



Parcours

SCRATCH, découverte et initiation

CM1-CM2

Cette séquence propose à la fois des activités débranchées (informatique sans ordinateur) permettant d'aborder des concepts de base de la science informatique (algorithme, langage, programme), et des activités branchées (nécessitant un ordinateur ou une tablette) pour s'initier à la programmation avec Scratch.

Le codage et la programmation dans les programmes de cycle 3

Sciences et technologie

Matériaux et objets techniques

- Le stockage des données, notions d'algorithmes, les objets programmables.
Les élèves découvrent l'algorithme en utilisant des logiciels d'applications visuelles et ludiques. Ils exploitent les moyens informatiques en pratiquant le travail collaboratif.

Mathématiques

Espace et géométrie

- (Se) repérer et (se) déplacer dans l'espace en utilisant ou en élaborant des représentations
- Se repérer, décrire ou exécuter des déplacements, sur un plan ou sur une carte.
- Accomplir, décrire, coder des déplacements dans des espaces familiers.
- Programmer les déplacements d'un robot ou ceux d'un personnage sur un écran.
- Vocabulaire permettant de définir des positions et des déplacements.
- Divers modes de représentation de l'espace.
- Situations donnant lieu à des repérages dans l'espace ou à la description, au codage ou au décodage de déplacements.

- Travailler : avec de nouvelles ressources comme les [] logiciels d'initiation à la programmation

Repères de progressivité

Une initiation à la programmation est faite à l'occasion notamment de activités de repérage ou de déplacement (programmer les déplacements d'un robot ou ceux d'un personnage sur un écran)

Pour aller plus loin : document d'accompagnement proposé par Eduscol

http://cache.media.eduscol.education.fr/file/Initiation_a_la_programmation/92/6/RA16_C2_C3_MATH_initiation_programmation_doc_maitre_624926.pdf

Références aux programmes et au socle commun pour le cycle 3

Socle commun de connaissances, compétences et culture

Domaines	Compétences travaillées
1. Les langages pour penser et communiquer	Pratiquer les langages scientifiques pour permettre de résoudre des problèmes
2. Les méthodes et outils pour apprendre	Organiser son travail et sa pensée
3. La formation de la personne et du citoyen	Développer la confiance en soi et le respect des autres
4. Les systèmes naturels et les systèmes techniques	Pratiquer des démarches scientifiques
5. Les représentations du monde et l'activité humaine	Mettre en place les notions d'espace, s'orienter, se déplacer

Espace et géométrie

Attendus de fin de cycle
(Se) repérer et (se) déplacer en utilisant des repères et des représentations.
Compétences associées
- Accomplir, décrire, coder des déplacements dans des espaces familiers. - Programmer les déplacements d'un robot ou ceux d'un personnage sur un écran.

Matériaux et objets techniques

Attendus de fin de cycle
Repérer et comprendre la communication et la gestion de l'information.
Compétences associées
- Le stockage des données, notions d'algorithme, les objets programmables

Ce parcours est une initiation à la pensée algorithmique et à la programmation. Les activités proposées aux élèves vont les conduire à :

- savoir décomposer un problème en tâches simples
- savoir reconnaître des tâches qu'on a déjà effectuées, ou qui se répètent
- apprendre à travailler ensemble
- favoriser leur imagination, leur créativité, sous une modalité attrayante

Partenaires :

ARTEM Game Lab ?

SOMMAIRE

Séances
Séance 1 en classe : Découvrir la notion d'algorithme.
Séance 2 en classe : Se repérer et se déplacer sur une grille.
Séance 3 en classe : Découvrir le langage de programmation de Scratch.
Séance 4 au Centre Pilote : Initiation à la programmation avec Scratch.
Séance 5 au Centre Pilote : Programmer un jeu de labyrinthe partie 1.
Séance 6 au Centre Pilote : Programmer un jeu de labyrinthe partie 2.
Séance 7 au Centre Pilote : Programmer un jeu de labyrinthe partie 3.
Annexes
Ressources

Séance 1 en classe

Découvrir la notion d'algorithme

Objectifs

Se familiariser avec la notion d'algorithme.

