Scratch et Snap! Pour initier à la programmation

publié le 24/02/2016 - mis à jour le 25/02/2016

Comparatif des interfaces Scratch et Snap !

Descriptif :

Cet article présente les fonctionnalités de l'interface Snap ! qui peut constituer une alternative au logiciel Scratch dans le cadre de l'initiation à la programmation au collège.

Sommaire :

- Scratch et l'initiation à la programmation
- Un clone de Scratch : Snap !
- Les fonctionnalités intéressantes de Snap !

Scratch et l'initiation à la programmation

Le contexte de la réforme du collège

Dans le cadre de la réforme du collège, l'initiation à la programmation fait son entrée dans les programmes à partir du cycle 2 :

> *Dès le CE1, les élèves peuvent coder des déplacements à l'aide d'un logiciel de programmation adapté, ce qui les amènera au CE2 à la compréhension, et la production d'algorithmes simples.*

Cette initiation se poursuit au cycle 3 :

> *Une initiation à la programmation est faite à l'occasion notamment d'activités de repérage ou de déplacement (programmer les déplacements d'un robot ou ceux d'un personnage sur un écran), ou d'activités géométriques (construction de figures simples ou de figures composées de figures simples). Au CM1, on réserve l'usage de logiciels de géométrie dynamique à des fins d'apprentissage manipulatoires (à travers la visualisation de constructions instrumentées) et de validation des constructions de figures planes. À partir du CM2, leur usage progressif pour effectuer des constructions, familiarise les élèves avec les représentations en perspective cavalière et avec la notion de conservation des propriétés lors de certaines transformations.*

Et prend vraiment tout son sens au cycle 4 :

> *En 5ème, les élèves s'initient à la programmation événementielle. Progressivement, ils développent de nouvelles compétences, en programmant des actions en parallèle, en utilisant la notion de variable informatique, en découvrant les boucles et les instructions conditionnelles qui complètent les structures de contrôle liées aux événements.*

Scratch, un outil adapté à la programmation au collège

Scratch est un langage de programmation visuel et gratuit. En glissant-déposant des blocs colorés, il est possible de créer des histoires interactives, des jeux, des animations, de la musique, ou des présentations. Il est ensuite possible de les télécharger sur Internet pour les partager avec les autres utilisateurs du monde entier. Scratch est conçu pour jouer, apprendre par soi-même et créer.
Il a été conçu par le MIT (Massachusetts Institute of Technology, un des instituts universitaires les plus importants

des États-Unis) pour initier les jeunes à des concepts importants en mathématiques et informatique, tout en apprenant à développer une pensée créative, un raisonnement systématique et à travailler en équipe.
De par son fonctionnement, il constitue une solution pertinente pour la programmation au collège. En effet :

- sa prise en main par les élèves est quasi-immédiate ;
- l'environnement est simple et efficace : constitué de trois parties, on a les instructions, la fenêtre du programme et la fenêtre d'exécution du programme sur le même écran ;
- il n'y a pas de syntaxe à connaître, ni à écrire : on déplace simplement des blocs d'instructions qui s'imbriquent par aimantation ;
- il est adapté à la programmation événementielle : les scripts démarrent à partir d'un événement et les objets peuvent communiquer entre eux par des messages ;
- un simple double-clic sur une instruction permet de l'exécuter, ce qui permet de vérifier facilement la bonne programmation d'un objet ;
- il est dynamique : il est possible de modifier le code d'un programme en cours d'exécution et d'exécuter plusieurs programmes en parallèle ;
- le débogage est très simple les blocs de programme peuvent être détachés les uns des autres et testés indépendamment, ce qui permet de rapidement isoler une erreur ;
- il apporte des rendus visuels grâce à des scènes et des costumes et constitue une interface attractive pour le jeune public ;
- à partir de l'interface de développement, on peut envoyer sur un espace de partage en ligne son programme ce qui facilite les échanges et favorise les interactions entre élèves ;
- on trouve sur cet espace un partage de nombreux programmes, parfois utiles pour débloquer des situations ;
- Il existe un forum en français autour de « Scratch ».

