ÉDUCATION NATIONALE
Luxembourg

9TE / PO Informatique

Logique Informatique

Programmation avec

SCRATCH

Version 2.0 du 23 mars 2015

ÉDUCATION NATIONALE
Luxembourg

9TE / PO Informatique

Logique Informatique

Programmation avec

Introduction

Scratch est un langage de programmation qui facilite l'apprentissage de la programmation de l'ordinateur. Il te permet de créer des histoires interactives et de développer des jeux et des animations.

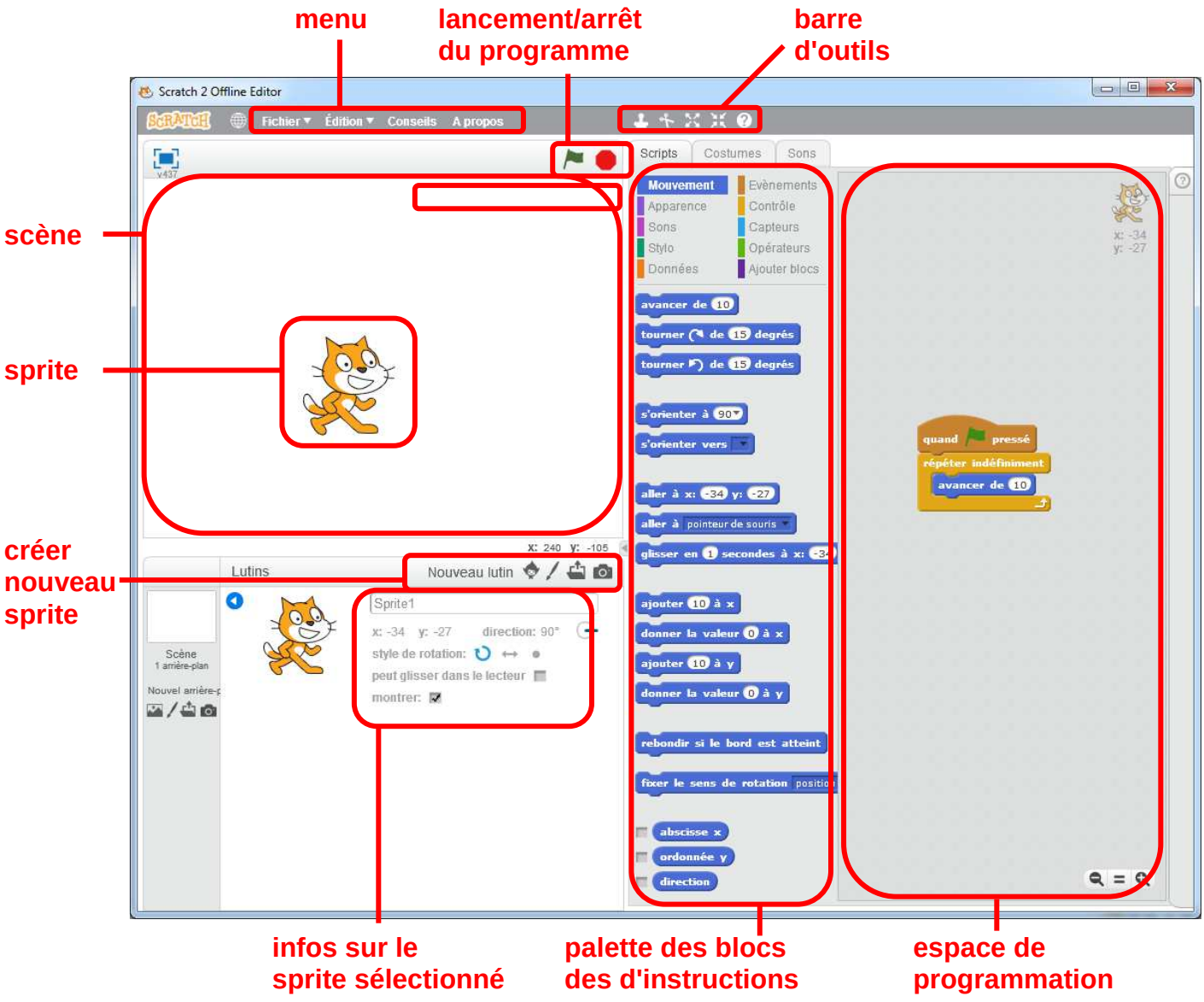
Tu peux télécharger le programme gratuitement à la page web scratch.mit.edu. Tu trouves également sur ce site une multitude d'informations sur Scratch (des modes d'emploi, des tutoriels vidéo, des cartes Scratch, la foire aux questions etc.). Et finalement tu peux t'inscrire sur ce site et publier tes propres projets Scratch.

Ce manuel va t'introduire étape par étape dans la programmation avec Scratch.

A. L'environnement Scratch

Ce chapitre te permet de découvrir l'environnement dans lequel tu développeras tes projets Scratch.

A.1. L'interface du programme



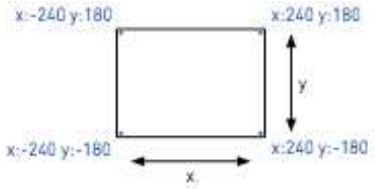
A.2. Eléments principaux d'un projet Scratch

A.2.1 La scène

La *scène* est le lieu où se déroulent les histoires, jeux et animations.

Les *sprites* (objets, voir plus loin) se déplacent sur une scène et agissent les uns avec les autres.

La scène a une largeur de 480 unités et une hauteur de 360 unités. Elle possède un système de coordonnées XY dont le point 0,0 se trouve au centre de la scène.



A.2.2 Les sprites

Un projet Scratch est composé d'un certain nombre d'**objets** appelés **sprites**. Sprite veut dire littéralement ogre mais désigne ici un graphique numérique. Dans la version française du programme il est appellé **lutin**.

Tu peux modifier l'aspect d'un sprite en lui donnant une apparence selon ton choix, p.ex. celle d'une personne, d'un train, d'un papillon etc.

Un sprite possède un nom. Scratch en donne automatiquement un (Sprite1, Sprite2...), mais pour que tu puisses mieux te repérer dans ton projet, il t'est conseillé de leur donner des **noms significatifs**.

A.2.3 Les instructions

Tu peux faire exécuter des ordres à un sprite, lui dire de bouger, de jouer de la musique ou de le faire agir avec d'autres sprites. Pour communiquer une **instruction** à un sprite, tu dois faire glisser les **blocs graphiques** dans l'espace de programmation.

Ces blocs d'instructions sont regroupés dans des palettes d'instructions. Chaque instruction possède une couleur selon son type et se trouve dans la palette correspondante à son type. Ainsi toutes les instructions servant au déplacement d'un sprite ont la couleur bleu foncé et sont regroupées dans la palette ***Mouvement***.

Voici les différentes palettes d'instructions :

En plus les différents blocs d'instructions se distinguent également par leur forme.

A.2.4 Les programmes

Un programme est une suite d'instructions qui sont exécutées l'une après l'autre. Pour développer un programme Scratch tu dois assembler les blocs d'instruction nécessaires comme les pièces d'un puzzle. Un sprite peut avoir plusieurs programmes. Un autre mot pour un programme Scratch est **Script**.

Un double clic sur ton programme permet d'exécuter les blocs l'un après l'autre, du haut vers le bas.

Si tu n'as plus besoin d'un bloc d'instruction, il te suffit de le faire glisser de nouveau dans une palette d'instructions.

Exercice A-01 Mon premier programme 15 min

Objectif :

Développer un programme Scratch.

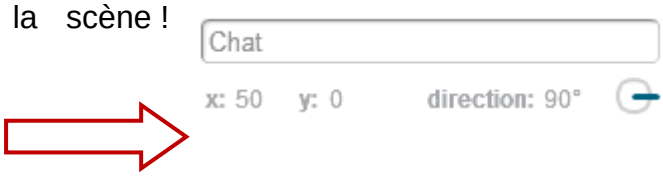
Description

a) Lance le programme Scratch.

b) Renomme le sprite en lui donnant le nom de ***Chat***.

c) Fais glisser le bloc 

vers l'espace de programmation. Effectue un double clic sur le bloc. Qu'est-ce qui ce passe ? Observe la position du sprite sur la scène ! Explique !



d) Développe le programme suivant :

e) Lance le programme par un double clic.

f) Arrête l'exécution du programme.

g) Y a-t-il un autre moyen de lancer le programme ?

h) Modifie le style de rotation du sprite et expérimente avec les différentes possibilités :

i) Explique les différents styles de rotation :

_____

_____

_____

Lequel des styles de rotation est le plus adapté pour ce projet ?

_____

j) Sauvegarde le projet sous le nom de ***Exercice A-01***.

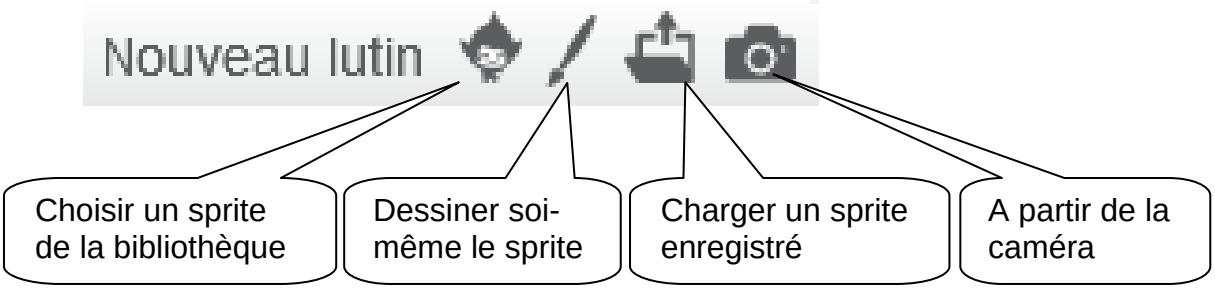
A.1. Modifier les sprites

Pour **déplacer** un sprite à un autre endroit de la scène, il suffit de le glisser avec la souris en ayant le bouton de la souris appuyé.

Tu peux **dupliquer**, **supprimer** un sprite ainsi qu'**agrandir** et **réduire** sa taille : l'option choisie est effectuée par un clic dans la barre d'outils ci-contre.

L'aiguille noire à droite te permet de faire **pivoter** le sprite à l'aide de la souris.

Il y a plusieurs possibilités pour **ajouter** un sprite au projet :



Exercice A-02 Créer d'autres sprites

Objectif :

Créer de nouveaux sprites dans un projet et les modifier.

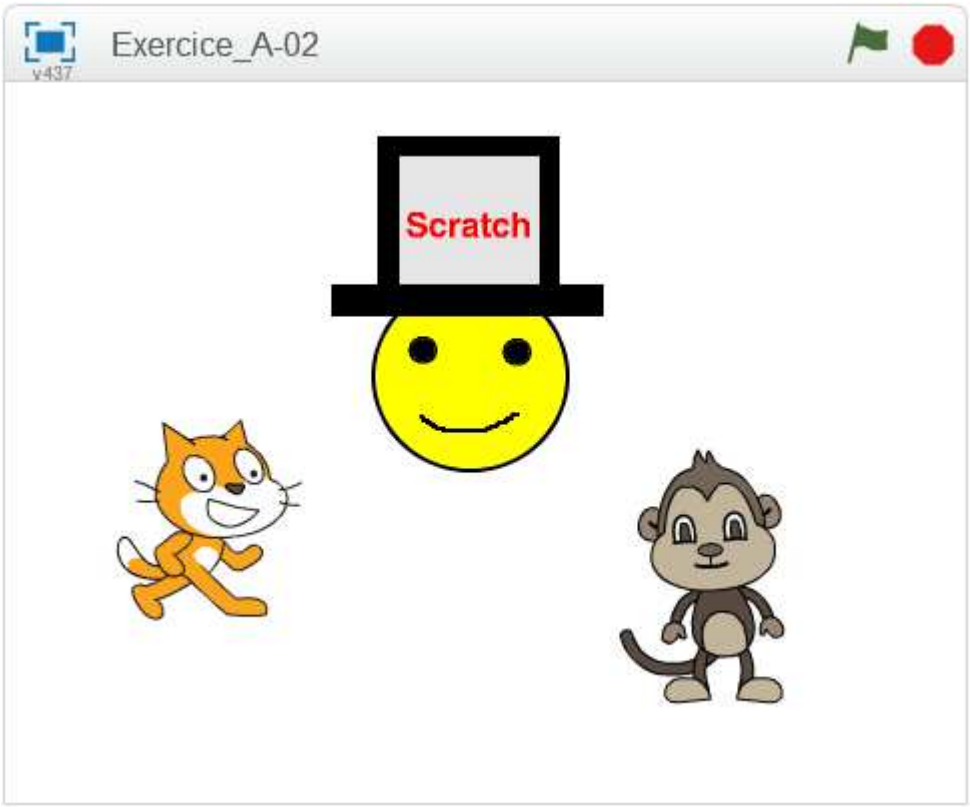
Description

a) Ouvre le projet de l'***Exercice A-01*** et sauvegarde-le sous le nom de ***Exercice A-02***.

b) Crée un nouveau sprite et place-le sur la scène comme indiqué sur le schéma ci-dessous. La souris peut être chargée d'un fichier, le smiley doit être dessiné dans l'éditeur graphique.

c) Renomme les sprites.