Notions

« Algorithme »

- Un "algorithme" est une méthode permettant de résoudre un problème
- Un test dit quelle action effectuer quand une condition est vérifiée
- Une condition est une expression qui est soit vraie, soit fausse
- Une boucle permet de répéter un certain nombre de fois (voire indéfiniment) toute une série d'instructions

Matériel

Par binôme d'élèves

Un jeu de 16 bâtonnets (allumettes, feutres...)

Phases de déroulement de l'activité

Situation déclenchante

Proposer une partie de jeu de Nim.

Présentation du jeu et de ses règles :

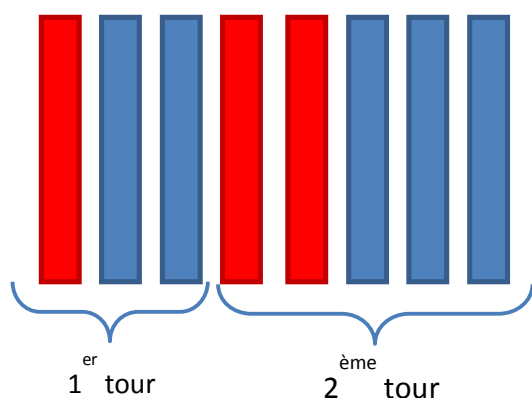
C'est un jeu simple permettant de comprendre la notion d'algorithme.

Prévoir une collection de 16 objets identiques (bâtonnets, allumettes, feutres...)

Avec 8 bâtonnets ou objets pour commencer

On dispose sur une table 8 objets identiques. Chacun leur tour, les deux joueurs ramassent 1, 2 ou 3 objets sur la table. Le joueur qui ramasse le dernier objet remporte la partie.

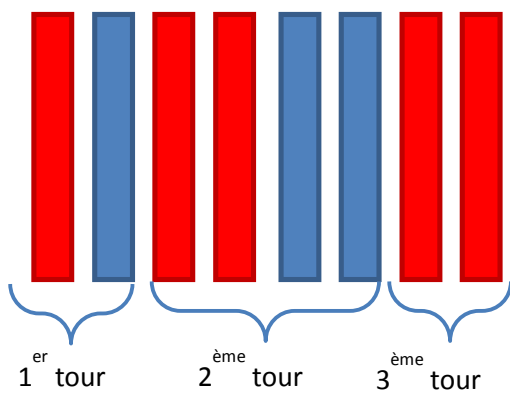
Quelques exemples de partie :



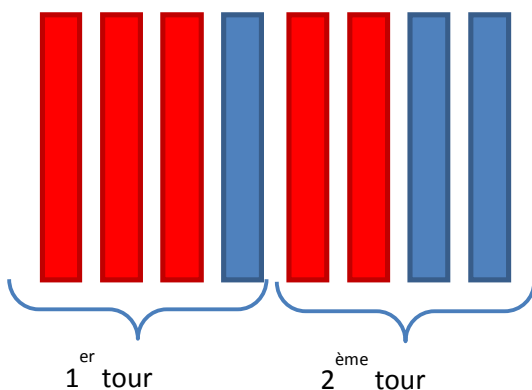
Premier tour : le premier joueur (J1) ramasse 1 objet (en rouge), le second joueur (J2) en ramasse 2 (en bleu).

Deuxième tour : J1 ramasse 2 objets, J2 en ramasse 3 dont le dernier.

J2 gagne.



Premier tour : J1 ramasse 1 objet, J2 en ramasse 1.
Deuxième tour : J1 ramasse 2 objets, J2 en ramasse 2.
Troisième tour : J1 ramasse les deux derniers objets.
J1 gagne.



