Bien commencer avec

Introduction à l'informatique

par Jeremy Scott

LIVRET DE L'ÉLÈVE

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 « Bien commencer avec Scratch » 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

Introduction 6
Qu'est-ce qu'un ordinateur ? 6
Les types d'ordinateurs 7
Les différents éléments d'un ordinateur 10
Le matériel informatique 11
Les logiciels 12
Les langages de programmation 13

Programmer avec Scratch 14

1 : Les bases de Scratch 17
Le monde est une grande scène de théâtre 17
L'ordre c'est important 19
As-tu compris ? 20
Paresseux ou astucieux ? 23

2 : C'est l'heure d'une histoire 24
Les erreurs de programmation 26
La programmation événementielle 28

3 : Le labyrinthe 31
De l'importance de la conception 31

4 : Une question d'image... 40
Imbrication 42

5 : Tir à l'arc en forêt 48
Les variables 51

Résumé 54

Un projet avec Scratch 56
Félicitations ! 65

Introduction

Sans même le savoir, tu as probablement déjà utilisé plusieurs ordinateurs aujourd'hui.

Si tu as envoyé un SMS, fait un tour en voiture ou regardé ta montre, alors tu as utilisé un ordinateur. Les mots que tu lis actuellement ont été tapés sur le clavier d'un ordinateur.

Les ordinateurs sont partout autour de nous. Ils sont si présents dans nos vies qu'il est important de comprendre comment ils fonctionnent.

Qu'est-ce qu'un ordinateur ?

Un ordinateur est une **machine** qui exécute les instructions que lui donne un humain. Sans instructions, les ordinateurs seraient incapables de faire quoi que ce soit.

Si tel est le cas, qu'est-ce qui les rend exceptionnels ? Eh bien, les ordinateurs...

- effectuent les tâches plus rapidement que les humains[2] ;
- sont plus précis que les humains ;
- peuvent stocker d'énormes quantités d'information qu'ils n'« oublient » jamais.

On pourrait croire que les ordinateurs peuvent pratiquement tout faire. Toutefois, il y a quelques points essentiels à garder à l'esprit :

- Les ordinateurs n'ont pas de cerveau ; ils ne sont pas plus intelligents que les humains.
- Les ordinateurs n'ont ni sentiments ni « bon sens ». Cela signifie que beaucoup des tâches quotidiennes que réalisent les humains ne peuvent être effectuées par un ordinateur.

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

1. _____
2. _____
3. _____

2 Au moment de la rédaction du présent document, un ordinateur individuel moderne était capable d'effectuer plus de 100 milliards de calculs par seconde !

Les types d'ordinateurs

Il existe des ordinateurs de formes et de tailles différentes. Parmi les ordinateurs que la plupart des gens reconnaîtront sans doute, on retrouve :

L'ordinateur de bureau

Un **ordinateur de bureau** est conçu pour être placé sur – ou sous – un bureau et être utilisé par une personne à la fois. Il doit être branché sur secteur et est constitué de plusieurs appareils distincts.

L'ordinateur portable

Les ordinateurs portables regroupent les différents appareils d'un ordinateur de bureau au sein d'une même unité. Ils peuvent être transportés et être branchés sur secteur ou être alimentés par batterie. Les netbooks et les ultrabooks sont simplement des types d'ordinateurs portables plus petits et plus légers.

La tablette

Cet appareil possède un grand écran tactile que l'on active avec le doigt (ou parfois avec un stylet). Il possède une batterie et se transporte très bien. Chaque tablette possède un clavier « virtuel » qui s'affiche à l'écran[3].

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 ?

_____

_____

_____

[3] Le terme « virtuel » est beaucoup utilisé en informatique. Il signifie simplement « non réel » et sert à qualifier quelque chose qui a été recréé sur ordinateur. Quelles autres choses virtuelles peut-on trouver sur un ordinateur ?

Il existe d'autres ordinateurs que le grand public connait moins bien ou a du mal à reconnaître comme tel, notamment :

L'ordinateur central

Il s'agit d'un énorme ordinateur susceptible d'occuper toute une pièce. Plusieurs utilisateurs peuvent l'utiliser simultanément, chaque utilisateur ayant son propre clavier, sa propre souris et son propre écran.

Les ordinateurs centraux coûtent très cher et une équipe de plusieurs personnes est nécessaire pour les faire fonctionner. Ils appartiennent à de grandes entreprises ou organisations qui doivent stocker et traiter d'énormes quantités d'informations.

Le serveur

Un serveur est un ordinateur qui fournit des services utilisés par d'autres ordinateurs au sein d'un réseau. Il existe par exemple :

- les serveurs de fichiers, qui stockent les fichiers des utilisateurs,
- les serveurs Web, qui donnent accès à des pages Web,
- les serveurs de messagerie, qui fournissent des services de courrier électronique

La console de jeux

Les consoles de jeux sont aussi des ordinateurs. La plupart des consoles possèdent un lecteur de disque pour charger des jeux. Elles disposent aussi d'un puissant processeur pour créer des images réalistes.

De nombreuses consoles permettent aussi aux utilisateurs de se connecter à Internet, afin d'acheter des jeux en ligne ou de se mesurer à d'autres joueurs à travers le monde.

Les systèmes embarqués

Plusieurs des appareils utilisés chez soi possèdent un **système embarqué** – une petite puce électronique qui exécute des instructions stockées en mémoire. La maison moderne possède plus de 100 de ces « ordinateurs », intégrés à des appareils tels que le grille-pain, la chaîne stéréo, la machine à laver, le réfrigérateur, la télévision, etc.

Une voiture moderne est susceptible de posséder 100 autres de ces systèmes embarqués, voire plus[4].

Activité

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

1. _____

2. _____

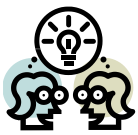
3. _____

Le smartphone

En anglais, « smart » veut dire « intelligent ». Les smartphones, tels que l'Android ou l'Apple iPhone, sont donc des téléphones intelligents, des ordinateurs de poche avec lesquels il est aussi possible de téléphoner. De nombreux smartphones possèdent un large écran tactile.

C'est un bon exemple de **convergence** où des technologies précédemment séparées sont maintenant regroupées en un même appareil.

Activité



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

1. _____

2. _____

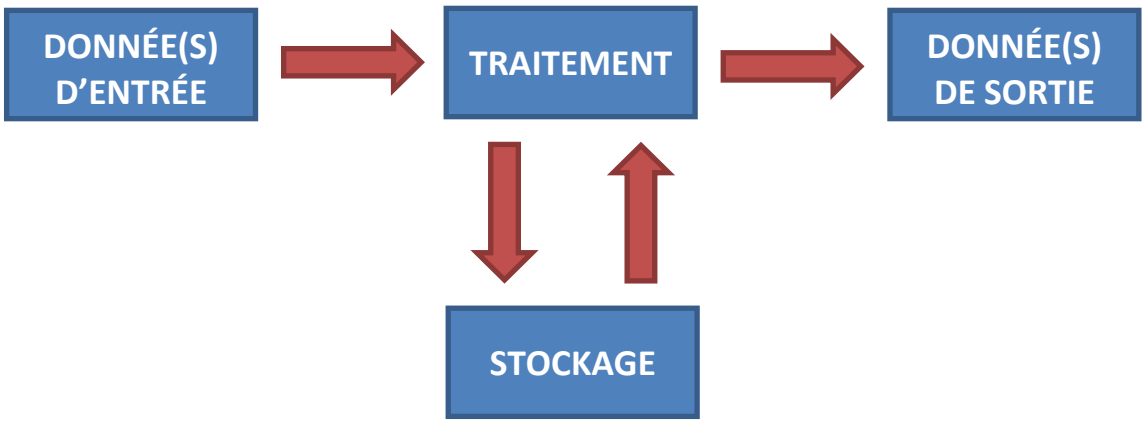
3. _____

[4] Pour en savoir plus : https://interstices.info/vehicules-surs-propres

Les différents éléments d'un ordinateur

Un ordinateur est une machine qui :

- recueille des informations
- stocke ces informations
- traite ces informations
- et renvoie les informations traitées.