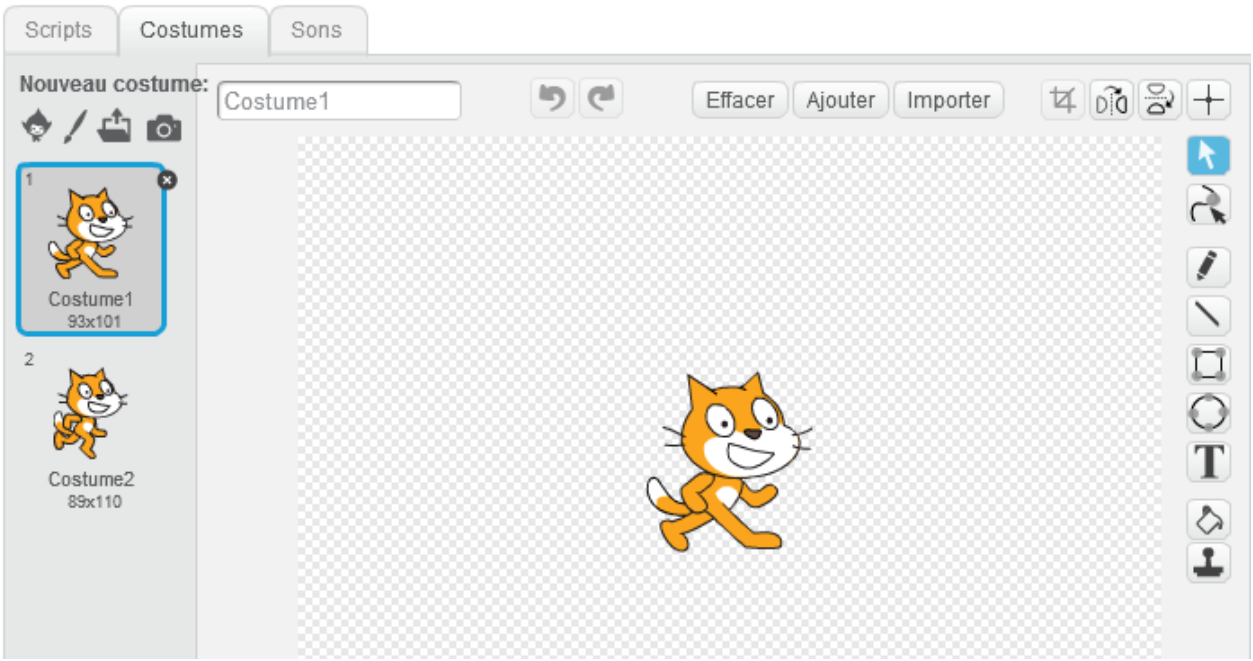
a) Sauvegarde le projet.



A.2. Les sprites ont des costumes

Un sprite peut posséder un ou plusieurs costumes qui lui permettent de changer d'apparence.

Tu peux voir les costumes d'un sprite quand tu sélectionnes d'abord le sprite et tu cliques ensuite sur l'onglet intitulé ***Costumes***.



Le chat a p. ex. deux costumes. L'habit actuel est mis en évidence. Pour basculer vers un autre costume, il suffit de cliquer sur son aperçu.

Il y a trois possibilités pour créer un nouveau costume :

- Clique sur  pour dessiner toi-même le costume dans l'éditeur Paint.
- Clique sur  pour charger une image graphique enregistrée sur le disque dur.
- Clique sur  pour charger une image à partir de la bibliothèque Scratch.
- Si ton ordinateur est équipé d'une webcam, alors clique sur  pour photographier de nouveaux costumes.

Scratch est capable de traiter de nombreux formats graphiques : JPG, GIF, BMP et PNG.

Tu peux modifier l'ordre des costumes en les déplaçant à l'aide de la souris.

A droite dans l'image se trouve une barre d'outils qui sert à éditer le costume actuel en mode vectoriel.

Les **graphiques vectoriels** sont composés de formes géométriques. Leur affichage est plus net que celui des graphiques bitmaps et ils peuvent être agrandis sans perte de qualité.

Si tu transforme le costume en graphique bitmap, alors tu trouves une autre barre d'outils à droite, celle qui sert à retravailler les graphiques bitmap.

Les **graphiques bitmap** sont composés de pixels (points de couleur). Ils se prêtent bien à l'affichage d'images complexes comme les photos.

Exercice A-03 Les costumes

Objectif : Modifier des costumes.

Description

a) Ouvre le projet de l'***Exercice A-01*** et sauvegarde le sous le nom de ***Exercice A-03.***

b) Charge le sprite ***Boy3 Walking*** à partir du disque dur et place-le sur la scène comme indiqué sur le schéma ci-dessous.

c) Nomme le sprite ***Garçon***.

d) Combien de costumes le garçon possède-t-il ?

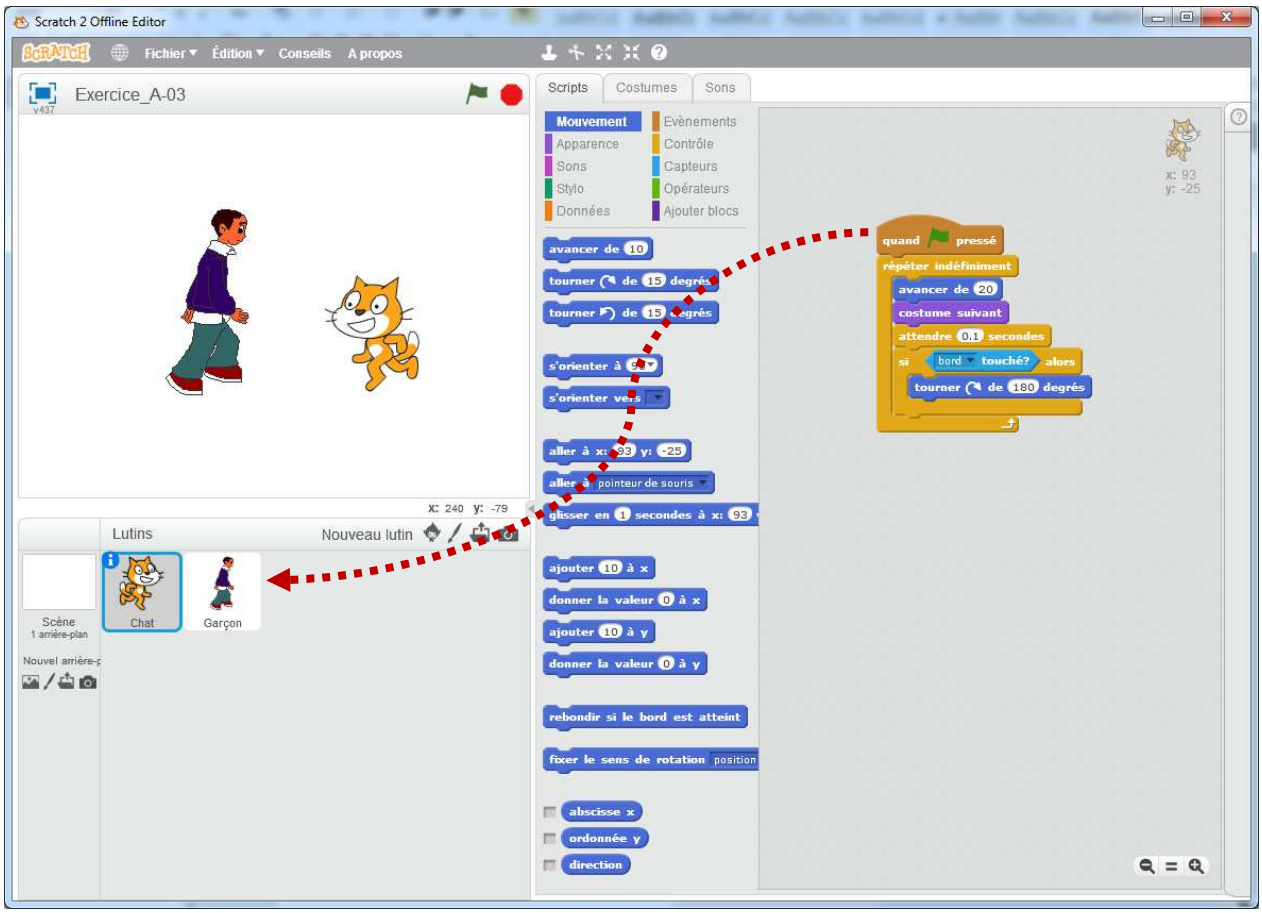
_____

e) Crée d'autres costumes en important les fichiers ***boy4-walking-b***, ***boy4-walking-c***, ***boy4-walking-d*** et ***boy4-walking-e***.

f) Copie le programme du sprite ***Chat*** dans le sprite ***Garçon*** (tire le programme de l'espace de programmation vers le sprite ***Garçon*** dans la liste des sprites comme indiqué sur le schéma).

g) Lance maintenant les deux programmes (clic sur drapeau vert).

h) Sauvegarde le projet.



A.3. La scène

De la même manière que les sprites changent d'apparence en basculant vers un autre costume, la scène peut changer d'apparence en basculant vers un autre **arrière-plan**.

Il est en plus possible de programmer la scène.

Pour modifier l'arrière-plan de la scène, clique sur l'icône scène qui se trouve à gauche de la liste des sprites.

Exercice A-04 La scène – les arrière-plans

Objectif :

Modifier l'arrière-plan de la scène.

Description

a) Ouvre le projet de l'***Exercice A-03*** et sauvegarde-le sous le nom de ***Exercice A-04***.

b) Modifie l'arrière-plan comme affiché ci-dessous.

c) Sauvegarde le projet.

A.4. Documentation du projet

Il est toujours conseillé de documenter son projet de manière suffisante. Ainsi quand tu voudras le modifier plus tard, une bonne documentation t'aidera à mieux t'y retrouver.

Les **Commentaires** te permettent d'annoter certaines parties de tes programmes. Cette documentation s'impose surtout pour des programmes complexes dans lesquels il est plus difficile de se retrouver.

Clique avec le bouton droit de la souris dans l'espace de programmation, choisis ***Ajouter un commentaire*** dans le menu contextuel, puis saisis le commentaire.

Exercice A-05 La documentation du projet

Objectif :

Documenter un projet

Description

a) Ouvre le projet de l'***Exercice A-04*** et sauvegarde-le sous le nom de ***Exercice A-05***.

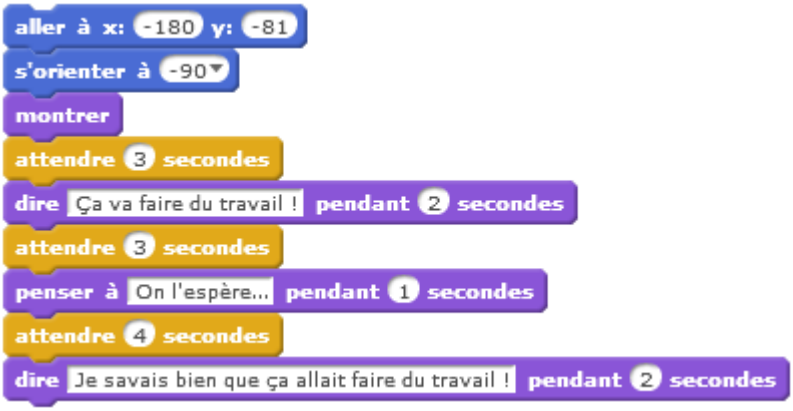
b) Essaie de comprendre comment fonctionne le programme du sprite ***Chat*** et saisis un commentaire pour chaque bloc d'instruction.

c) Sauvegarde le projet.

