et de boucles en évaluant leurs réponses au moyen de la clé de correction.

Adaptation

- Les actions utilisées dans l'activité pratique peuvent être adaptées pour répondre aux besoins de mobilité ou d'accessibilité des étudiants.
- Le système de symboles utilisé pour indiquer les algorithmes d'événements et de boucles peut être adapté pour répondre aux besoins d'accessibilité des étudiants (y compris la disposition d'images physiques des étapes).

Extensions

- Les étudiants qui terminent de programmer plus tôt peuvent ajouter d'autres facteurs qui influencent leur sprite de plante. Les détails relatifs aux possibilités d'extension sont compris dans le **Guide de programmation Pousser avec du code.**

• Une variante du guide de programmation est incluse parmi les ressources pour cette leçon qui peut être utilisée par les étudiants qui apprennent toujours à lire et qui peuvent trouver le texte dans les blocs de programmation Scratch inaccessibles.	