Activité **Indique les <u>données d'entrée et de 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		
Téléphoner		
Regarder la télé		

Les éléments d'un ordinateur peuvent être répartis en **deux** grandes catégories : le **matériel informatique** et les **logiciels**.

Le matériel informatique

Le matériel informatique est un **ensemble d'appareils distincts**.

Ces appareils peuvent être répartis en **quatre** grandes catégories :

Les **périphériques d'entrée**	utilisés pour faire **entrer** des données dans l'ordinateur.
L'**unité centrale**	où l'ordinateur exécute les instructions données par les programmes. Plus le processeur est rapide, plus l'ordinateur exécute rapidement les tâches.
Les **périphériques de sortie**	utilisés pour faire **sortir** des données de l'ordinateur.
Les **périphériques de stockage**	utilisés pour **stocker** des programmes et des données. C'est là que l'on sauvegarde son travail.

Autrement dit, si ça se touche, c'est du matériel informatique.

Activité

À quelle catégorie penses-tu que les appareils ci-dessous appartiennent ? Répartis les périphériques d'<u>entrée</u>, de <u>sortie</u> et de <u>stockage</u> 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	écran

Les logiciels

Un ordinateur peut réaliser différentes tâches en fonction des **instructions** qui lui sont données.

Une liste d'instructions est appelée un **programme**. Sans programme pour lui dire quoi faire, un ordinateur ne serait qu'un ensemble (inutile) d'appareils informatiques.

On appelle « **logiciels** » les programmes et les informations qu'ils utilisent.

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	Internet Explorer
Jouer à des jeux	Angry Birds
Modifier un fichier vidéo	iMovie

Les langages de programmation

Les ordinateurs **suivent les instructions** que leur donnent les humains. Ils ne peuvent résoudre que les problèmes qui leur sont posés par des humains. Pour dire à un ordinateur ce qu'il doit faire, tu dois savoir quel problème tu souhaites résoudre et avoir un plan pour y parvenir.

Malheureusement, il n'est pas possible de simplement donner ces instructions à un ordinateur en français. Un ordinateur peut exécuter des tâches très rapidement, mais il n'est pas intelligent au contraire des humains.

Un ordinateur :

- **ne fera que ce qu'on lui demande** et
- **exactement ce qu'on lui demande**.

Cela signifie que les programmes informatiques doivent être écrits d'une façon très précise, selon des règles strictes. Il ne doit y avoir aucune ambiguïté quant à la signification des instructions.

L'ensemble d'instructions et de règles permettant d'écrire un programme est ce qu'on appelle un **langage de programmation**.

Programmer avec Scratch

Dans la suite de ce cours, nous apprendrons à écrire des programmes informatiques.

Tu utiliseras **Scratch,** créé par le MIT (Massachusetts Institute of Technology), un des instituts universitaires les plus importants des États-Unis.

Scratch est un puissant outil de développement de logiciels. Il permet de créer des programmes (appelés « **projets** ») en associant sons, éléments graphiques et animations.

Tu peux mettre en ligne des projets sur le site Web de Scratch et échanger avec d'autres *Scratchers* à travers le monde. C'est vraiment le top du top !

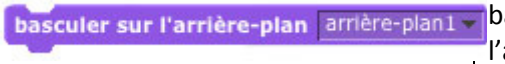
Tu apprendras à utiliser Scratch à travers une série de leçons. À la fin de chaque leçon, il y aura quelques questions pour vérifier que tu as bien compris les concepts abordés.

La mascotte de Scratch au Media Lab du MIT

Avertissement :

Les ressources dans la suite de ce livret sont souvent en anglais, ainsi que le contenu des programmes de scratch (plus précisément, tu verras, le texte des blocs) ; pas de panique, l'anglais utilisé est simple, mais nous t'accompagnons en te donnant la traduction, soit sous les textes, soit comme pour les blocs dans le tableau ci-après pour que tu puisses t'y référer si besoin.

		attendre … secondes
		quand le drapeau vert est cliqué
		répéter jusqu'à …
		(répéter) indéfiniment
		envoyer (à tous)
		quand je reçois
		quand espace est pressé
		quand Lutin… est pressé
		si
		répéter jusqu'à…
		arrêter tout
		mettre l'effet … à …

		dire… pendant … secondes
		basculer sur l'arrière plan
		avancer de x pas
		Se diriger vers…
		aller à x… /y…
		tourner à droite de x degrés
		jouer le son
		touche espace pressée
		couleur pressée
		effacer
		mettre la couleur du stylo à …
		stylo en position d'écriture
		relever le stylo

1 : Les bases de Scratch

Cette leçon abordera

- 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 son et animation

Introduction

Regarde la vidéo d'introduction à Scratch (elle est en anglais). Elle te fera découvrir Scratch et son interface.

http://www.youtube.com/watch?feature=player_detailpage&v=jxDw-t3XWd0

Le monde est une grande scène de théâtre

Un programme en Scratch contient des **lutins**, sorte de personnages qui « jouent » sur une **scène**. Les lutins et la scène ont trois types de **propriétés** (ou réglages) :

1. **Les scripts**

   Il s'agit des **instructions** qui contrôlent un lutin. Les scripts sont constitués de **blocs**.

   Il existe huit différents types de blocs – liés au mouvement, au contrôle, à l'apparence, etc. – et plus de 100 blocs au total. **Il est à noter que les lutins ont besoin des scripts pour réaliser une tâche.**

2. **Les costumes et les arrière-plans**

   Les **costumes** sont les « tenues » des lutins. Un même lutin peut avoir plusieurs costumes et peut donc changer d'apparence.

   La **scène** peut avoir plusieurs **arrière-plans** que l'on peut modifier. Les arrière-plans sont à la scène ce que les costumes sont aux lutins.

3. **Les sons**

   Il s'agit des sons que peuvent utiliser les lutins ou la scène. Encore une fois, chaque lutin (ou la scène) peut disposer de plusieurs sons différents. Scratch te permet d'**importer** (d'introduire) des sons enregistrés ou même d'enregistrer tes propres sons grâce à un microphone.

Tâche nº 1 : Premiers pas

Regarde la capture vidéo intitulée « **Catwalk** ».

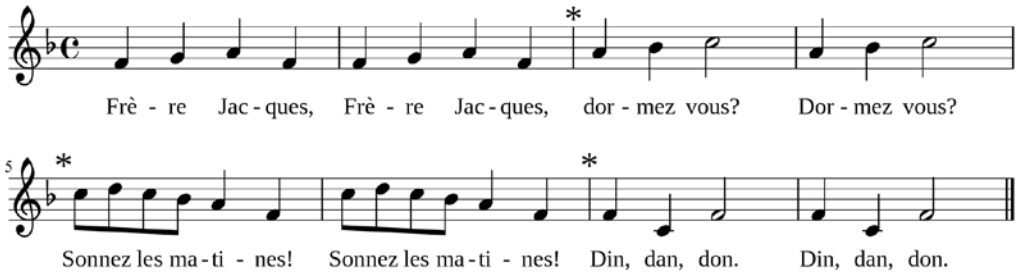
Elle introduit les principaux éléments de Scratch et t'accompagne dans la création de ton premier programme informatique. En cas de problème, retourne à la capture vidéo ou demande de l'aide à ton coéquipier.

Tâche nº 2 : Frère Jacques

Regarde la capture vidéo « **FrereJacques** ».