Pour une présentation plus approfondie de cet outil, vous pouvez consulter ce guide en ligne (cliquer sur l'image) :

Copie d'écran du guide de Scratch.

ainsi que le guide de prise en main proposé lors des journées de formation *"algorithmique et programmation"* :

Livret d'accompagnement pour la prise en main de Scratch (PDF de 2.3 Mo)
Livret d'accompagnement pour la prise en main de Scratch

Les limites de Scratch

Le fonctionnement de Scratch repose sur la technologie "Flash", qui nécessite l'installation d'Adobe Air sur la machine pour une utilisation locale et l'installation du plug-in Adobe Flash Player dans le navigateur pour une utilisation en ligne.
Outre le fait que ces deux logiciels soient des logiciels propriétaires, cela pose un problème supplémentaire lorsqu'on veut utiliser Scratch sur une tablette puisque le format "Flash" n'est pas lisible de manière native sur ces appareils.
Des solutions existent mais elle ne sont pas satisfaisantes pour une utilisation en classe avec des élèves.
Par ailleurs, bien qu'il soit possible de créer ses propres blocs de commande sous Scratch, ceux-ci restent attachés à un projet et ne peuvent pas être exportés vers d'autres projets, à moins d'exporter le lutin dans lequel le bloc a été défini.

Un clone de Scratch : Snap !

Principe du langage

Logo de Snap !

Snap ! est un langage de programmation visuel, basé sur l'imbrication de blocs de commandes, au même titre que Scratch. En effet, **c'est une ré-implémentation de Scratch** développée par l'université de Berkeley en Californie et qui permet de créer ses propres blocs de programmation. Par ailleurs, Snap ! intègre des fonctionnalités plus pointues en programmation et peut donc être utilisé par un public plus avancé.
En outre, Snap ! est écrit en Javascript et utilise le format HTML pour les animations, ce qui le rend portable sur n'importe quel navigateur : Chrome (le plus adapté), Safari, FireFox, Internet Explorer (quelques limitations),..
De plus, il est aussi utilisable sur les appareils mobiles ayant une connexion Internet (téléphones et tablettes android (privilégier chrome), téléphone et tablettes Apple (à partir d'IOS8 mais pas de lecture de sons).
Le principe de fonctionnement est le même que Scratch : on construit des scripts associés à des objets (lutins ou scène) en imbriquant par aimantation des blocs de programmation. Le glisser-déposer sur une tablette est très simple et se fait avec un seul doigt.
Snap ! peut être affichée en français et **la palette des blocs contient tous les blocs présents dans Scratch**, traduits de la même façon.

Présentation de l'interface

L'image interactive suivante présente l'interface graphique de Snap !

Image interactive de l'interface de Snap! (Thinglink)
Image interactive décrivant l'interface graphique de Snap !