Premier tour : J1 ramasse 3 objets, J2 en ramasse 1.
Deuxième tour : J1 ramasse 2 objets, J2 en ramasse 2 dont le dernier.
J2 gagne.

Phase d'appropriation

Les élèves jouent par binôme. Qui gagne le plus souvent : le joueur 1 qui commence la partie ou bien le joueur 2 ?

Phase de recherche

L'enseignant joue plusieurs parties (en tant que joueur 2) et bien sûr, gagne à chaque fois !
Quelle est la stratégie qui permet de gagner à coup sûr ?

Mise en commun

Les élèves exposent leur solution.

Formalisation de la stratégie gagnante

Le jeu de Nim est sans suspense : le premier à jouer perd, car il existe une astuce pour que le deuxième joueur gagne à tous les coups. La stratégie gagnante est de laisser 4 objets à l'adversaire.

Pour se convaincre de l'efficacité de la stratégie gagnante, prenons le deuxième et dernier tour du troisième exemple de partie ci-dessus.

Il reste 4 objets, et J1 (joueur 1, rouge) joue :

~ si J1 prend 1 objet, alors J2 (joueur 2, bleu) en prend 3 (dont le dernier) ;

~ si J1 prend 2 objets, alors J2 en prend 2 (dont le dernier) ;

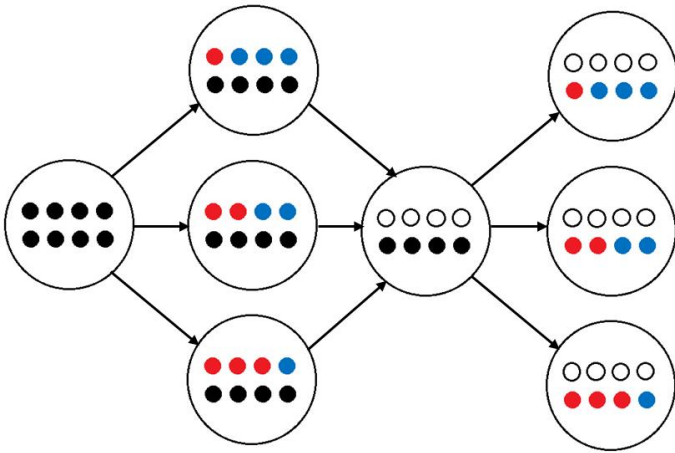
~ si J1 prend 3 objets, alors J2 en prend 1 (le dernier).

Dans ce cas, si J2 sait jouer, J1 perd à tous les coups.

En effet, J2 peut guider le jeu de manière à passer de 8 objets à 4, puis à 0.

Donc, si J2 sait jouer, J1 a perdu la partie avant même de commencer.

Pour amener les participants à découvrir la stratégie gagnante, on peut grouper les objets par 4, rendant ainsi l'astuce plus visible.

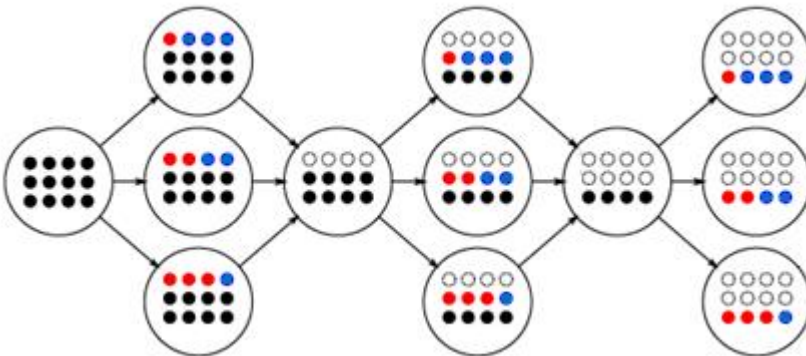


Les élèves rejouent en essayant d'appliquer la stratégie gagnante et d'atteindre le plus possible la victoire !