B. Développer un projet Scratch à l'aide de séquences

La séquence est la forme la plus simple pour la structure d'un programme. Plusieurs instructions sont exécutées l'une après l'autre.

Voici un exemple d'une séquence Scratch :



Exercice B-01

Objectif :

Réaliser un projet à l'aide de séquences.

Description

a) Crée un nouveau projet et sauvegarde-le sous le nom de ***Exercice B-01***.

b) Il faut que le chat

- se rende à la position x=0, y= -100
- regarde à gauche
- dise « Coucou ! » pendant 2 secondes.

c) Sauvegarde le projet.

Exercice B-02 "Le corbeau et le renard!"

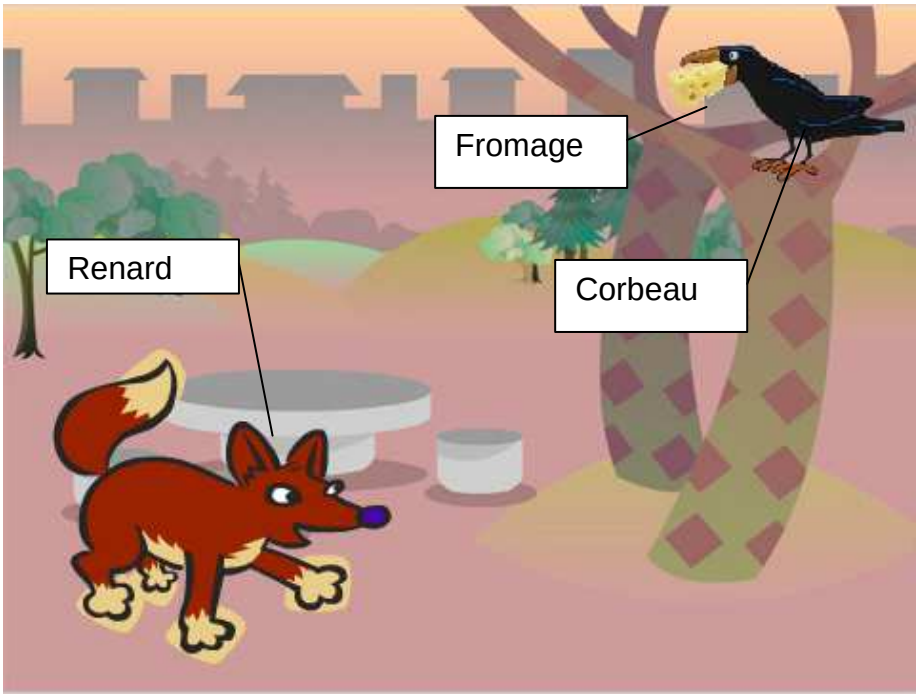
Objectif : Créer un projet à l'aide d'une séquence d'instructions.

Description

a) Développe un projet avec les sprites ***Renard***, ***Fromage*** et ***Corbeau*** (voir image). Les deux derniers (***Fromage*** et ***Corbeau***) n'existent pas dans la bibliothèque. Tu peux les dessiner toi-même ou bien les charger d'Internet.

b) ***Renard*** doit réciter phrase par phrase (l'une après l'autre) le texte suivant :

Et Bonjour Monsieur du Corbeau,
Que vous êtes joli! Que vous me semblez beau!
Sans mentir, si votre ramage se rapporte à votre plumage
Vous êtes le phénix des hôtes de ces bois



c) Le ***Corbeau*** dit ensuite le texte « Merci », après quoi le ***Fromage*** tombe par terre.

d) Condition de départ : Quand on clique à nouveau sur le drapeau vert, le ***Fromage*** doit se trouver à sa position initiale, c.-à-d. dans la bouche du ***Corbeau***. Autrement dit, il faut placer au début le Fromage dans cette position à l'aide d'une instruction "*aller à*". On parle dans ce cas d'Initialisation. A partir de maintenant il faudra tenir compte de ces réflexions dans tous les exercices.

e) Sauvegarde le projet sous le nom de ***Exercice B-02***.

C. Réagir à des événements

Les événements sont symbolisés par des blocs appelés ‚chapeaux'. Ces blocs possèdent une bordure supérieure arrondie comme p.ex. :

Le programme accolé est exécuté quand on clique sur le drapeau vert.

Le programme accolé est exécuté quand on clique sur le sprite sélectionné.

Le programme accolé est exécuté quand on appuie sur la touche indiquée.

Un tel bloc forme le sommet d'une pile de blocs assemblés. Il attend qu'un certain événement ait lieu (p.ex. on appuie sur une touche indiquée). Lors de cet événement les blocs en-dessous du bloc chapeau sont exécutés.

Exercice C-01

Objectif :

Réagir au clic de la souris

Description

a) Crée un projet représentant une bouteille pleine et un verre vide. Vu que ce sprite n'existe pas dans la collection, il te faut le dessiner ou le charger d'Internet. AIDE : La bouteille et le verre doivent faire partie du même sprite. Celui-ci doit posséder au moins trois costumes.

b) Quand on clique sur le sprite, alors la bouteille se vide à fur et à mesure tandis que le verre se remplit. Une fois la bouteille vide, le tout recommence dès le début.

c) Sauvegarde le projet sous le nom de ***Exercice C-01***.

Exercice C-02

Objectif :

Réagir aux événements de clavier

Description

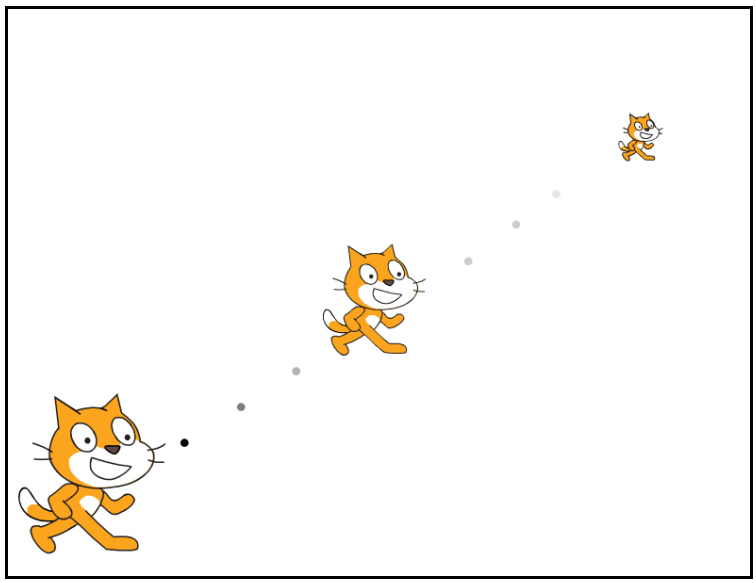
a) Crée un projet dans lequel le chat se déplace selon la touche flèche appuyée (gauche/droite/haut/bas).

b) Sauvegarde le projet sous le nom de ***Exercice C-02***.

Extension pour avancés

Touche « H » : Le chat se déplace en diagonal, d'en bas à gauche vers le haut à droite, tout en devenant de plus en plus petite. On crée ainsi un effet de profondeur.

Touche « V » : Le chat se déplace en diagonal, d'en haut à droite vers le bas à gauche, tout en devenant de plus en plus grande.



D. La boucle infinie

Dans cette unité, nous aborderons le bloc de contrôle intitulé "*répéter indéfiniment*". Il nous permet de répéter des actions indéfiniment. En termes de programmation, nous parlons de "*boucle infinie*".

Si nous insérons les blocs d'instructions des exercices précédents dans cette boucle infinie, nous allons voir qu'ils seront exécutés jusqu'à ce qu'on clique sur l'icône stop rouge.

Exercice D-01 La chauve-souris infatigable

Objectif :

Utiliser la boucle infinie.

Description

a) Crée un projet avec le sprite et la scène ci-dessous :

b) Dispositions au lancement du programme :

- La chauve-souris est placée au milieu de l'écran et est orienté vers la droite.
- Le style de rotation "gauche-droite" est à choisir à l'aide d'un bloc d'instruction.

c) La chauve-souris dispose de deux costumes qui permettent de la voir voler. Quand nous cliquons sur le drapeau départ nous pouvons l'observer voler sur la scène.

d) Sauvegarde le projet sous le nom de ***Exercice D-01***.

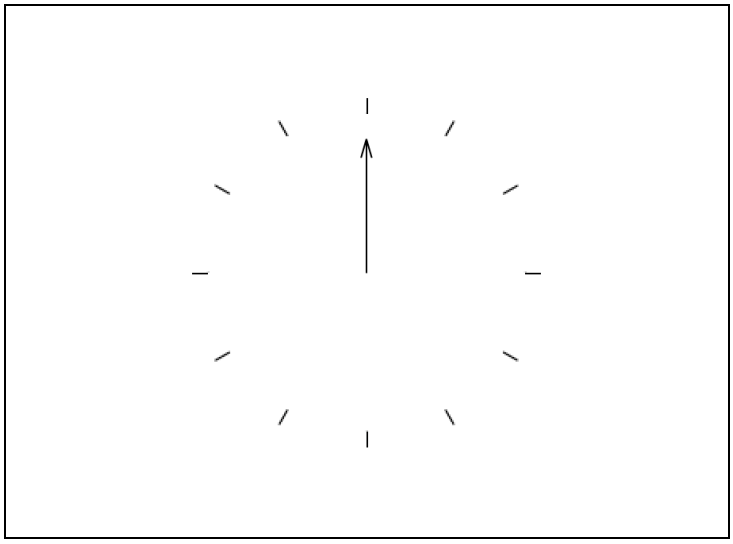
Exercice D-02 Une montre basique

Objectif :

Utiliser la boucle infinie.

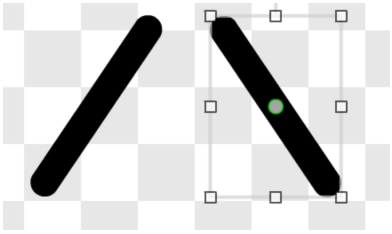
Description

a) Crée un projet avec un nouveau sprite représentant la trotteuse d'une montre. Crée ce nouvel objet ***Aiguille*** à l'aide du logiciel intégré de traitement de graphiques vectoriels (voir plus loin).



Il placer soigneusement le point de rotation (du costume) de l'Aiguille. Afin de réaliser la tête symétrique de l'Aiguille il est conseillé de travailler avec les outils "*Dupliquer*" et "*Retournement horizontal*".

Construction de la pointe de l'aiguille



original

dupliqué et tourné

assemblé

b) Dispositions au lancement du programme :

L'aiguille est placée au milieu de l'écran et est orienté vers le haut.

c) L'aiguille est animée. Il accomplit une rotation complète (360°) en 60 secondes. L'angle de rotation est logiquement de 360 : 60 = 6°. L'aiguille n'arrête jamais de tourner dès que le drapeau vert est appuyé.

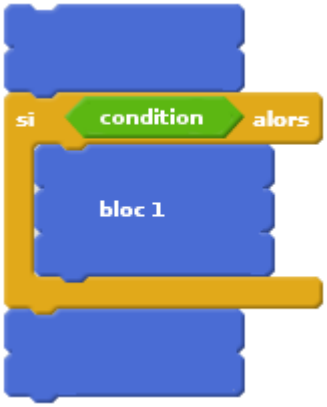
d) Afin d'améliorer l'affichage il faut encore dessiner le cadran. A cela sert la séquence d'instruction suivante. La subdivision du cadran est contrôlée à partir du *point (0 ; 0)*.

e) Sauvegarde le projet sous le nom de ***Exercice D-02***.

E. Prendre des décisions

Tu as certainement remarqué qu'il est souvent nécessaire, lors de la programmation de prendre une décision (comme dans la vie réelle). Ainsi ton sprite devra faire une chose ou une autre selon les circonstances qu'il rencontre.

Voici les blocs utilisés pour réaliser ce choix :



si *la condition est vérifiée*,

alors *le bloc d'instructions* **bloc 1** *est exécuté.*

si *la condition est vérifiée*,

alors *le bloc d'instructions* **bloc 1** *est exécuté,*

sinon *le bloc d'instructions* **bloc 2** *est exécuté.*

La condition doit être ou bien "vraie" ou bien "fausse" pour faire exécuter les instructions de la partie "si".