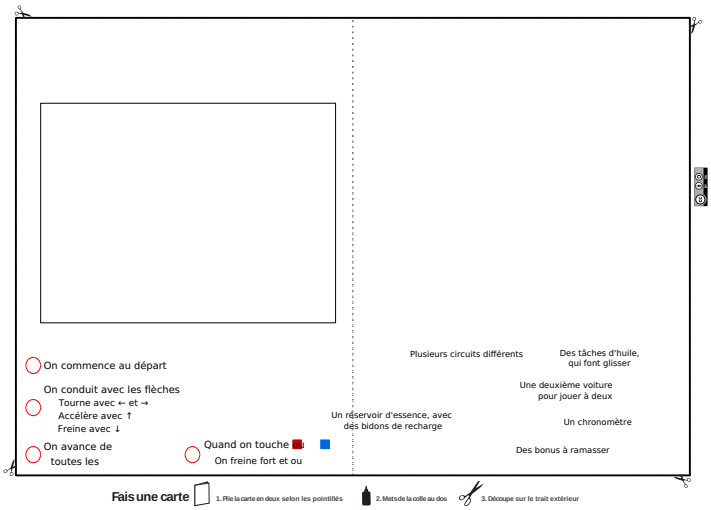
Création et export de projets

L'utilisation se fait directement en ligne. Les projets créés peuvent être sauvegardés sur le serveur dans un espace dédié (Browser) et l'on ne pourra accéder à cet espace que si on utilise la même machine (reconnaissance avec adresse IP) utilisant le même navigateur. Il est aussi possible de se créer un compte et d'enregistrer ses projets dans un espace associé à ce compte (Cloud). En outre, un projet peut être partagé avec la communauté en ligne (Share) mais peut aussi être sauvegardé dans la machine au format *.xml*.
Des blocs de commande qui auraient été créés dans un projet peuvent aussi être exportés au format *.xml* pour être réutilisés dans un autre projet.
De la même manière, des lutins peuvent être exportés au format *.xml* et l'on peut les réutiliser dans un autre projet en important le fichier *.xml* correspondant.

Quelques exemples

Le site coding4kids propose quelques exemples clés en main présentés sous forme de cartes :

- "Course de voitures" :



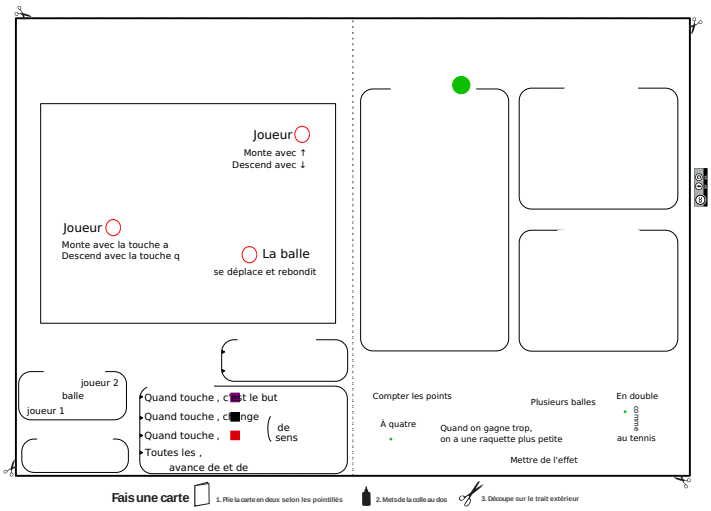
Carte du jeu de course de voiture (PDF de 120 ko)

Carte du jeu de course de voiture

Fichier source du jeu de course de voiture (XML de 23.8 ko)

Fichier source du jeu de course de voiture

- "Jeu de Ping Pong" :



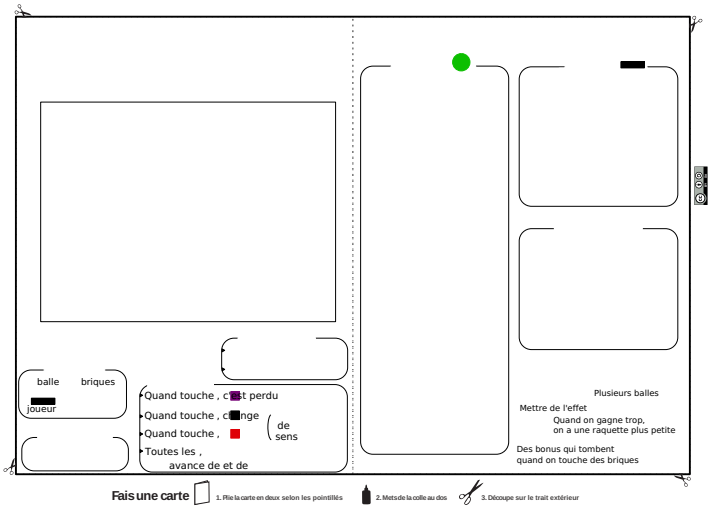
Carte du jeu de ping pong (PDF de 153.7 ko)

Carte du jeu de ping pong

Fichier source du jeu de ping pong (XML de 9.6 ko)

Fichier source du jeu de ping pong

- "Jeu de casse-briques" :



Carte du jeu de casse-briques (PDF de 131.9 ko)

Carte du jeu de casse-briques

Fichier source du jeu de casse-briques (XML de 33 ko)

Fichier source du jeu de casse-briques

Les fonctionnalités intéressantes de Snap !