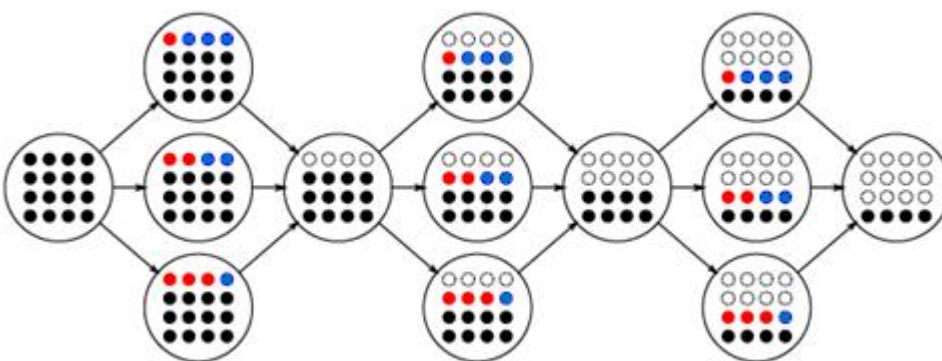
Remarque : dans les deux premiers exemples de partie, la stratégie gagnante n'est pas appliquée.

Proposer de faire une partie avec 12 objets.

Les enfants réussissent-ils à appliquer la stratégie gagnante ?



Et une partie avec 16 objets ?



Et une partie avec 15 objets ?

Il faut alors modifier les règles :

- chaque joueur peut prendre 1, 2, 3 ou 4 objets, et la stratégie gagnante consiste pour le joueur 2 à laisser des multiples de 5.

(possible également avec des multiples de 3).

Les notions de test et de condition sont abordées dans cette activité :

SI le joueur 1 prend 1 objet, **ALORS** le joueur 2 prend 3 objets.

Les termes test et condition sont définis ainsi :

Un test dit quelle action effectuer quand une condition est vérifiée.

Une condition est une expression qui est soit vraie, soit fausse.

La condition est ici : *le joueur 1 prend 1 objet.*

Elle est soit vraie et le test dit alors : *le joueur 2 prend 3 objets.*

Si la condition est fausse, une autre condition est posée :

SI le joueur 1 prend 2 objets, **ALORS** le joueur 2 prend 2 objets.

etc.

Autre notion abordée, celle de boucle : chaque fois que 4 objets sont enlevés, la procédure recommence.

La procédure est répétée 2 fois pour 8 objets, 3 fois pour 12 objets, etc.

Conclusion

Comme pour le jeu de Nim, un algorithme est une stratégie gagnante permettant de trouver la solution à un problème donné. Ici le problème était « comment gagner au jeu de Nim ? »

Imaginons que l'on veut apprendre à un ordinateur comment jouer et gagner au jeu de Nim : il faut indiquer à la machine quelle est la stratégie gagnante. L'algorithme correspondant à cette stratégie doit pour cela être traduit dans un langage compréhensible par l'ordinateur. On parle de langage de programmation. La conception d'un programme commence donc par la conception d'un algorithme.

Dans les séances à venir, les élèves vont découvrir le langage du logiciel Scratch et seront amenés à construire des programmes traduisant des algorithmes avec ce langage.

Trace écrite

La classe synthétise collectivement ce qui a été appris au cours de cette séance :

Un algorithme est une méthode permettant de résoudre un problème.

Un test dit quelle action effectuer quand une condition est vérifiée.

Une condition est une expression qui est soit vraie, soit fausse.

Une boucle permet de répéter plusieurs fois une série d'instructions.

Remarques

Prolongements

Demander aux enfants de réfléchir et décrire un algorithme simple (sous la forme d'une suite de phrases simples) pour décrire les actions qui répondent à un problème donné. Voir fiche Annexe 1.1

Séance 2 en classe

Se repérer et se déplacer sur une grille.