Elle explique comment créer une simple mélodie dans Scratch. En cas de problème, retourne à la capture vidéo ou demande de l'aide à ton coéquipier.

Savais-tu que... ? *Frère Jacques* est l'une des chansons les plus connues à travers le monde. Sais-tu de quoi parle vraiment la chanson ? Elle raconte l'histoire d'un moine, le frère Jacques, dont le rôle est de faire sonner les cloches du matin. C'était avant l'ère des réveils. Mais le pauvre moine ne s'est pas réveillé à l'heure !



Tâche nº 3 : Mes mélodies

Après avoir réalisé la tâche n° 2, essaie de créer un programme qui joue une autre chanson simple.
Choisis une chanson dont les lignes de musique se répètent, pour pouvoir utiliser la commande **répéter**.

Félicitations ! Tu viens d'entamer une aventure qui va faire de toi un programmeur !

L'ordre c'est important

Les blocs **d'un même script** sont exécutés les uns après les autres, **de façon séquentielle**.

Les blocs **de scripts différents** peuvent parfois être exécutés **simultanément**. On parle alors de **traitement en parallèle** : il s'agit de demander à l'ordinateur de faire plus d'une chose à la fois.

Par exemple, si tu as plusieurs scripts de type 

, ils seront tous exécutés **en même temps** lorsque tu cliqueras sur le drapeau vert.

Activité supplémentaire n° 1 : Faut que ça danse !

Essaie de faire danser un lutin au son de ta musique, en lançant le programme grâce à un clic sur le drapeau vert. Il y a **deux** méthodes pour y arriver :

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

Tu peux t'inspirer d'une autre capture vidéo (« **Dancing Queen** ») disponible à l'adresse suivante : http://info.scratch.mit.edu/node/164.

Toutefois, attention à bien créer une mélodie plutôt que de simplement utiliser une boucle de musique !

Activité supplémentaire n° 2 :

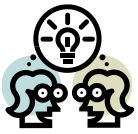
Fais des essais en ajoutant à ton programme d'autres blocs comme les blocs d'apparence tels que 

.

Ils te permettent de créer des effets vraiment sympas !

As-tu compris ?

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 ?



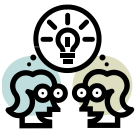
_____

_____

Pourquoi ? _____

_____

À présent, teste toi-même le code pour voir si tu avais raison.

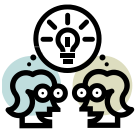


1.2 Maintenant, ajoute un bloc **attendre 1 seconde** entre les deux blocs de mouvement. Décris ce qui se passe.

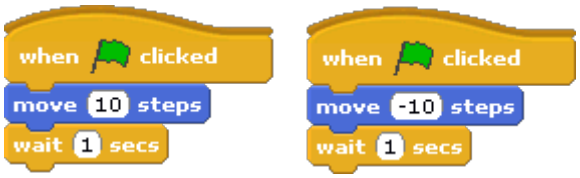
_____

Pourquoi obtient-on ce résultat ? _____

_____



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 ?

_____

_____

Pourquoi ? _____

_____

À présent, teste toi-même le code pour voir si tu avais raison.

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

_____

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 !

Pourquoi ?

_____

_____

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 ?

_____

_____

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**, avec des blocs de mouvement parmi des 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** ?

_____

_____

1.8 Prépare une autre question du même genre que les questions 1.1 à 1.6 et pose-la à ton voisin.

_____

_____

_____

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 « Frères 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.

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

Cette leçon abordera

- la création d'histoires et de pièces de théâtre
- la rédaction de séquences d'instructions
- les événements
- la commande « envoyer à tous »

Tâche nº 1 : Une mauvaise blague

Regarde la capture vidéo intitulée « **BadJoke** » (en anglais). Elle montre comment utiliser Scratch pour raconter une blague ou monter une pièce avec deux lutins.

Après avoir vu la vidéo, essaie de raconter une autre blague – du type « Toc toc ! – Qui est là ? » par exemple – avec deux personnages, comme dans l'exemple.

Réfléchis bien au moment où chaque personnage (lutin) « parle » ; planifie le code, y compris les temps de parole et d'attente, comme dans l'exemple ci-dessous.

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.

_____

_____

_____

_____

Tâche nº 2 : Une petite pièce de théâtre

Écris une petite histoire ou une petite pièce. Il devrait y avoir **deux ou trois arrière-plans** pour lesquels les acteurs (les lutins) changeront de costumes.

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

Astuce : Tu peux utiliser le bloc **envoyer à tous** pour permettre à un lutin de déclencher un événement tel qu'un changement de scène. Par exemple,

Dans le script du lutin	Dans le script de la scène
	

Tu peux t'inspirer d'une autre capture vidéo (« **Haunted Scratch** ») disponible à l'adresse suivante : http://info.scratch.mit.edu/node/165.

Activité supplémentaire nº 1 : Marcher c'est bien

Fais en sorte que tes personnages marchent à l'écran et s'arrêtent à certains moments durant la pièce.

Astuce : tu devras faire partir les lutins acteurs du bord de l'écran et utiliser les blocs **montrer** et **cacher** pour qu'ils apparaissent toujours à l'endroit voulu.

Les erreurs de programmation

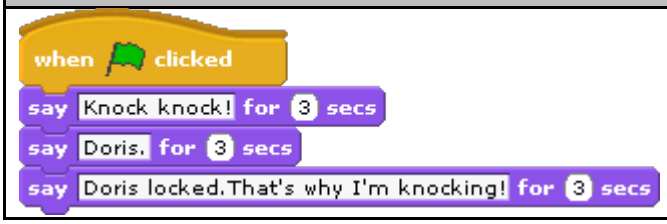
Une **erreur de programmation** (ou « **bug** ») est une erreur qui empêche ton code de fonctionner comme prévu. Il existe **deux** grandes catégories d'erreurs de programmation :

- **Les erreurs de syntaxe**
  Ce genre d'erreur survient si l'on ne respecte pas 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** ».

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
	

traduction : « Toc-toc » « qui est là ? »

« Doris » « Doris qui ? »

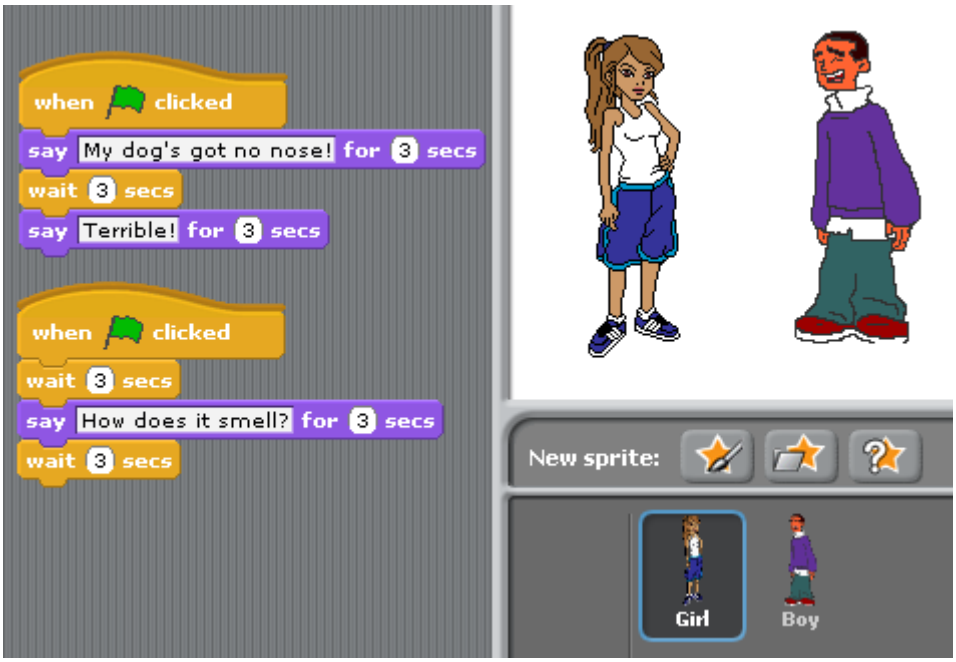
« Doris enfermée. Voilà pourquoi je frappe !»