Créer des lutins animés

Tout comme dans Scratch, il existe un éditeur de dessin qui permet de dessiner ses propres lutins mais celui-ci est moins élaboré que dans Scratch (pas de possibilité d'insérer du texte, par exemple).
En revanche, il est possible de créer un lutin en plusieurs parties puis de les associer ensemble, afin de les animer : un des lutins joue le rôle d'ancre et les autres lutins viennent s'attacher à celui-ci comme des parties.
Pour comprendre ce procédé, on peut s'appuyer sur l'exemple d'un lutin *hélicoptère* dont on a animé les hélices :

- duplication du lutin *hélicoptère* en trois exemplaires ;
- placement des lutins dans la scène pour reconstituer l'hélicoptère ;
- création par effacement du lutin *hélicoptère seul*, du lutin *hélice avant* et du lutin *hélice arrière* ;
- glissement de l'icône de l'*hélice avant* (dans la zone des objets) sur le lutin *hélicoptère* (dans la scène) afin de l'attacher à celui-ci. Même action pour l'*hélice arrière* ;
- animation des hélices.

Un petit aperçu du fichier en images :

Les icônes des hélices comportent une icône en haut à droite, signifiant leur attachement à un autre lutin

Les hélices tournent indéfiniment dès que le drapeau vert est pressé.

Les hélices sont attachées à l'hélicoptère et tournent indéfiniment.

Créer ses blocs de commandes

La grande plus-value de Snap ! est la possibilité de créer ses propres blocs. Les blocs créés peuvent être de tous types. Par exemple :

- il est possible de créer une commande graphique *rectangle* de variables *longueur* et *largeur* et de l'utiliser pour tracer des rectangles de dimensions variables :

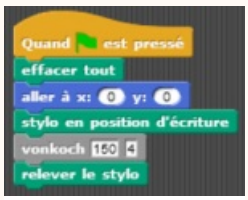
Copie d'écran illustrant la définition de la commande Rectangle

Copie d'écran illustrant une utilisation de la commande Rectangle

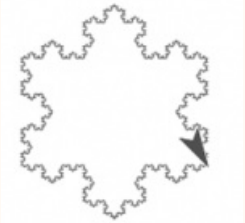
Copie d'écran illustrant un tracé obtenu avec la commande Rectangle.

- on peut définir des fonctions récursives comme le flocon de Von Koch :

Copie d'écran illustrant les définitions des commandes segment et vonkoch

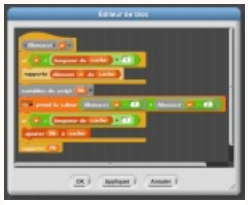


Copie d'écran illustrant une utilisation de la commande vonkoch



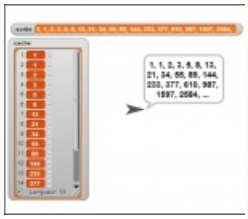
Tracé d'un flocon de Von Koch utilisant la commande vonkoch

- on peut définir des fonctions d'une variable, comme une fonction *fibonacci* qui calcule les termes successifs de la suite de Fibonacci puis une commande *liste_fibonaci(N)* qui affiche les N premiers termes de cette suite :



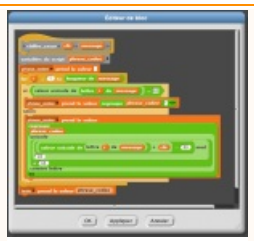
Copie d'écran de la définition de la fonction "fibonacci"

Copie d'écran de la définition de la commande "liste_fibonacci"



Affichage des premiers termes de la suite de Fibonacci

- on peut définir une fonction de cryptage avec le chiffrement de César (décalage des lettres de l'alphabet donné par une clé) : celle-ci sera utilisée pour chiffrer un texte puis pour décrypter un texte par la force brute en affichant tous les cryptages possibles dans une liste :