Voici deux types de conditions, il y en a encore d'autres types :

te permet de tester si un sprite ou le bord a été touché

te permet de tester si une couleur a été touchée.

Tu trouves ces blocs dans la palette '**Capteurs**'.

Exercice E-01 Cherche le repas !

Objectif : Le chat cherche son repas

Description :

a) Dispositions au lancement du programme :

Chat (c'est le nom du sprite) :

- se trouve en bas à gauche de l'écran
- est orienté vers la droite
- le style de rotation est "gauche-droite"

Repas (c'est aussi le nom du sprite) :

- se trouve en bas à droite de l'écran
- sa taille est mise à la moitié de sa taille d'origine
- le repas est caché (invisible)

b) Le chat a très faim, (il est vorace). Pendant qu'il court dans la pièce il pense « Mais où est passé mon repas ? » Le repas n'apparait qu'après avoir appuyé sur la touche espace.

c) Quand le chat trouve son repas le chat dit « Le voilà enfin ! » et le programme s'arrête.

d) Sauvegarde le projet sous le nom de ***Exercice E-01***.

Exercice E-02 Free the crab!

Objectif : Il faut libérer le crabe de sa cage.

Description :

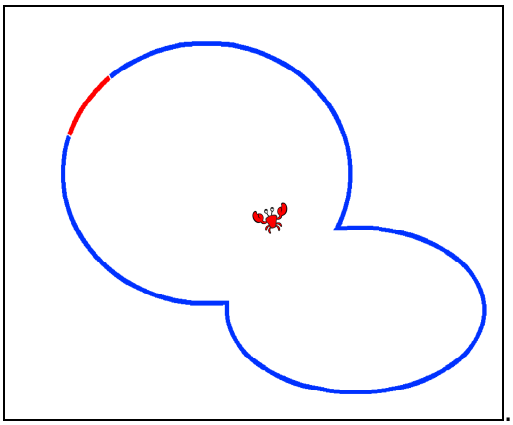
a) Dispositions au lancement du programme :

Crabe (c'est le nom du sprite) :

- se trouve au milieu de l'écran
- est orienté vers une direction aléatoire

b) Utilise le sprite ***Crabe***  de la collection Scratch

c) Crée pour la scène un arrière-plan ("cage") qui possède un bord d'une couleur. La sortie est d'une deuxième couleur comme dans l'exemple ci-dessous :



d) Le crabe essaie de s'échapper de sa cage. Quand il rencontre la couleur (bleue) du bord il pense « Hmmm… », se tourne un peu et essaie à nouveau. Quand il rencontre la couleur (rouge) de la sortie, il s'écrie « enfin libre » et le programme s'arrête deux secondes plus tard.

e) Sauvegarde le projet sous le nom de ***Exercice E-02***.

Extensions pour avancés:

f) Les 2 costumes du crabe ne se distinguent que peu. Dessine un costume plus rigolo, par exemple avec des yeux qui sortent. 

g) Tu veux tricher et influencer la direction.

Tu peux faire tourner le crabe vers sa droite en appuyant sur la "**flèche droite**". La "**flèche gauche**" te permet de le faire tourner vers sa gauche.

h) Sauvegarde le projet sous le nom de ***Exercice E-02***.

F. Boucles à condition

Jusqu'à ce point, tu t'es déjà beaucoup exercé en Scratch et tu as appris beaucoup de choses. Tu sais programmer des séquences d'instructions et des boucles infinies et tu sais utiliser des conditions.

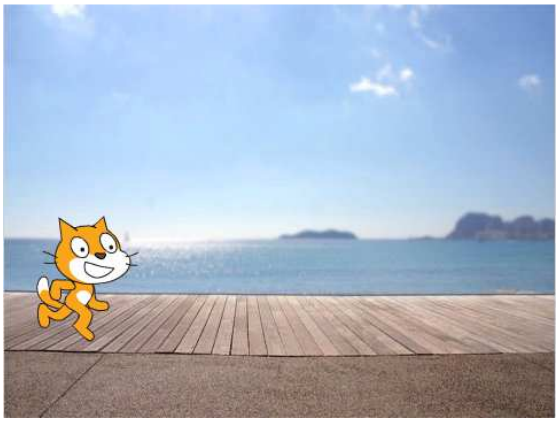
Savais-tu qu'on pouvait parfois combiner les deux : boucles infinies et conditions ? En le faisant tu obtiens ceci :

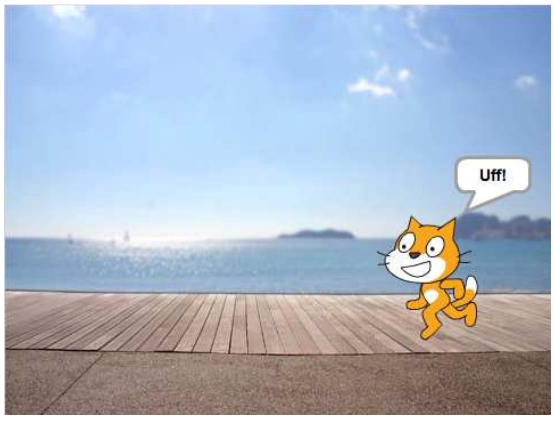
Dans ce cas, le programme regarde d'abord si la condition est vérifiée. Si oui, le contenu de la boucle est ignoré et le programme continue avec l'instruction qui suit la boucle. Sinon, elle exécute la séquence intérieure de la boucle (instruction 1, instruction 2, ...), puis contrôle à nouveau si la condition est vérifiée. la séquence intérieure est exécutée jusqu'à ce que la condition soit vérifiée. Alors l'exécution se poursuit avec l'instruction qui suit la boucle.

Exercice F-01 Jogging au bord de l'eau

Objectif :

Le chat doit poursuivre son jogging jusqu'à ce que la touche espace soit appuyée.





Description :

a) Le chat fait son jogging par des allers-retours jusqu'à ce que la touche espace soit appuyée. Elle s'arrête ensuite et dit "Ouf !". Utilise la boucle ***répéter jusqu'à*** !

b) Sauvegarde le projet sous le nom de ***Exercice F-01***.

c) Essaye de résoudre cet exercice en utilisant la boucle ***répéter indéfiniment***. Quelle solution est la meilleure ? Justifie ta réponse !

Exercice F-02 Liberté au chat !

Objectif : Libérer le chat de sa cage

Description :

a) Le chat court de gauche à droite et de droite à gauche (c.-à-d. il court tant qu'il touche la prison). Il change de direction en touchant le bord gris de la prison. Dessine toi-même le sprite de la cage.

b) Quand on appuie sur la touche 'espace', la cage s'ouvre et le chat te remercie pour sa liberté.

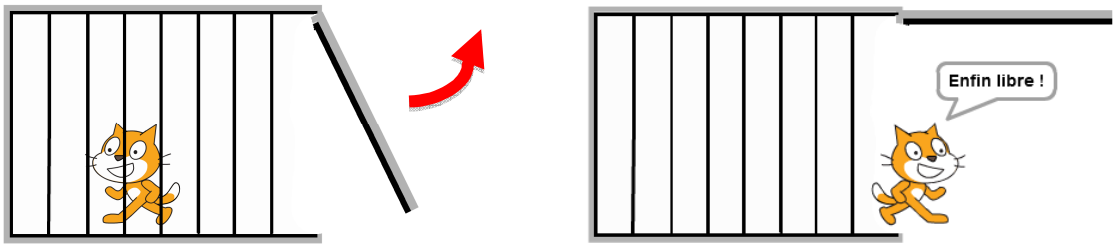
c) Assure-toi qu'au début du programme, que la cage se trouve en avant-plan sur l'écran, que le chat se trouve dans la cage et que la cage soit fermée.

d) Sauvegarde le projet sous le nom ***Exercice F-02***.

Extension :

e) Pendant que le chat court dans sa cage il pense tous les 10 secondes "Comment sortir d'ici ?"

Extension pour avancés :



f) Quand la touche espace est appuyé, alors la porte de la cage s'ouvre lentement.

g) Quand le chat a quitté sa cage il te remercie pour sa liberté.

Aide : Pour résoudre cette extension tu peux utiliser les conditions complexes (voir chapitre I.2.2).

G. Envoyer et recevoir des messages

Dans certaines situations, un sprite aimerait exécuter une action (programme) à un autre sprite.

Scratch le rend possible par l'envoi et la réception de messages.

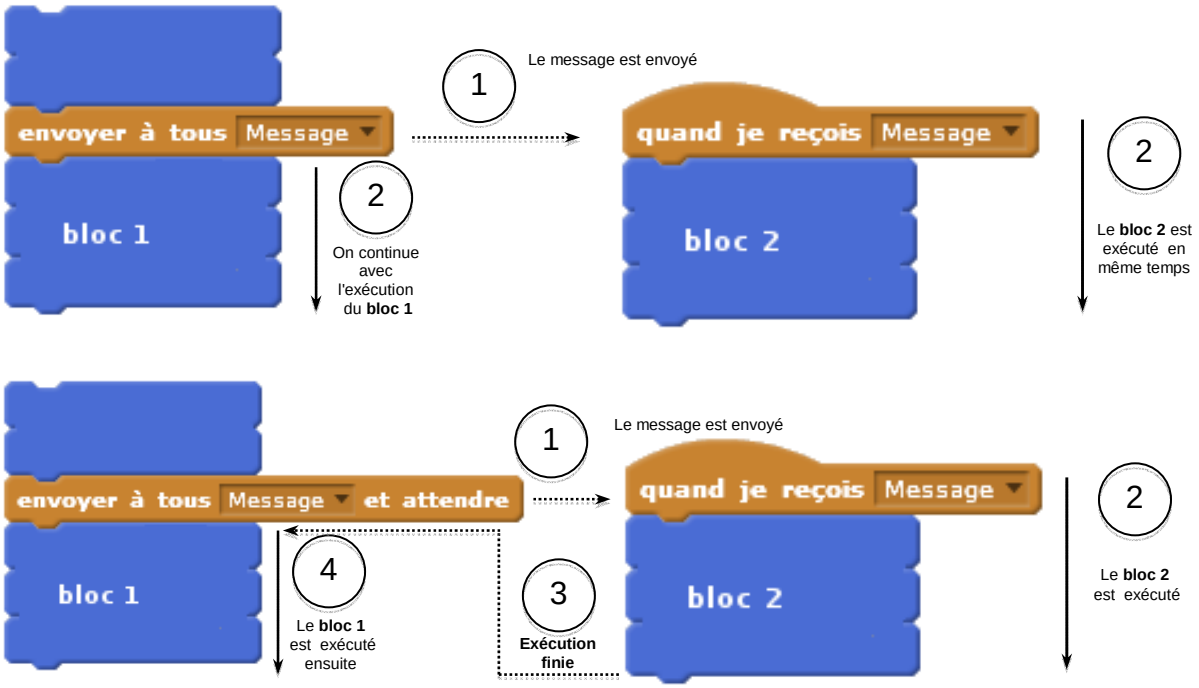
Il existe deux instructions qui servent à envoyer un message :

Envoie un message à tous les sprites et la scène, ce qui peut les amener à exécuter une action. Le programme continue tout de suite son exécution sans attendre que les autres sprites aient fini leurs actions.

Envoie un message à tous les sprites et la scène, ce qui peut les amener à exécuter une action. Le programme attend de continuer son exécution jusqu'à ce que les autres sprites aient fini leurs actions.

Lorsqu'un un sprite ou la scène reçoit le message alors la séquence d'instructions qui se trouve sous le bloc chapeau est exécutée.

Les schémas suivants illustrent la différence entre les deux instructions d'envoi :



Exercice G-01 « Vite sur le tapis ! »

Objectif :

Envoyer et recevoir des messages.

Description

a) Crée un projet avec les sprites suivants : *Mamie, Toutou* et *Tapis*

Dispositions au début du programme :

- Le Toutou se trouve en bas à gauche de l'écran.

b) Toutou devrait avoir au moins deux costumes pour le voir se dandiner.

c) Quand on clique sur la Mamie, alors elle dit « Vite sur le tapis ! » et Toutou doit se rendre à sa corbeille.