« gémissement »

_____

_____

2.2 Le programme ci-après montre les scripts permettant à deux lutins de se raconter une blague. Mis à part le fait que la blague est très mauvaise, qu'est-ce qui ne va pas dans ce programme ?



traduction : « Mon chien n'a pas de museau » « c'est terrible »

« Comment sent-il ? »

_____

_____

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
when clicked say Knock, knock! for 2 secs wait 3 secs say Doris. for 2 secs wait 3 secs say Doris locked, that's why I'm knocking! for 3 secs	when clicked wait 3 secs say Who's there? for 2 secs wait 3 secs say Doris who? for 2 secs wait 3 secs say GROAN! for 3 secs

_____

_____

_____

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 appuie 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** ».

2.5 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.

_____

_____

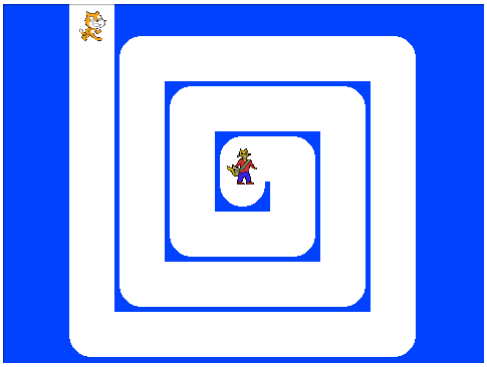
_____

_____

_____

3 : Le labyrinthe

Cette leçon abordera



- La création d'un jeu
- La détection de collisions

Introduction

Tu vas maintenant créer un jeu simple où le joueur guide un personnage « explorateur » dans un labyrinthe grâce aux touches de direction.

Le jeu s'achève quand l'explorateur sauve son ami piégé au centre du labyrinthe.

Introduction

Regarde la capture vidéo intitulée « **Maze** » pour découvrir comment créer un jeu de labyrinthe.

Tâche nº 1 : Planter le décor

Commence par importer l'arrière-plan adéquat – un labyrinthe – et deux lutins : un explorateur et un ami à sauver. **Ne fais rien de plus pour l'instant**.

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** (l'apparence) 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.
- Pour concevoir le **code**, écris **en français** la liste des étapes à exécuter. C'est ce que l'on appelle un **algorithme**. C'est comme rédiger les étapes d'une recette de cuisine.

C'est là l'essence même de la programmation. Il s'agit de résoudre des problèmes, et non de taper des commandes sur ordinateur.

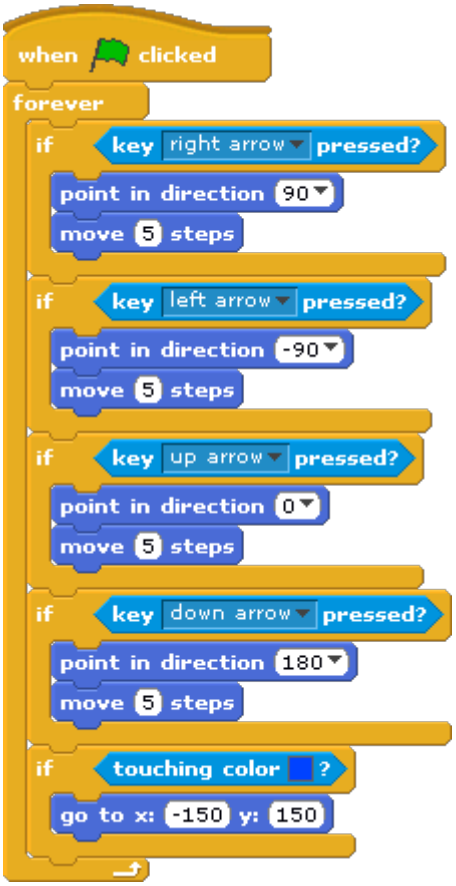
Tout bon programmeur conçoit des algorithmes avant de commencer à coder !

Tâche nº 2 : Concevoir la solution

Revenons aux 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)

Le tableau ci-dessous montre un **algorithme** permettant de faire avancer l'explorateur et le **code** correspondant en Scratch.

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	

Les algorithmes permettent aux programmeurs de se concentrer sur ce que le programme doit faire plutôt que sur la façon de le faire sur ordinateur. Une fois l'algorithme finalisé, écrire le code devient facile !

As-tu remarqué qu'un algorithme est **indenté** pour que l'on sache quelles parties sont **imbriquées** dans d'autres ? Par exemple,

répéter indéfiniment
→ si l'utilisateur appuie sur la touche de déplacement vers la droite
est à l'intérieur de l'instruction **répéter indéfiniment**
→ pointer vers la droite ..... est à l'intérieur de l'instruction **si l'utilisateur appuie sur la touche de déplacement vers la droite**
→ avancer de 5 pas ..... est à l'intérieur de l'instruction **si l'utilisateur appuie sur la touche de déplacement vers la droite**

Tâche n° 2 : Concevoir la solution (suite)

Le tableau ci-dessous montre un algorithme pour le lutin ami de l'explorateur.

En te basant sur cet algorithme, essaie de créer toi-même le code. **Pense à bien le mettre dans le script du lutin ami.**

Algorithme pour atteindre le centre du labyrinthe	Code pour le lutin ami
lorsque l'utilisateur clique sur le drapeau montrer le lutin répéter indéfiniment si le lutin explorateur est touché dire « Merci ! » cacher le lutin arrêter tous les scripts	**Code toi-même le script !**

À présent, teste ton jeu pour voir s'il fonctionne.

Activité supplémentaire nº 1 : Quand la musique est bonne

Ajoute une musique de fond à ton jeu (le son « xylo1 » semble adéquat, mais choisis celui que tu préfères).

Réfléchis aux points suivants :

- Quel serait le meilleur endroit pour stocker ce son, dans la mesure où il s'applique à l'ensemble du jeu ?
- Comment vas-tu faire pour que la musique joue sans interruption ?
- Faudrait-il utiliser un bloc **jouer le son** ou **jouer le son complètement** pour jouer le morceau ?

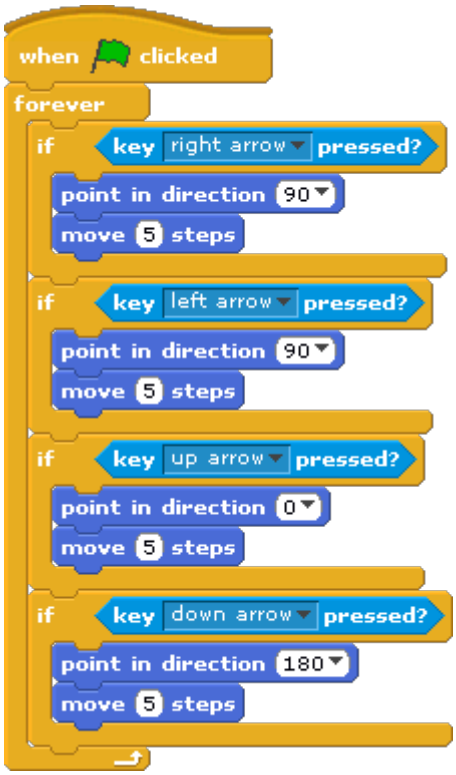
Activité supplémentaire nº 2 : Ajouter un ennemi

Ajoute un lutin qui se déplace sans arrêt d'un bout à l'autre de la scène.
Si ton explorateur touche l'ennemi, l'explorateur doit retourner au point de départ.