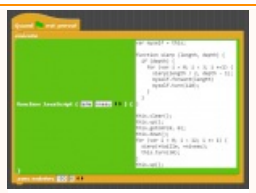
Définition de la fonction de codage par le chiffrement de César

Script du lutin qui intègre les méthodes de cryptage et de décryptage

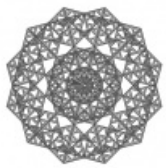
Liste des cryptages possibles pour le décryptage d'un texte chiffré

- on peut aussi définir une commande à partir d'un code JavaScript, que l'on intègre ensuite dans un bloc *Exécuter* :

Définition de la fonction à partir d'un code JavaScript



Inclusion de la fonction dans un bloc d'exécution



Tracé d'un exemple

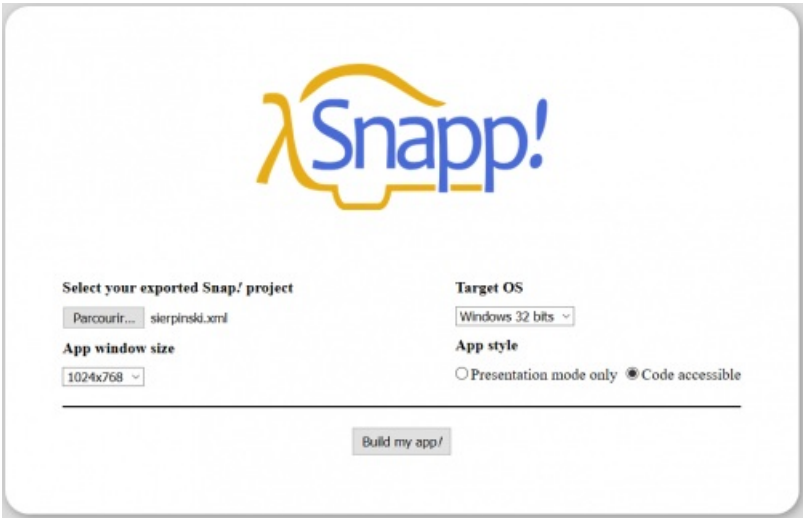
Convertir un fichier Scratch en un fichier Snap !

Un deuxième avantage de Snap ! est que l'on peut facilement migrer de Scratch vers Snap ! en utilisant un convertisseur en ligne : l'utilitaire Snapin8r2 permet de convertir un fichier *.sb2* en un fichier *.xml* qui pourra être ouvert dans l'interface Snap !. Les lutins et la scène présents dans le fichier d'origine sont importés ainsi que les costumes et arrière-plans associés.

Logo de l'interface Snapin8r2

Exporter un fichier Snap ! en un module autonome

Un dernier avantage offert par Snap ! est la possibilité d'exporter un projet sous la forme d'un module autonome : l'utilitaire Snapp ! permet de convertir un fichier *.xml* en un dossier compressé *.zip* pour lequel on peut spécifier le système d'exploitation (Windows, Linux ou OSX) :



Copie d'écran de l'invite de Snapp !

Il reste ensuite à décompresser ce dossier dans lequel se trouve un fichier *.exe* que l'on exécute sans aucune installation supplémentaire. On obtient la même interface que sur le web et on peut modifier les valeurs des différentes variables. Le fichier est protégé en écriture et les manipulations effectuées ne sont pas sauvegardées.

Un exemple de module autonome sur le triangle de Sierpinski : pour Windows, pour Linux et pour OSX.

Portfolio

Documents joints

Carte du jeu "Pose la fusée" (PDF de 122.9 ko)

Carte du jeu "Pose la fusée"

Fichier source de l'hélicoptère (XML de 55.6 ko)

Fichier source de l'hélicoptère

Fichier source du jeu "pose la fusée" (XML de 51.7 ko)

Fichier source du jeu "pose la fusée"

Académie de Poitiers

Avertissement : ce document est la reprise au format pdf d'un article proposé sur l'espace pédagogique de l'académie de Poitiers.

Il ne peut en aucun cas être proposé au téléchargement ou à la consultation depuis un autre site.