Programme la ***Mamie*** et ***Toutou*** en utilisant les instructions suivantes.

d) Sauvegarde le projet sous le nom de ***Exercice G-01***.

e) Ajoute l'action suivante :
Quand Toutou arrive à sa corbeille alors la mamie dit « Bon Toutou ! »

Exercice G-02 Une aiguille pour les minutes

Objectif :

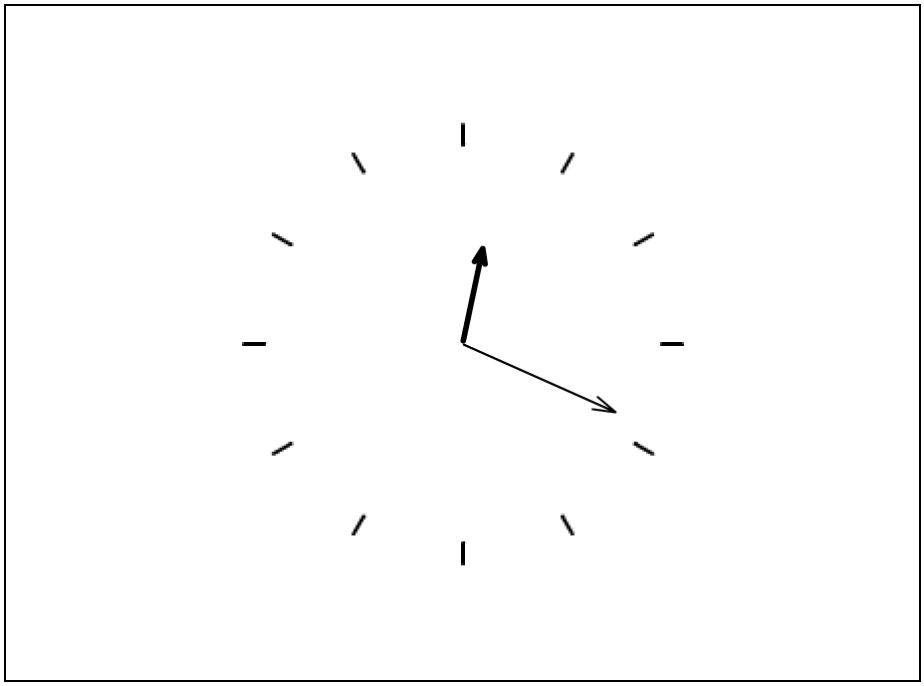
Envoyer et recevoir des messages.

Description

a) Ouvre le projet de l'***Exercice D-02*** et sauvegarde-le sous le nom de ***Exercice G-02***.

b) Dessine l'aiguille pour les minutes et ajoute-le à la montre.

c) **Fonctionnement** : chaque fois que l'aiguille des secondes pointe sur le 12, il envoie un message à l'aiguille des minutes pour que celui-ci avance d'un cran.



d) Sauvegarde le projet.

Pour avancés:

e) Sauvegarde le projet ***Exercice G-02*** sous le nom de ***Exercice G-03***. Ajoute une aiguille pour les heures. N'oublie pas de sauvegarder.

H. Contrôle à l'aide de la boucle

Dans ce chapitre, tu apprendras comment tu peux contrôler l'exécution répétée d'instructions à l'aide d'une boucle.

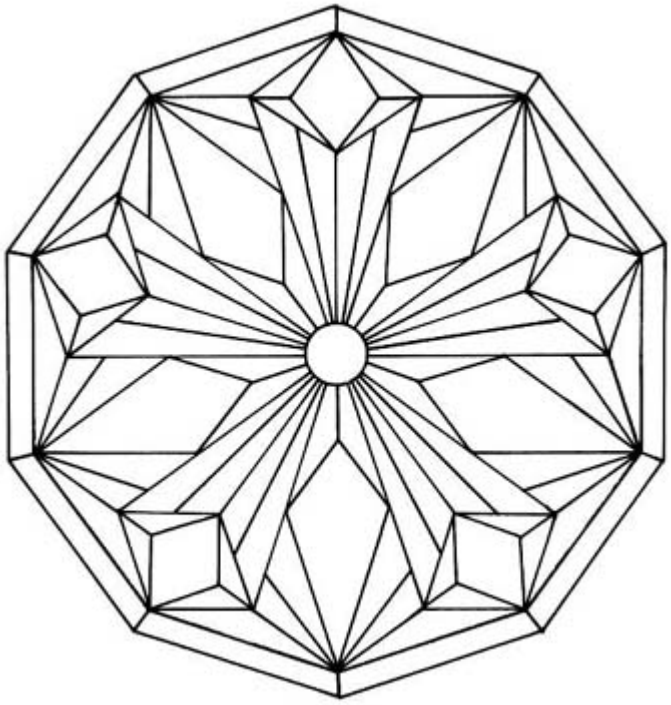
Pour réaliser cette Unité, il est important de rafraichir nos notions de géométrie (*calcul d'angles* ; *symétrie centrale ; rotation ; système de coordonnées, quatre quadrants*).

Mandala

*Le mot **Mandala** (du sanscrit) veut dire 'cercle' et désigne une forme symbolique circulaire ou carrée avec un centre. A l'origine il a été utilisé uniquement dans un contexte religieux (Wikipédia).*

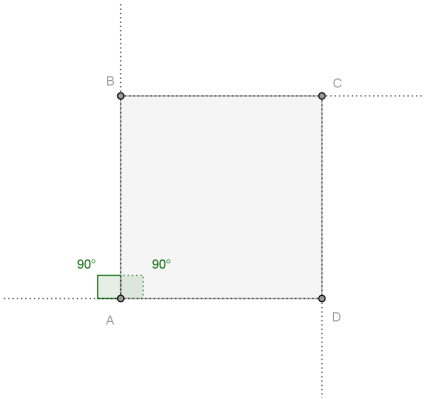
Aujourd'hui le mandala sert plutôt pour les feuilles de coloriage pour enfants et adolescents.

Dans cette unité tu programmeras un mandala… plus simple que l'exemple ci-contre.



Le mandala ci-dessus est composé
de parties égales (ici 5 ou plutôt 2x5) autour du centre.

H.1. Dessiner le carré



Le carré est dessiné autour du centre de la scène (point 0;0).

Un carré possède …… côtés ……. …….. et …… angles ……. …….. La somme des angles d'un carré est de …… Un angle mesure exactement ……., il s'agit d'un angle ……..

Exercice H-01 Un carré

a) Dessine un carré d'une longueur de 100. A la fin du script le sprite doit de nouveau pointer vers la direction de départ.

Blocs d'instruction utilisés :

b) Que remarques-tu ?

_____

_____

_____

c) Sauvegarde le projet sous le nom de ***Exercice H-01***.

Exercice H-02 Un carré 2

a) Dessine de nouveau un carré d'une longueur de 100 en utilisant maintenant la boucle 

.

Condition de départ :

- Le carré est dessiné quand on appuie sur la touche '4'.
- Le stylo doit être positionné à la position (0 ; 0).
- Toutes traces d'un dessin précédent doivent être effacées.

b) Sauvegarde le projet sous le nom de ***Exercice H-02***.

H.2. Polygones réguliers

"Un polygone est dit régulier lorsque ses côtés et ses angles sont tous égaux."[1]

Dessine des polygones réguliers après avoir analysé et compris leurs caractéristiques.

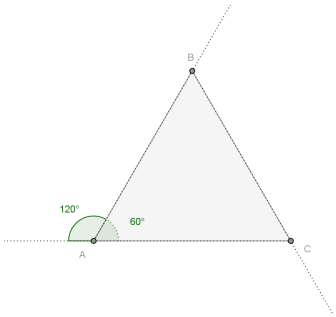
Exercice H-03 Triangle équilatéral

a) Ouvre le projet de l' ***Exercice H-02*** et sauvegarde-le sous le nom de ***Exercice H-03***.

b) Duplique le script présent et modifie-le afin de dessiner un triangle équilatéral (de longueur 100). Fait d'abord l'analyse du triangle pour déterminer l'angle à appliquer.

Conditions de départ :

- Le carré est dessiné quand on appuie sur la touche '3'.
- Le stylo doit être positionné à la position (0 ; 0).
- Toutes traces d'un dessin précédent doivent être effacées.



c) Sauvegarde le projet.

Exercice H-04 Polygone à n sommets

Réfléchis, en te basant sur les connaissances acquises lors des exercices précédents, comment trouver les valeurs qui manquent dans le tableau ci-dessous. Il s'agit dans chaque ligne de créer un polygone régulier. Tu n'as pas besoin de Scratch pour cet exercice.

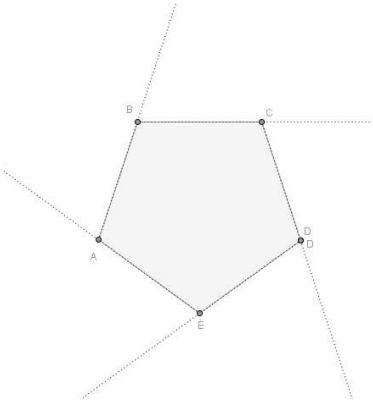
Figure géométrique	Nombre de côtés	Angle (°)	Somme des angles (°)
Triangle			
Carré			
Pentagone			
Polygone à n sommets			

[1] http://fr.wikipedia.org/wiki/Polygone

Exercice H-05 Pentagone

a) Ouvre le projet de l'***Exercice H-03*** et sauvegarde-le sous le nom de ***Exercice H-05***.

b) Ajoute un script pour dessiner un pentagone de longueur 100.



Conditions de départ :

- Le carré est dessiné quand on appuie sur la touche '5'.
- Le stylo doit être positionné à la position (0 ; 0).
- Toutes traces d'un dessin précédent doivent être effacées.

c) Sauvegarde le projet.

Exercice H-06 Hexagone régulier

a) Ouvre le projet de l'***Exercice H-05*** et sauvegarde-le sous le nom de ***Exercice H-06*** ab.

b) Ajoute un script pour dessiner un hexagone régulier de longueur 100.

Conditions de départ :

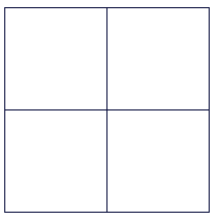
- Le carré est dessiné quand on appuie sur la touche '6'.
- Le stylo doit être positionné à la position (0 ; 0).
- Toutes traces d'un dessin précédent doivent être effacées.

c) Sauvegarde le projet.

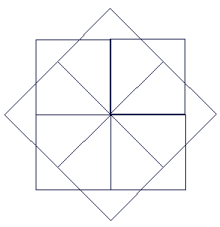
H.3. Mandalas simples

Le mandala qu'on créera maintenant est composé de plusieurs carrés qui sont dessinés en rotation autour du centre.

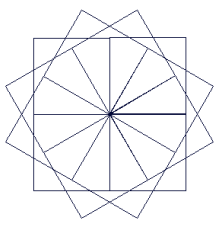
Exemple :



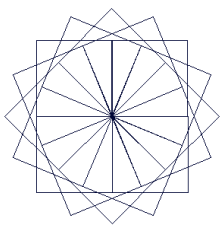
4 carrés



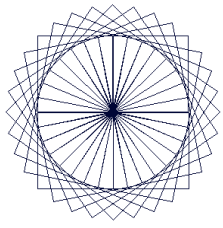
8 carrés



12 carrés

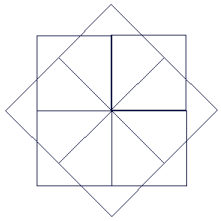


16 carrés



32 carrés

Exercice H-07 Mon premier mandala



Objectif : Un mandala composé de 8 carrés.

a) Crée un nouveau projet est sauvegarde-le sous le nom de ***Exercice H-07***.

b) Il faut d'abord calculer l'angle de rotation entre deux carrés consécutifs. Faites en sorte que le dessin final soit symétrique.

Calcule maintenant l'angle utilisé pour le Mandala :

Angle = …….° / ……. = …….°

c) Ajoute le script qui dessine le mandala composé de 8 carrés.

Conditions de départ :

- Le mandala est dessiné quand on appuie sur la touche '8'.
- Le stylo doit être positionné à la position (0 ; 0).
- Toutes traces d'un dessin précédent doivent être effacées.

d) Sauvegarde le projet.

Exercice H-08 Dessiner un cadre

a) Ouvre le projet de l' ***Exercice H-08****Erreur ! Source du renvoi introuvable.* et sauvegarde-le sous le nom de ***Exercice H-08*** ab

b) Le mandala est mis en valeur si tu l'entoures d'un cadre double.