Astuce : fais en sorte que ton lutin ennemi ne puisse bouger que vers la gauche et vers la droite.
Le bloc **rebondir si le bord est atteint** est utile pour bondir et rebondir sur le bord de la scène.

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-t-il commise ?

_____

_____

_____

_____

_____

_____

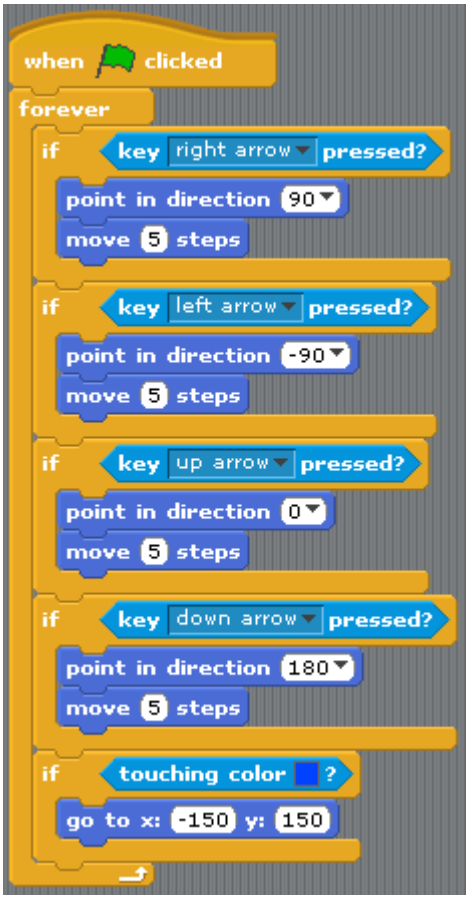
_____

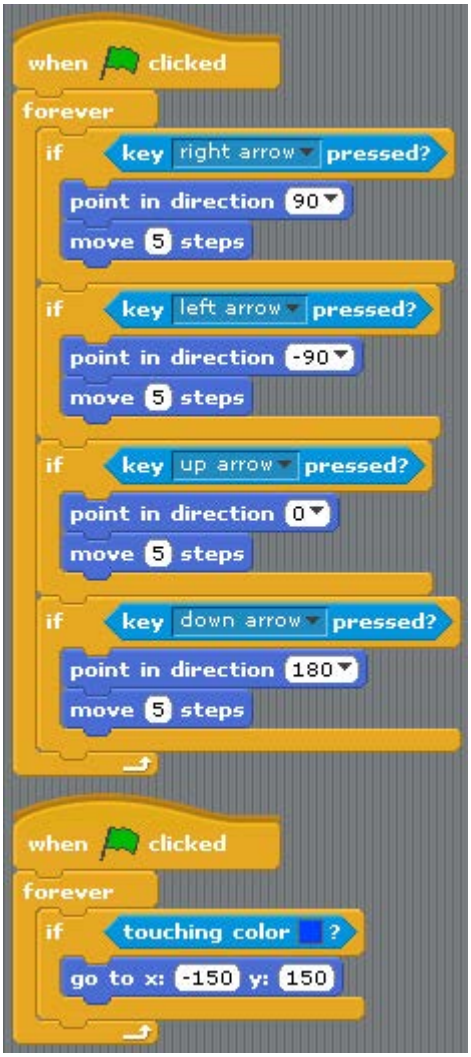
_____

_____

_____

3.2 Étudie les exemples de code ci-dessous.





Accomplissent-ils la même tâche ? _____

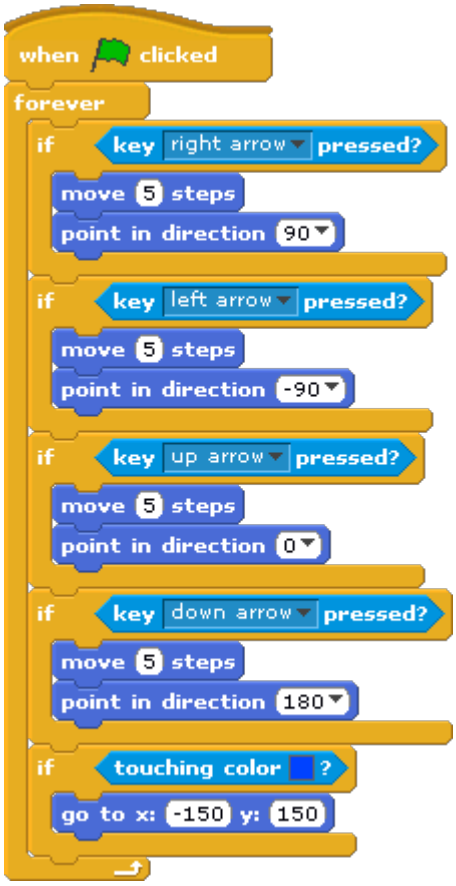
Explique ta réponse _____

_____

_____

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 faite ?

_____

_____

_____

_____

_____

_____

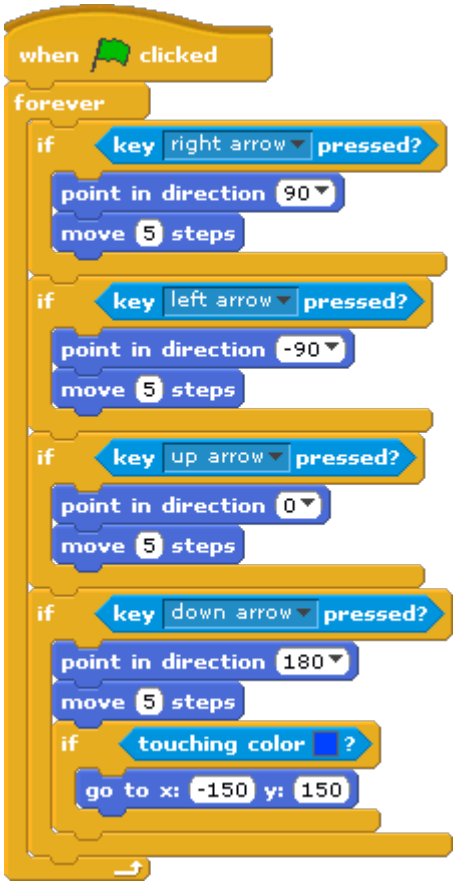
_____

_____

_____

_____

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 faite ?

_____

_____

_____

_____

_____

_____

_____

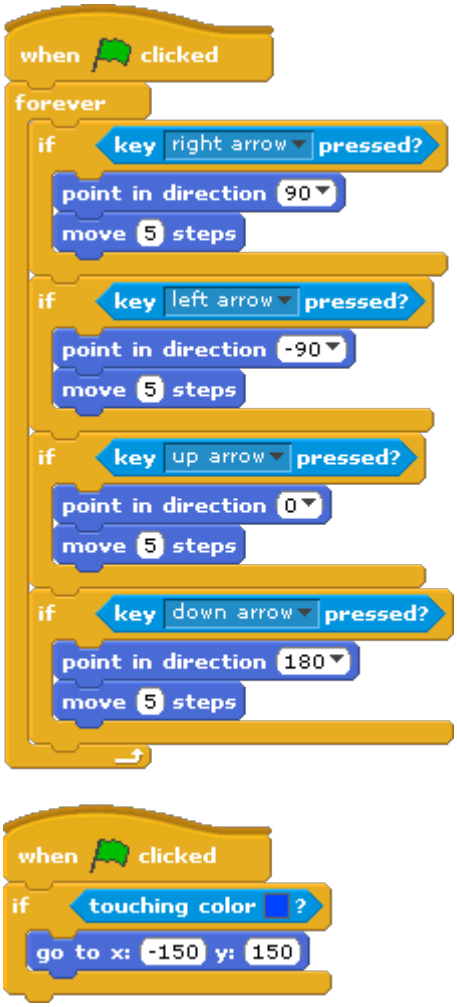
_____

_____

_____

_____

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 faite ?