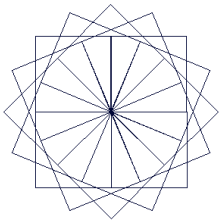
- Point du coin en haut à droite du cadre extérieur : 230 ; 170
- Taille du stylo 1
- Point du coin en haut à droite du cadre intérieur : 220 ; 160
- Taille du stylo 2

Condition de départ :

- Le cadre est dessiné quand on appuie sur la touche 'b'.

c) Sauvegarde le projet.

Exercice H-09 Mon deuxième mandala



a) Ouvre le projet de l' ***Exercice H-09****Erreur ! Source du renvoi introuvable.* et sauvegarde-le sous le nom de ***Exercice H-09***.

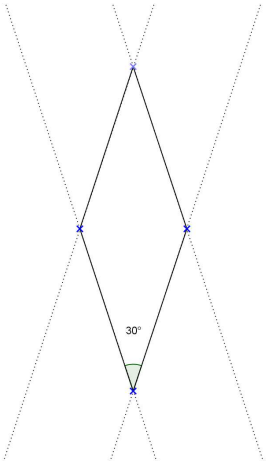
b) Ajoute le script qui dessine le mandala composé de 16 carrés (voir les opérateurs du chapitre pour le calcul de l'angle).

Condition de départ :

- Ce mandala est dessiné quand on appuie sur la touche 'm'.

c) Sauvegarde le projet.

Exercice H-10 Un mandala encore plus beau



Le mandala devient bien plus joli s'il est dessiné avec des losanges au lieu de carrés.

a) Ouvre le projet de l' ***Exercice H-09*** et sauvegarde-le sous le nom de ***Exercice H-10***.

b) Crée un mandala composé de 12 losanges identiques. Commence avec un seul losange et détermine les angles par un calcul et inscris-le sur le schéma ci-contre.

Condition de départ :

- Ce mandala est dessiné quand on appuie sur la touche 'r'.

c) Sauvegarde le projet.

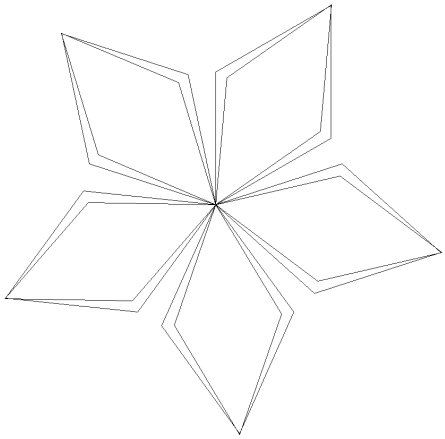
H.4. Quelques mandalas composés simples

On peut réaliser un mandala en combinant deux ou plusieurs motifs.

Exercice H-11 Mandalas composés

a) Ouvre le projet de l' ***Exercice H-10*** et sauvegarde-le sous le nom de ***Exercice H-11***.

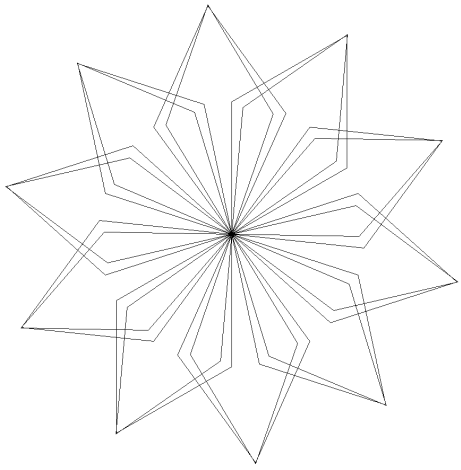
b) Crée le mandala ci-contre.



Condition de départ :

- Ce mandala est dessiné quand on appuie sur la touche '1'.
- Le grand losange a des côtés de longueur 82 et des angles de 130° respectivement de 50°.
- Le petit losange a des côtés de longueur 80 et des angles de 140° respectivement de 40°.
- Le mandala est composé de 5 sous-éléments.

c) Crée ensuite le mandala ci-contre.



Condition de départ :

- Ce mandala est dessiné quand on appuie sur la touche '2'.
- Le mandala est composé de 10 sous-éléments, les mêmes que pour le mandala précédent.

H.5. Dessiner des cercles

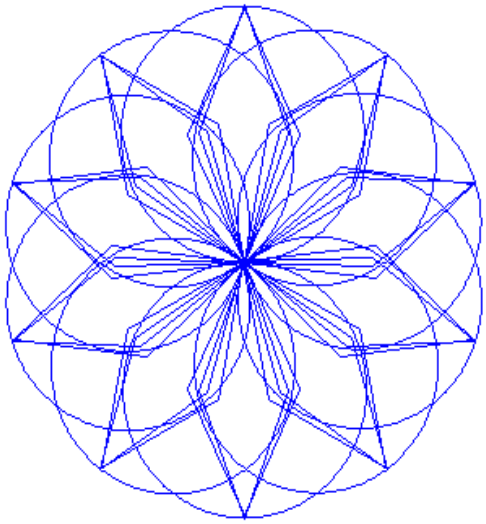
On peut simuler le dessin d'un cercle en dessiner un polygone régulier avec beaucoup de sommets (par exemple 360 sommets, en se tournant de 1° à chaque fois).

Exercice H-12 Mandalas composés 2

a) Ouvre le projet de l' ***Exercice H-11*** et sauvegarde-le sous le nom de ***Exercice H-12***.

b) Copie le script du mandala qui est dessiné quand on appuie sur '2'.

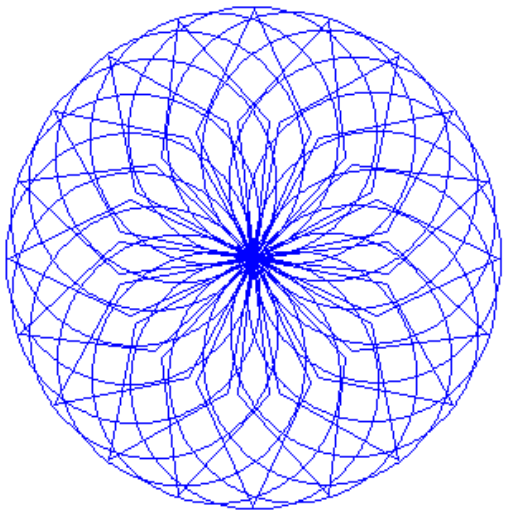
c) Modifie le script pour réaliser le mandala ci-contre.



Condition de départ :

- Ce mandala est dessiné quand on appuie sur la touche 'c'.
- Le cercle est un polygone à 60 côtés d'une longueur de 8.
- Le mandala est composé de 10 sous-éléments.

d) Crée ensuite le mandala ci-contre.



Condition de départ :

- Ce mandala est dessiné quand on appuie sur la touche 'd'.
- Le mandala est composé de 20 sous-éléments.

H.6. Dessiner des cercles (pour avancés)

La difficulté pour le dessin d'un cercle est de déterminer la longueur ℓ des segments pour arriver à un radius *r* donné.

Longueur totale des 360 segments (longueur ℓ) : _____ (1)

Circonférence du cercle (radius *r*) : _____ (2)

Si dans une équation mets (1) à égalité de (2), tu trouves une formule pour la longueur ℓ :

$$\ell = \text{———————}$$

Exercice H-13 Déterminer la longueur d'un côté (pour avancés)

a) Ouvre le projet et sauvegarde-le sous le nom de ***Exercice H-13***.

b) Calcule la longueur d'un segment en sachant que le cercle doit avoir un radius de 50.

c) Examine to dessin pour voir si ton radius est correct.

d) Sauvegarde ton projet.

I. Variables et expressions mathématiques

I.1. Variables

Nous sommes très souvent amenés à devoir nous souvenir de certaines choses : un numéro de maison, le code PIN d'un téléphone, un mot de passe, le nom d'un interlocuteur au téléphone, combien d'argent il nous reste dans la poche, ce qu'il faut encore faire l'après-midi, etc. En programmation, nous utilisons à cette fin les 'variables' au lieu d'un bout de papier ou d'un nœud dans un mouchoir.

Les variables nous permettent de mémoriser des valeurs, de les relire et de les changer. En Scratch, des variables de types différents sont utilisés : les nombres entiers, les nombres décimaux, ou des textes.

I.1.1 Créer et afficher des variables

La palette Données nous permet de créer de nouvelles variables, de les modifier et de les supprimer. Lors de la création d'une variable nous devons d'abord lui donner un nom significatif :

Nous définissons également quels objets ont le droit d'utiliser les variables, voire de les modifier.

Principe :

En général, une variable devrait toujours rester 'privée' (option "Pour ce lutin uniquement"). Aucun autre objet n'a le droit de modifier volontairement ou involontairement notre variable. (Nous ne laissons pas non plus trainer notre carnet d'adresses au milieu de la salle de classe.)

Dans quelques cas d'exception, nous mettons notre variable à disposition des autres objets.

Dès sa création, la variable est présentée elle-même sous forme d'un bloc :

(Le crochet indique que la variable est visible sur la scène)

L'affichage de la variable sur la scène peut se faire de plusieurs manières :

sortie normale

sortie large

potentiomètre

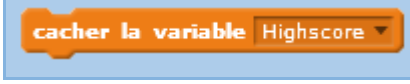
L'affichage (la 'sortie') de la variable peut être modifié par un double clic ou par un clic droit de la souris.

Cas spécial du potentiomètre :

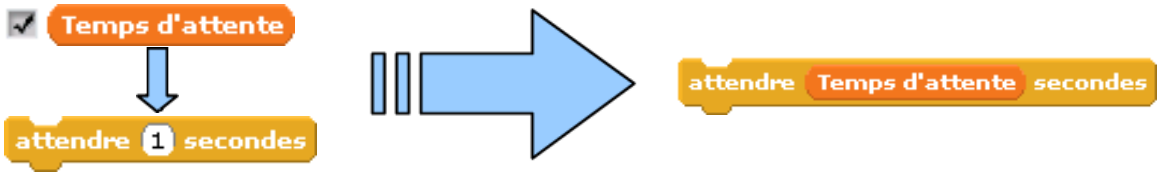
- Le potentiomètre ne fonctionne évidemment que pour les nombres entiers.
- Il est possible de définir également par un clic droit son intervalle de validité, c.-à-d. qu'on peut définir le minimum et le maximum du potentiomètre.

I.1.2 Travailler avec des variables

Après avoir créé la première variable, nous voyons apparaître quelques blocs qui nous permettent de modifier les variables dans nos programmes :

	Sélectionner une variable puis entrer une valeur (il est possible d'entrer des nombres négatifs, des décimales ou du texte)
	Sélectionner une variable puis entrer une valeur (il est possible d'entrer des nombres négatifs ou des décimales – mais **pas de texte**)
	Sélectionner une variable : Ces deux blocs nous permettent de rendre une variable visible sur la scène - ou de la cacher.
	

Dans un programme, tu peux utiliser les variables partout où jusqu'à présent tu as saisi directement des nombres ou des textes. Il suffit de les tirer avec la souris au bon endroit du bloc d'instruction.



Attention : il n'est pas possible de mélanger les textes et les nombres !

- Les variables numériques peuvent être utilisées partout.
- Les variables alphanumériques (de type texte) devraient être utilisées aux endroits prévus : (dit [ ]... , pense [ ] ... , demande [ ] ...)

Exercice I-01 Free the crab - reloaded!

Objectif :

Le crabe doit tenter de se libérer de la cage et compter combien de fois il rentre dans le mur de la cage.

Description :

a) Ouvre le projet de l' ***Exercice E-02*** („Free the crab“) et sauvegarde-le sous le nom de ***Exercice I-01***.

b) Ajoute une variable '*Collisions*', qui sera incrémentée (+1) à chaque fois que le crabe rentre dans le mur.

c) Relance le programme quand le crabe a trouvé la sortie et a été replacé au milieu de la cage. Qu'est-ce que tu observes ? Résous ce problème !

Exercice I-02 Free the crab - revisited!

Objectif :

Le crabe doit tenter de se libérer de la cage et compter combien de fois il rentre dans le mur de la cage. La vitesse du crabe peut être saisie.

Description :

a) Ouvre le projet de l' ***Exercice I-01*** („Free the crab - reloaded!“) et sauvegarde-le sous le nom de ***Exercice I-02***.

b) Ajoute une variable '*Temps d'attente*' représentée comme potentiomètre. Règle le potentiomètre à un intervalle entre 0 et 0.5. Incorpore maintenant la variable dans ton programme afin de régler avec elle la vitesse du crabe.

Exercice I-03 Fishchomp - II

Objectif : **Ajouter à un jeu existant un score et une limitation de temps.**

Description :

a) Ouvre le jeu ***FishChomp*** que ton enseignant a mis à ta disposition et sauvegarde-le sous le nom de ***Exercice I-03***.

Principe : le gros poisson suit la position de la souris et tente de manger les petits poissons.

Analyse les scripts (séquences d'instructions) des différents objets et essaie de comprendre comment le programme fonctionne.

b) Crée une nouvelle variable ‚*Score*' qui est incrémentée de 10 points pour chaque petit poisson que le gros poisson attrape. Le score doit bien-entendu être remis à zéro au lancement du programme.

c) Le jeu doit être limité à 30 secondes. Crée une variable '*Countdown*', qui est mis à 30 au début du programme, puis décrémentée (diminuée) de 1 à chaque seconde. Après l'écoulement de 30 secondes, le jeu s'arrête.

d) N'affiche la règle du jeu (le texte affiché en haut à droite) que pendant les 5 premières secondes du jeu.

I.1.3 Les variables prédéfinies

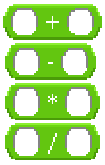
Il existe pour chaque objet quelques variables prédéfinies qu'on peut utiliser de la même manière que les variables créées par nous-mêmes :

- dans la palette **Mouvement** : position x,y, direction (angle)
- dans la palette **Apparence** : costume n°, taille
- dans la palette **Sons** : volume, tempo
- dans la palette **Capteurs** : réponse, chronomètre, volume sonore

I.2. Les expressions

Comme tout autre langage de programmation, Scratch permet de comparer des valeurs (les nombres et les textes) et de calculer avec elles. On n'a pas besoin de beaucoup de blocs d'opération pour obtenir finalement un puissant langage de programmation.

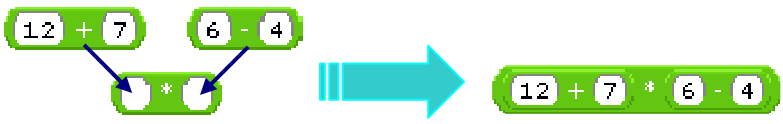
La plupart de ces blocs se situent dans la palette '**Opérateurs**' et traitent des nombres :

	addition, soustraction multiplication, division	pour les nombres entiers et décimaux
	donner un nombre aléatoire	réservé aux nombres entiers
	le reste de la division	réservé aux nombres entiers
	arrondir un nombre vers l'entier le plus proche	pour les nombres décimaux, le résultat est un nombre entier
	plusieurs fonctions mathématiques	La fonction la plus importante : *racine carrée, valeur absolue* Autres fonctions : *sin, cos, tan, asin, acos, atan, ln, log, e^, 10^*

Il est possible d'imbriquer les blocs les uns dans les autres afin de réaliser des calculs plus complexes. On peut comparer cela avec les parenthèses en mathématiques :

Exemple :

(12+7)*(6-4)
correspond à



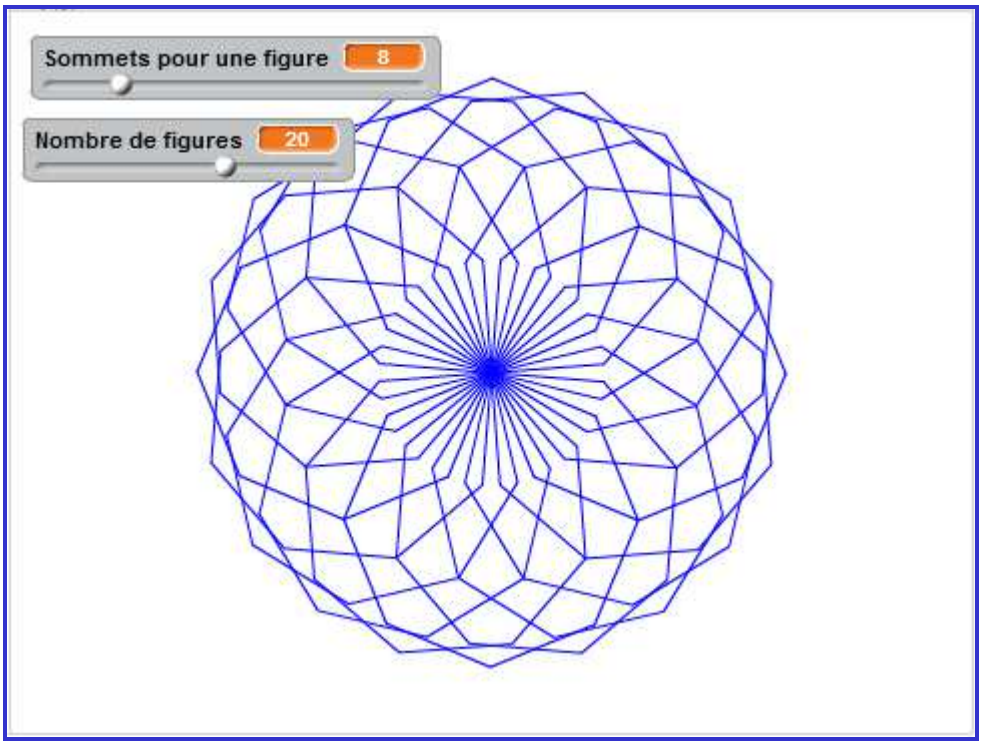
Il faut bien sûr veiller à imbriquer les blocs dans le bon ordre.

Principe : Le système procède en commençant avec le bloc le plus haut et en finissant avec le bloc le plus bas.

Exercice I-04 Générateur de mandalas

Objectif :

Calculer automatiquement des mandalas



Description :

a) Définis 2 variables **'Sommets pour une figure'** et **'Nombre de figures'**, affichés sous forme de potentiomètres. L'intervalle de validité pour les deux variables est [3…60] pour la 1ère et [0…60] pour la 2e.

b) Quand on appuie sur la touche 'espace', le programme dessine un mandala composé d'autant de figures que la valeur de la variable **'Nombre de figures'**, et chaque figure possède d'autant de coins que la valeur de la variable **'Sommets pour une figure'**.
La longueur d'un côté doit être calculée automatiquement de manière à ce que la figure ne dépasse pas la scène.

c) Affine les mandalas selon ton goût (couleur, taille et intensité du stylo, repeindre, ...)

d) Sauvegarde le projet sous le nom de **Exercice I-04**.

Exercice I-05 Montre analogue avec des « *variables de temps* » (pour avancés)

a) Ouvre le projet de l' ***Exercice D-02*** et sauvegarde-le sous le nom de ***Exercice I-05***.

b) Anime l'aiguille existante des secondes en utilisant la *variable de temps* suivante : 

c) Ajoute les aiguilles pour les minutes et les heures.

d) Montre aussi la date actuelle.

e) Sauvegarde le projet.

I.2.2 Vrai ou faux

Comme nous avons vu dans le chapitre E (Prendre des décisions), les blocs aux extrémités pointues servent à formuler des **conditions**.

Plus précisément, un bloc aux extrémités pointues donne ou bien le résultat '**vrai**' (condition vérifiée) ou bien le résultat '**faux**' (condition non vérifiée). De tels blocs peuvent être utilisés comme sous-blocs p.ex. dans les blocs "si <...> ...sinon...", "attendre jusqu'à <...>" ou "répéter jusqu'à <...>".

Les blocs opérateurs suivants nous permettent de comparer des valeurs (des nombres ou des textes) et d'utiliser leur résultat dans le contrôle de nos scripts.

Opérateurs de comparaison : 

Exemples :

Exercice I-06 Fishchomp - III

Objectif : **Retenir et afficher le score maximal (Highscore)**

Description :

a) Ouvre ton jeu *FishChomp* – II (***Exercice I-03***) et sauvegarde-le sous le nom de ***Exercice I-06*** ab.

b) Modifie le compte à rebours afin de pouvoir utiliser la boucle '*répéter jusqu'à*'. (Ceci donne une solution plus flexible puisqu'il ne faut changer qu'à un seul endroit la valeur de départ de la variable '**Countdown**' pour modifier la durée du jeu.).

c) Crée une variable '**Highscore**' qui vérifie à la fin du jeu si 'Score' dépasse le Highscore actuel. Dans ce cas, le score actuel devient le nouveau Highscore et un son (de ton choix) se fait entendre.

Exercice I-07 Devinette

Objectif :

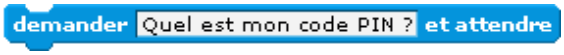
Deviner un nombre secret aléatoire calculé par l'ordinateur

Contexte :

Il faut absolument que tu puisses téléphoner, mais les piles de ton portable sont vides. Cassy a son portable sur elle mais elle ne veut pas que tu l'utilises. Elle te lance le défi suivant : *« Si tu arrives à deviner mon code PIN à 4 chiffres en moins de 15 essais alors tu peux utiliser mon portable autant que tu veux. A chaque essai, je ne te dis que si mon code PIN est plus grand ou plus petit que le nombre de ton essai. »*

Description :

Développe un programme qui simule Cassy et sa devinette :

- Au début du programme, il faut enregistrer dans une variable (invisible) ***Pin*** un code aléatoire entre 1000 et 9999. C'est le nombre que le joueur doit deviner.
- Tu peux également présenter le contexte de l'histoire. (P.ex. Cassy téléphone au début et dit *« Ah bon, tu veux utiliser mon téléphone. Eh bien, je te le laisse si tu arrives à deviner mon code à 4 chiffres en moins de 15 essais ! »*)
- Utilise le bloc 

pour attendre la saisie du nombre suivant. Tu trouves ce bloc dans la palette **Capteurs**. La valeur saisie par le joueur se trouve alors automatiquement dans la variable prédéfinie 

.
- Une deuxième Variable (visible) ***Essais*** mémorise le nombre d'essais du joueur.
- Si le joueur arrive à deviner le code PIN en moins de 15 essais, Cassy se fâche et donne le portable ☺
- Si, par contre, il n'y est pas arrivé après 14 essais, elle danse de joie pendant une seconde…
- Tu trouves des images pour Cassy dans la bibliothèque de Scratch. Pour la danse tu peux importer des sons de la bibliothèque ou utiliser tes propres sons. Pour rendre la danse plus rigolote encore, jette un coup d'œil sur les effets de la palette **Apparence** :

Amuse-toi bien pour la programmation et la devinette !

I.2.3 Pour avancés : conditions combinées

Nous avons souvent besoin de plusieurs conditions pour vérifier un état. Ces conditions doivent alors être combinées.

En mathématiques p.ex. il faut souvent vérifier si un nombre se trouve à l'intérieur d'un intervalle : 0 < X < 1000

Une condition combinée (ou complexe) peut être réalisée en utilisant les blocs Scratch suivants :

<**et**> veut dire que les deux conditions doivent être vérifiées

<**ou**> veut dire qu'au moins une des deux conditions doit être vérifiée

<**non**> veut dire que la condition ne doit pas être vérifiée

Pour vérifier si 0 < X < 1000 on peut écrire :

Pour vérifier si X≤0 ou X≥1000 on peut écrire en Scratch :

Mais on peut également écrire :

J. Procédures

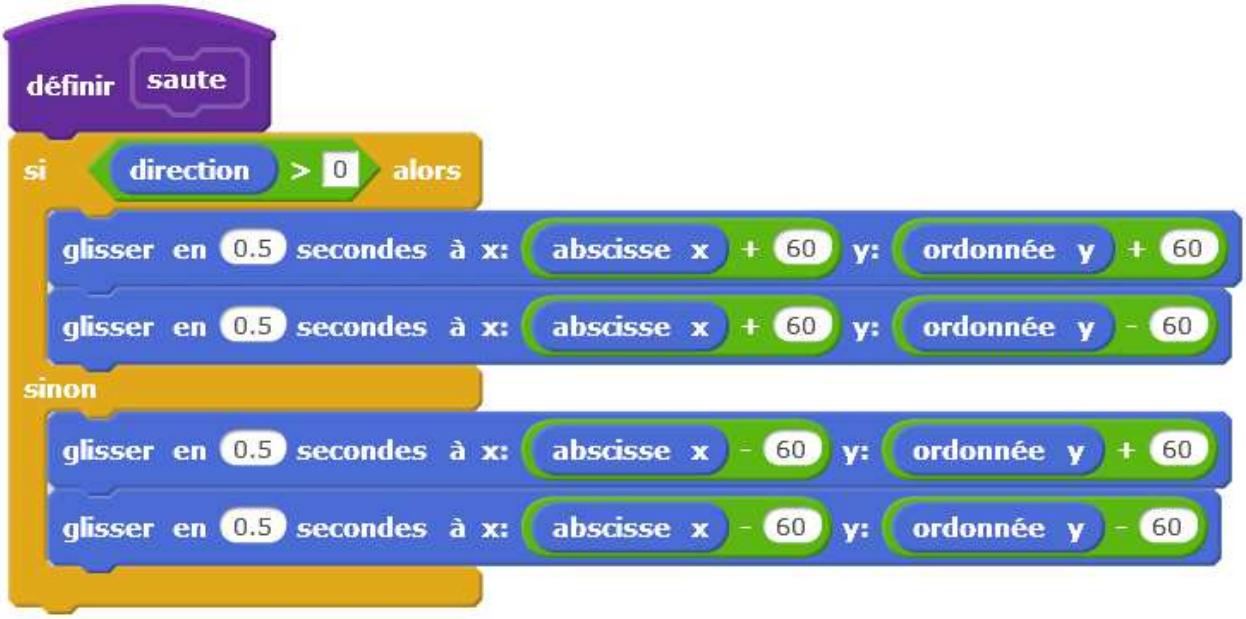
Il est possible de créer de nouveaux blocs d'instructions à partir des blocs existants. Ces nouveaux blocs sont appelés **procédures**. On peut les utiliser dans un script de la même manière qu'on utilise les instructions.