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

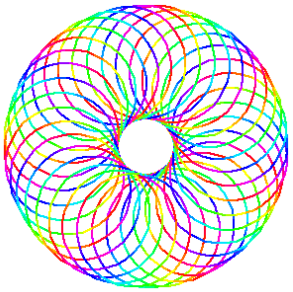
_____

_____

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

4 : Une question d'image...

Cette leçon abordera



- L'environnement de Scratch
  - Les lutins
  - Les blocs de code
- Les boucles « répéter »
- La commande « envoyer à tous et attendre »
- La programmation d'éléments graphiques

Introduction

Dans cette leçon, tu vas écrire des programmes permettant de créer de simples éléments graphiques en utilisant les blocs **Stylo** de Scratch.

Tâche nº 1 : Remise en forme

Regarde la capture vidéo intitulée « **Graphics** ». Elle illustre comment utiliser Scratch pour créer de simples éléments graphiques.

Complète le tableau ci-dessous avec les programmes permettant de créer l'**heptagone** (7 côtés) et le **triangle** :

Carré	Pentagone	Hexagone	Heptagone	Triangle
				

Maintenant, teste tes programmes (double-clique sur la pile de blocs ou ajoute un bloc quand drapeau pressé au tout début).

Tes programmes fonctionnent-ils ? _____

Si la réponse est « non », explique pourquoi. _____

_____

_____

La règle de la rotation

As-tu identifié quelle règle s'applique ici ?

Dans chaque forme, nous avons fait un **tour complet** (360°). Pour déterminer de combien de degrés il faut tourner à chaque fois, il suffit...

De diviser le nombre total de degrés nécessaires pour faire un tour complet par le nombre de rotations à effectuer

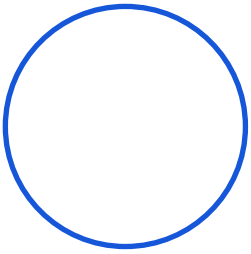
Donc... dans un carré, pour faire un tour complet de 360° il faut 4 rotations, donc 360/4 = 90° à chaque rotation ;

dans un pentagone, pour faire un tour complet de 360° il faut 5 rotations, donc 360/5 = 72° à chaque rotation ;

Tâche n° 2 : Une étoile est née !

Maintenant, utilise la règle de la rotation pour dessiner une étoile à cinq branches (ci-contre).
Astuce : Fais **très attention** à ce que dit la règle !

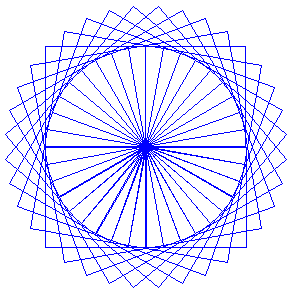
Tâche n° 3 : Le cercle



Crée un cercle. C'est plus simple à faire que tu pourrais le penser : il suffit de

```
répéter 36 fois
    avancer de 5 pas
    tourner de 10 degrés
```

Tâche n° 4 : Motif circulaire



Réalise un motif constitué de 36 carrés formant un unique cercle comme ci-contre.

```
répéter 36 fois
    dessiner un carré.... insère ici le code permettant de dessiner un carré
    tourner de 10 degrés
```

Essaie avec des carrés, des triangles ou des hexagones.

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.

Activité supplémentaire n° 1 : L'événement principal

Crée tes propres scripts quand je reçois pour dessiner chacune des formes que tu as déjà créées (carré, triangle, pentagone, etc.). Pense à utiliser le bloc envoyer à tous et attendre pour déclencher les blocs quand je reçois.

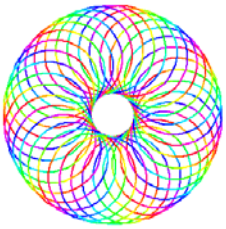
Une fois terminé, adapte le programme que tu as créé pour la **tâche n° 4 « Motif circulaire »**, afin d'utiliser un bloc envoyer à tous et attendre pour la forme qui se répète.

Activité supplémentaire n° 2 : Notre maison

Dessine une maison comme celle ci-contre.
Écris un algorithme – c'est à dire, planifie les étapes sur papier –
avant de tenter de coder ceci !
Tu devras utiliser les blocs relever le stylo **et** abaisser le stylo.

Astuce : Réfléchis à la façon dont tu pourrais utiliser la commande « **envoyer à tous et attendre** » pour réduire la quantité de code à créer.

Activité supplémentaire nº 3 : Oh, les beaux anneaux !



Adapte le motif créé précédemment pour créer un anneau multicolore.
Écris un algorithme avant de tenter de coder ceci !

Astuce : Il y a 36 cercles, mais il faut relever le stylo et le déplacer légèrement avant d'abaisser le stylo et de dessiner le prochain cercle. Le programme utilise aussi le bloc modifier la couleur du stylo par pour ajouter de la couleur.

Activité supplémentaire nº 4 : Les anneaux olympiques[5]

Les choses se compliquent ! Essaie d'écrire un programme permettant de dessiner les cinq anneaux olympiques.
Écris un algorithme avant de tenter de coder ceci !

Astuce : réalise chaque cercle en utilisant une commande envoyer à tous et attendre et réfléchis à l'espacement entre les centres.

Savais-tu que... ? Le drapeau olympique a été hissé pour la première fois aux Jeux Olympiques d'été de 1920, à Anvers, en Belgique. Depuis, il a été hissé à chaque édition des Jeux Olympiques.

Les cinq anneaux représentent les cinq continents : l'Amérique, l'Afrique, l'Australie, l'Asie et l'Europe. Les couleurs – bleu, jaune, noir, vert et rouge sur fond blanc – ont été choisies parce qu'au moins une de ces couleurs est présente dans chaque drapeau national.

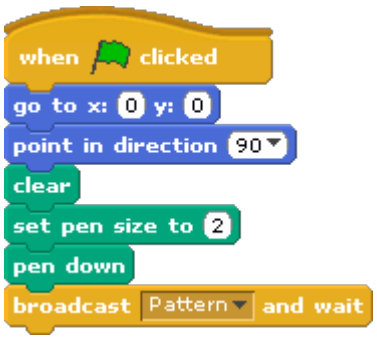
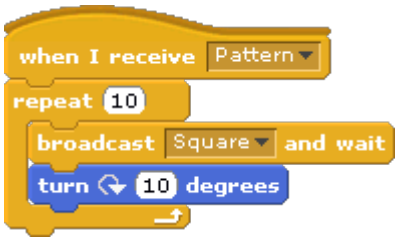
---

[5] 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 scripts sont exécutés une fois que l'utilisateur a cliqué sur le drapeau vert (numérote-les de 1 à 3).

Numéro	Script
	
	
	

Maintenant, décris ce que va faire le code.

_____

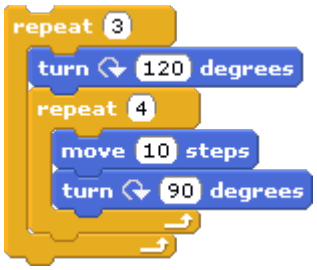
_____

_____

_____

_____

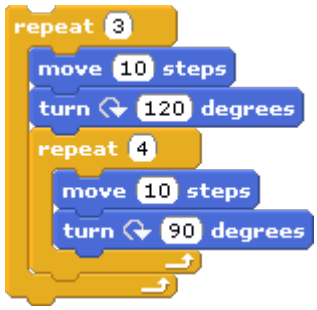
4.2 Étudie les exemples de code ci-dessous.