J.1. Définir une procédure

On crée une nouvelle procédure par un clic sur le bouton Créer un bloc de la palette Ajouter blocs. Il faut saisir ensuite le nom de la procédure dans la fenêtre de dialogue.

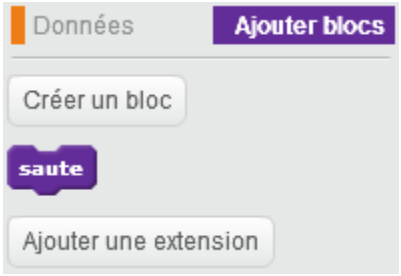
Après avoir cliqué sur OK, le chapeau "**définir**" apparaît dans l'espace de programmation. Nous pouvons maintenant assembler sous ce chapeau les instructions qui doivent être exécutées lors d'un appel de la procédure.

L'exemple suivant montre la définition de la procédure "**saute**". Elle permet au sprite de sauter 120 pixels en longueur et 60 pixels en hauteur.



J.2. Appeler une procédure

Après la définition de la procédure, celle-ci apparait sous forme de bloc dans la palette et peut être inséré dans un script, comme les autres instructions.

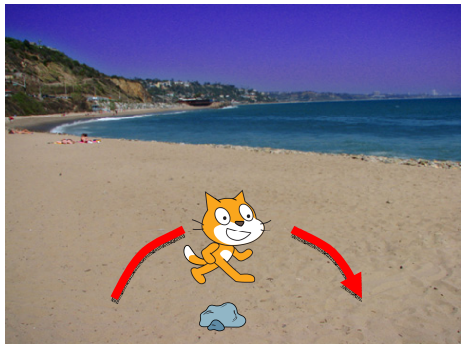


Notons :

Dans un sprite on peut utiliser les procédures définis dans celle-ci, mais pas les procédures définis dans d'autres sprites.

Exercice J-01 Le chat saute

Objectif : Le chat saute par-dessus des obstacles.



Description :

a) Définis la procédure **saute** comme indiqué plus haut.

b) Quand on appuie sur la touche espace, le chat saute. Utilise la procédure **saute** !

c) Quand on clique sur le drapeau vert, le chat se promène sur la plage en faisant des allées et venues. Quand il touche la pierre, il doit sauter par-dessus automatiquement. Utilise la procédure **saute** !

d) Sauvegarde le projet sous le nom de ***Exercice J-01***.

e) Imagine-toi avoir réalisé le programme sans la procédure **saute**. Quelles conclusions peux-tu en tirer ? Quels sont les avantages des procédures ?

J.3. Arguments et paramètres

Beaucoup d'instructions que tu connais déjà possèdent des **arguments**.

Exemples :

L'instruction **avancer** a un argument qui indique la grandeur du pas (10 dans l'exemple).

Le champ d'édition est arrondi puisqu'il attend une valeur numérique.

L'instruction **dire** a un argument qui indique le texte que le sprite doit dire ("Coucou !" dans l'exemple).

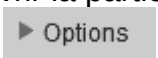
Le champ d'édition est rectangulaire puisqu'il attend une valeur de texte.

L'instruction **attendre** a un argument qui indique une condition qu'il faut attendre avant de continuer l'exécution.

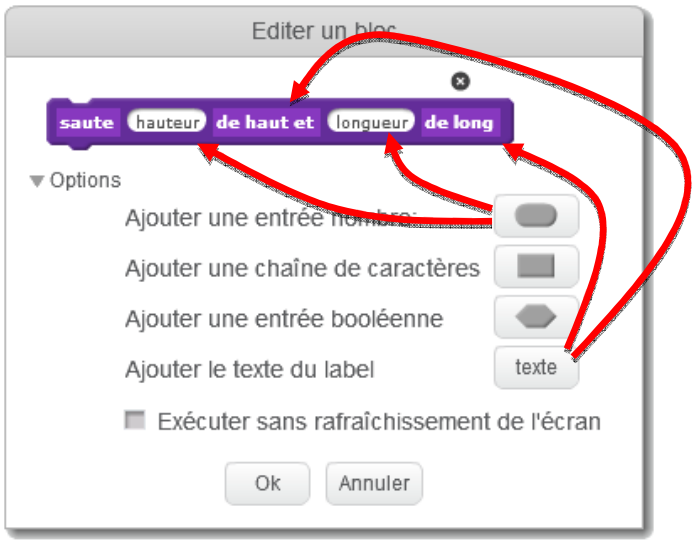
Le champ d'édition a des bouts pointus puisqu'il attend une condition (dont la valeur est ***vrai*** ou ***faux***).

Cette commande a deux arguments : Le texte à afficher (champ d'édition rectangulaire) et la durée d'affichage du texte (champ d'édition arrondi).

Une procédure peut également posséder des arguments. Lors de la définition de la procédure nous pouvons décrire les **paramètres**. C'est ainsi qu'on appelle les variables qui vont recevoir les arguments à l'aide de champs d'édition.

Pour y accéder, il faut ouvrir la partie inférieure en cliquant sur 

.



Dans cet exemple nous venons d'ajouter deux paramètres à notre procédure **saute**, un pour la hauteur et un pour la longueur du saut. Nous pouvons donc maintenant indiquer la hauteur et la longueur du saut lors de l'appel de la procédure.

Comme nos paramètres attendent des valeurs numériques, nous avons cliqué sur les boutons arrondis. Nous avons ensuite nommé les paramètres et ajouté des textes descriptifs pour rendre le bloc plus compréhensible.

Après un clic sur OK, le chapeau **définir** apparaît, muni des deux nouveaux paramètres avec leurs indications textuelles.

Il est maintenant facile de glisser ces paramètres aux endroits souhaités.

Notons :

Pour modifier la définition d'une procédure, il faut cliquer avec la touche droite de la souris sur son bloc et choisir ensuite **éditer**.

Exercice J-02 Le chat saute plus haut

Objectif : Le chat saute par-dessus d'obstacles plus hauts encore.

Description :

a) Ouvre le projet ***Exercice J-01*** et sauvegarde-le sous le nom de ***Exercice J-02*** ab.

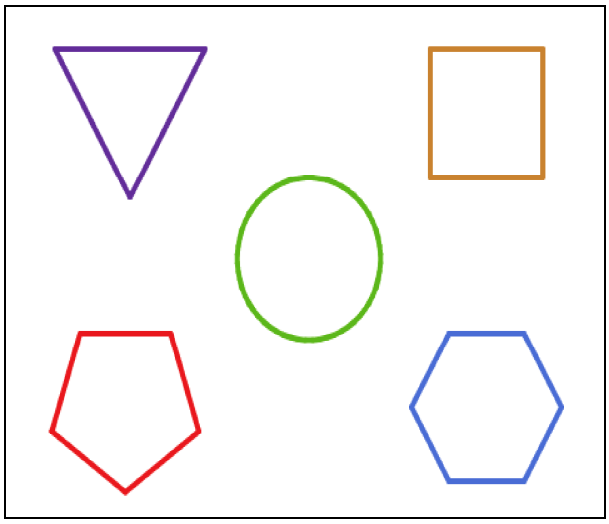
b) Définis la procédure **saute** comme indiqué plus haut avec les paramètres **hauteur** et **longueur** et complète la programmation des scripts.

c) Place un palmier (anglais : *palmtree*) sur la plage et modifie le script pour que le chat saute par un petit saut au-dessus de la pierre mais avec un grand saut au-dessus du palmier !

d) Sauvegarde le projet.

Exercice J-03 Polygone

Objectif : Dessiner des polygones réguliers.



Description :

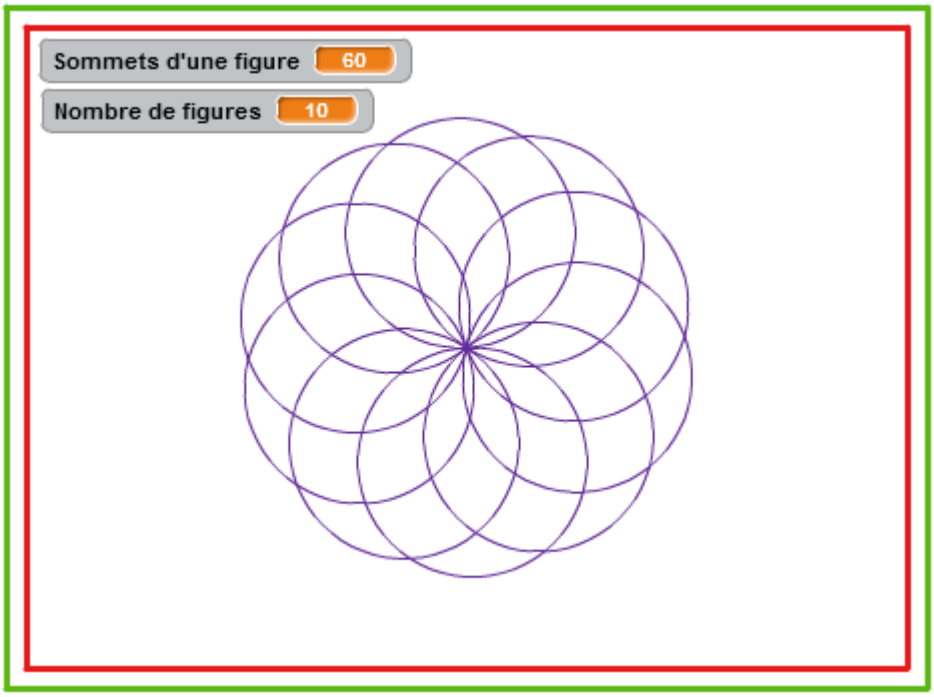
a) Crée une procédure **polygone** qui dessine un polygone régulier. Le nombre de sommets ainsi que la longueur d'un côté sont donnés sous forme d'arguments.

b) Crée un programme qui dessine des polygones de différentes formes et couleurs sur la scène.

c) Enregistre le projet sous le nom de ***Exercice J-03***.

Exercice J-04 Générateur de mandalas

Objectif : Générateur de mandalas avec procédures.



Description :

a) Ouvre le projet ***Exercice J-03*** et sauvegarde-le sous le nom de ***Exercice J-04***.

b) Crée une procédure **mandala** qui dessine un mandala composé d'un nombre donné de polygones. Utilise une 2e procédure **polygone**. Essaye de trouver toi-même les paramètres nécessaires pour les procédures.

c) Utilise la procédure **mandala** pour créer un générateur de mandalas (voir ***Exercice I-04*** du chapitre I).

d) Crée une procédure **rectangle** qui dessine un rectangle. La longueur et la largeur du rectangle sont les arguments de la procédure.

e) Utilise la procédure **rectangle** pour dessiner un cadre double autour du mandala (lancé par la touche 'r').

f) Modifie la définition de la procédure en cochant la case à cocher "Exécuter sans rafraichissement de l'écran" et relance le générateur de mandalas. Qu'est-ce que tu remarques ?

g) Sauvegarde le projet.