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

Pourquoi ? _____

_____



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

Pourquoi ? _____

_____

_____

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

a) Se préparer pour l'école

b) Préparer le petit déjeuner

_____ _____

_____ _____

_____ _____

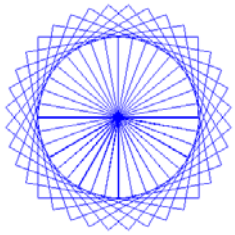
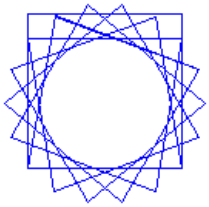
_____ _____

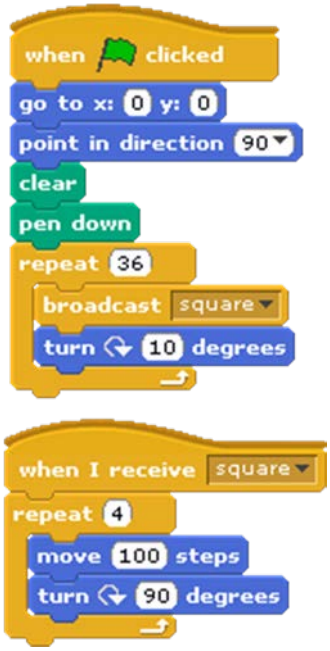
_____ _____

_____ _____

Réfléchis-y : dans chaque exemple, y a-t-il des étapes qui pourraient faire l'objet de scripts séparés et être réalisées 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 commise ?
Astuce : la réponse est en lien avec la vitesse d'exécution de l'ordinateur.

_____

_____

_____

_____

_____

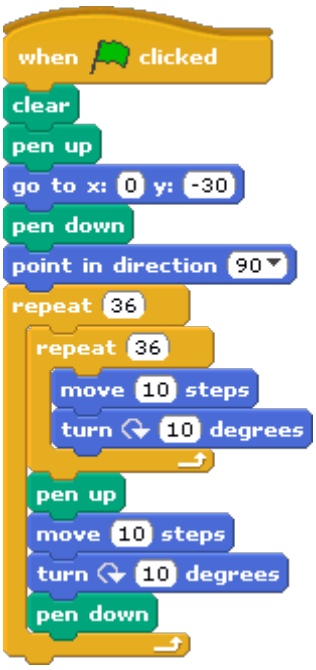
_____

_____

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 commise ?

Ne t'inquiète pas si tu ne trouves pas immédiatement la solution – cet exercice n'est pas simple ! Si nécessaire, reproduit le script dans Scratch et exécute-le pour t'aider à comprendre ce qui se passe.

_____

_____

_____

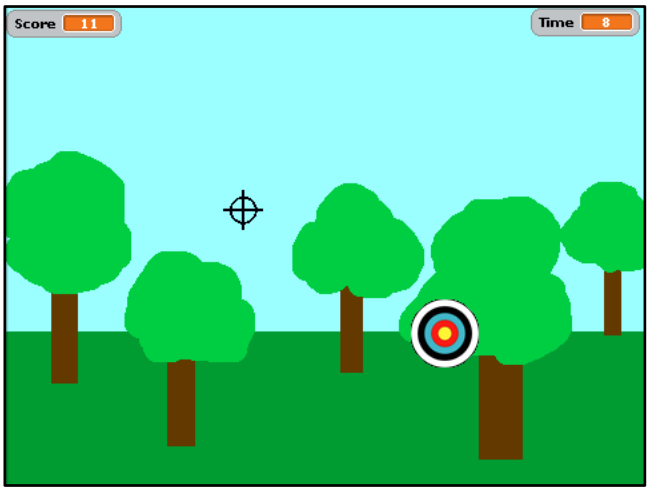
_____

_____

_____

_____

5 : Tir à l'arc en forêt



Cette leçon abordera

- Les instructions conditionnelles
- les boucles conditionnelles
- les variables
- les nombres aléatoires
- l'animation
- le son

Introduction

Regarde la capture vidéo « **ForestArchery** » pour comprendre comment créer ce jeu.

Tâche n° 1 : Concevoir la solution

Revenons aux deux principaux éléments que nous devons coder pour notre jeu :

1. déplacer la cible
2. tirer sur la cible

Essaie de coder ton programme à partir des algorithmes donnés ci-après, plutôt que de visionner à nouveau la capture vidéo.

Algorithme pour déplacer la cible (dans le script du lutin cible)

lorsque l'utilisateur clique sur le drapeau
répéter indéfiniment
  glisser en 1 seconde jusqu'à une position aléatoire*

* *x est un nombre aléatoire entre -240 et 240*
* *y est un nombre aléatoire entre -180 et 180*

Algorithme pour déplacer le viseur et tirer (dans le script du lutin viseur) / …

Algorithme pour déplacer le viseur et tirer (dans le script du lutin viseur)

lorsque l'utilisateur clique sur le drapeau
répéter indéfiniment
    aller à l'emplacement du pointeur de la souris (*les positions x et y de la souris*)
    si l'utilisateur a cliqué avec la souris (*souris pressée*)
        si le lutin touche le lutin cible
            ajouter 1 à la variable « score »
            jouer le son « Pop ! »
            Dire « Touché ! » pendant 0,5 seconde

Tâche n° 2 : Toucher ou manquer

Modifie ton code pour que le programme compte les **coups manqués (en retranchant 1 point du score)** en plus des **coups réussis** :

Si la cible est touchée
    ajouter 1 point au score
    jouer le son « Pop ! »
    Dire « Touché ! » pendant 0,5 seconde
sinon
    retrancher 1 point du score (ajouter -1)
    jouer le son
    dire « Manqué ! » pendant 0,5 seconde

Tâche n° 3 : Contre la montre

Ajoute une **variable** chronomètre à ton programme pour que le jeu ne dure que 30 secondes. Fais en sorte que la variable apparaisse à l'écran pendant le compte à rebours de 30 jusqu'à 0.

lorsque l'utilisateur clique sur le drapeau
répéter 30 fois
    attendre 1 seconde
    retrancher 1 de la variable « temps » (ajouter -1)
arrêter tous les scripts

Tâche nº 4 : En plein dans le mille !

En utilisant les blocs **si** et **couleur touchée**, modifie le programme pour que lorsque la cible est atteinte, le score augmente selon la répartition des points suivante :

- Zone blanche – 1 point
- Zone noire – 2 points
- Zone bleue – 3 points
- Zone rouge – 4 points
- Zone or – 5 points (en disant « Dans le mille ! »)

Tâche nº 5 : Restons positifs !

Adapte le programme pour que le joueur ne puisse **jamais** avoir un score négatif.
Astuce : enlève un point si et seulement si le score est supérieur à zéro.

Savais-tu que... ? Cela fait au moins 10 000 ans que l'homme pratique le tir à l'arc. Le tir à l'arc a d'abord été une méthode de chasse (voir la peinture rupestre ci-contre[6]), avant d'être utilisé à la guerre.

Dans l'Angleterre médiévale, tous les hommes avaient l'obligation de pratiquer le tir à l'arc régulièrement, afin d'être prêts à combattre en cas de guerre.

De nos jours, le tir à l'arc est un loisir populaire très pratiqué à travers le monde.

---

[6] © Instituto de Turismo de España (TURESPAÑA). Cette photo d'une peinture rupestre de Cova dels Cavalls demeure la propriété exclusive de Turespaña et ne peut être utilisée ou reproduite sans accord écrit préalable de Turespaña.

Les variables

Dans ce jeu, nous avons introduit l'idée de stocker un score en utilisant un bloc variable.

Une **variable** est un espace dans la mémoire d'un ordinateur où il est possible de garder des informations utilisées par notre programme – c'est comme conserver des choses dans une boîte.

Lorsque l'on nomme une variable, il est important de lui donner un **nom** qui nous indique clairement **le type d'information qui y est stockée** – c'est comme placer une étiquette sur la boîte afin d'indiquer ce qu'elle contient.

Pour créer une variable dans Scratch, nous **créons un bloc de type « variable »**.

Une fois la variable créée, il est possible de lui **attribuer** une valeur ou de **modifier** cette dernière (il est donc possible de faire varier la valeur, d'où le terme « variable »).

Activité supplémentaire n° 1 : Top chrono !

Nous allons maintenant ajouter une nouvelle fonctionnalité au jeu de **labyrinthe** de la leçon n° 3 – un **chronomètre** qui donne à l'utilisateur 30 secondes pour finir le jeu.

Pour ce faire, rajoute une variable appelée « **temps** » et crée un nouveau script qui exécutera les étapes suivantes :

Lorsque l'utilisateur clique sur le drapeau vert
attribuer à la variable « temps » une valeur de 30
répéter jusqu'à ce que le temps = 0
    attendre 1 seconde
    retrancher 1 de la variable « temps »
dire « Perdu »
arrêter tous les scripts

Avant d'écrire ce script, pense à l'endroit où il serait préférable de le placer.
Astuce : est-ce un script qui s'applique à un unique lutin ou à l'ensemble du jeu ?

Activité supplémentaire n° 2 : Le labyrinthe se complique

Maintenant, crée ton propre labyrinthe avec plus d'un chemin vers le centre.
Astuce : Pour cela, il te suffit de créer un nouvel arrière-plan.

Activité supplémentaire nº 3 : Et ma récompense ?

Ajoute à ton jeu de labyrinthe de nouveaux lutins qui seront des bonus pour le joueur qui progresse.

Il faut que ces lutins disparaissent (se cachent) lorsque l'explorateur les touche et que cela entraîne une augmentation de la variable « score ». Veille à placer certains d'entre eux **ailleurs** que sur le chemin le plus court entre le centre et la sortie, pour donner un peu de piment au jeu !

Activité supplémentaire nº 4 : C'est tout vu…

Rajoute du code à ton jeu de labyrinthe pour montrer ou cacher les lutins bonus à des intervalles de temps aléatoires, par exemple entre 1 et 5 secondes (fais des essais pour voir ce qui convient le mieux).

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 ? _____

Explique ta réponse. _____

_____

_____

_____

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

Résumé

Les concepts de l'informatique

Tu as appris d'importantes notions en informatique :

- Ce qu'est un ordinateur
- Les types d'ordinateurs
- Le matériel informatique
- Les logiciels
- La conception de programme à l'aide d'algorithmes
- Les erreurs de programmation

Les structures/commandes de programmation

Dans ce cours, tu as utilisé les fonctionnalités de programmation suivantes :

- La réaction aux événements
- Les instructions conditionnelles
  - si
  - si... sinon
- Les variables comme
  - les scores
  - les chronomètres
- Les boucles
  - « répéter » (finies et infinies)
  - conditionnelles (répéter indéfiniment si)
- La détection de collisions
  - si ... touché
  - si ... couleur touchée

Scratch possède bien d'autres commandes, mais tu en sais désormais suffisamment pour passer à l'étape suivante.

Les caractéristiques de Scratch/...

Les caractéristiques de Scratch

Tu t'es aussi familiarisé(e) avec les caractéristiques suivantes de Scratch :

- Les lutins et la scène
- Les propriétés
    - Les scripts
    - Les costumes et les arrière-plans
    - Les sons
- L'animation
- Les outils graphiques

Tu as désormais toutes les compétences nécessaires pour créer avec Scratch des projets vraiment géniaux !

Un projet avec Scratch

En travaillant en binôme ou en groupe, tu vas maintenant **créer ton propre projet avec Scratch** !

Tu as peut-être déjà quelques idées, mais la création d'un programme passe généralement par une série d'étapes :

1. Analyse
2. Conception
3. Mise en œuvre
4. Tests
5. Documentation
6. Évaluation
7. Maintenance

Ou... **A**ccepte **C**e **M**erveilleux **T**rophée **D**es **E**lèves **M**alins !

L'analyse

En travaillant en binôme ou en petits groupes, **trouve trois idées pour ton projet.** Réfléchis à la manière dont il pourrait être mis en rapport avec d'autres matières que tu étudies.

Réfléchis aux domaines que tu as déjà étudiés...
Ton projet va-t-il utiliser davantage de la musique ou des éléments graphiques ? Vas-tu créer une histoire ? Un jeu ?

Tu trouveras peut-être quelques pistes en visitant la galerie de Scratch (http://scratch.mit.edu).

1. _____

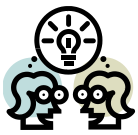
_____

2. _____

_____

3. _____

_____



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.

_____

_____

_____

_____

_____

_____

Conception (interface)

Réalise un scénarimage de ton projet.

Annote ton croquis pour indiquer ce qui se passe et ce que fait chaque lutin.

Conception (code)

Conçois les étapes à coder (algorithme) :

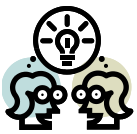
- Réfléchis aux étapes que devra exécuter **chaque lutin ou la scène**. Écris-les en français.
- Réfléchis aux **variables** que ton projet utilisera.

Lutin/Scène	**Algorithme**

Lutin/Scène	**Algorithme**

- Réfléchis aux **variables** que ton projet utilisera.

Nom de variable	**Information stockée**



Mise en œuvre

Maintenant crée ton projet !

- Rassemble les **lutins**, **les costumes**, **les sons** et les **arrière-plans**.
  N'oublie pas de leur donner des noms qui ont du sens.
- Ensuite, crée les **scripts**.
  Assure-toi que tes algorithmes sont à portée de main !

Tests

Teste ton projet pour vérifier qu'il fonctionne.

Laisse tes camarades de classe le tester aussi et note leurs commentaires ci-dessous :

Points positifs : _____

_____

Points négatifs : _____

_____

Décris les erreurs de programmation trouvées (par toi ou par les autres testeurs) et explique comment tu les as résolues :

Erreur de programmation : _____

Solution : _____

_____

Erreur de programmation : _____

Solution : _____

_____

Documentation

Imaginons que tu vas mettre en ligne ton projet sur le site Web de Scratch.

Écris ci-dessous une brève description (50 mots max.) :

- des **principales fonctionnalités** de ton projet et
- **de comment les utiliser**.

Rappelle-toi : tu veux que les gens testent ton projet !

_____

_____

_____

_____

_____

_____

_____

_____

Écris la description et conserve-la dans les notes de ton projet **(Fichier→Notes de projet...).**

Évaluation

Quel résultat as-tu obtenu **comparé à ce que tu avais initialement planifié** ?

_____

_____

_____

Quelles **erreurs** as-tu commises au cours de l'élaboration du projet ?

_____

_____

_____

Si tu devais recommencer depuis le début, que ferais-tu **différemment** ?

_____

_____

_____

Reviens à ton **code**.
Aurais-tu pu prendre un raccourci pour rendre ton code plus **élégant** ?

_____

_____

_____

Maintenance

Quelles **fonctionnalités pourrais-tu rajouter** pour améliorer ton projet ?

_____

_____

_____

_____

_____

_____

Félicitations !

Tu es arrivé(e) au bout de cette introduction à l'informatique avec Scratch !

Rappelle-toi que tu peux télécharger et utiliser Scratch chez toi, alors il n'y a aucune raison pour que s'achève ici ton aventure en tant que programmeur.

http://scratch.mit.edu