Objectifs

Se repérer, décrire ou exécuter des déplacements, sur un plan ou sur une carte.

Notions

Vocabulaire permettant de définir des positions et des déplacements.

Matériel

Pour la classe : vidéo Annexe 2.1

Par élève : fiches Annexe 2.2

Phases de déroulement de l'activité

Les activités qui vont suivre (en classe et au centre pilote) proposent une initiation à la programmation avec Scratch.

Scratch est un logiciel qui a été conçu pour apprendre à programmer.

Le langage de programmation utilisé permet de créer facilement des animations, des histoires interactives, des jeux vidéo, des créations musicales et artistiques, etc...

Phase 1 : Visionner une courte animation où on voit un personnage se déplacer (vidéo Annexe 2.1_fond blanc). Cette animation a été réalisée avec **Scratch**.



Interroger les élèves : que voit-on sur la vidéo ?

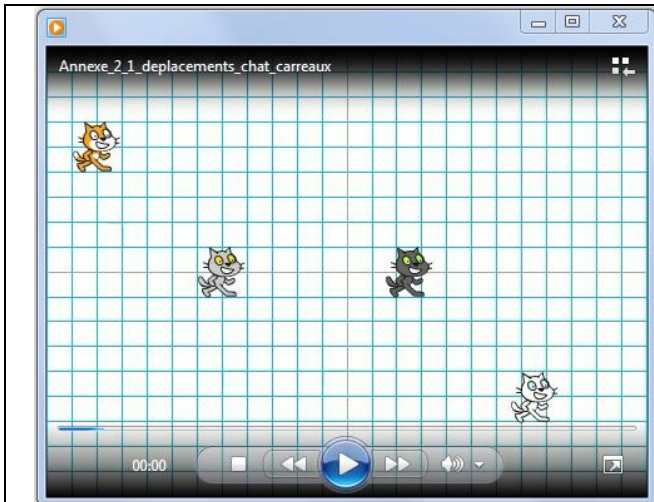
Comment décririez-vous le déplacement du chat orange à l'écran ?

De façon précise ?

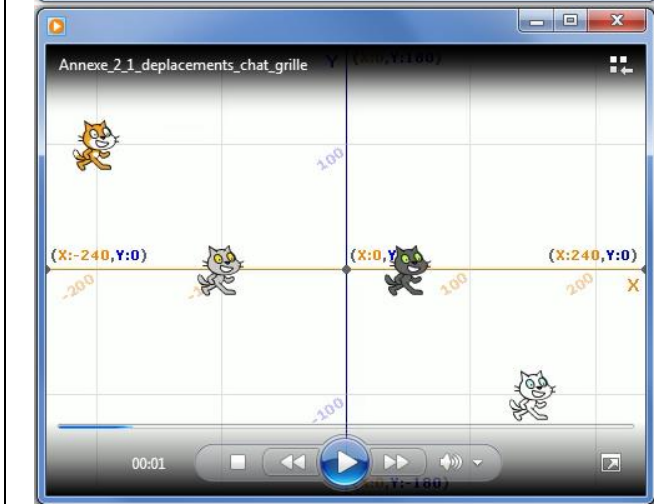
Cet échange amène à l'idée que pour réaliser une telle animation avec Scratch, il faut savoir repérer la position de personnages à l'écran et décrire des déplacements pour pouvoir ensuite les programmer.

Comment faire ce repérage et décrire les déplacements ?

Si les élèves n'ont pas d'idée, montrer la même animation avec un fond quadrillé (Annexe 2.1_carreaux).



Avec le quadrillage, on peut dire de combien de carreaux se déplace le chat orange vers la droite, le bas, etc.



Puis montrer la même animation avec comme fond la grille Scratch (Annexe 2.1_grille). La grille permet de repérer avec précision les positions des personnages à l'écran et de décrire leurs déplacements.

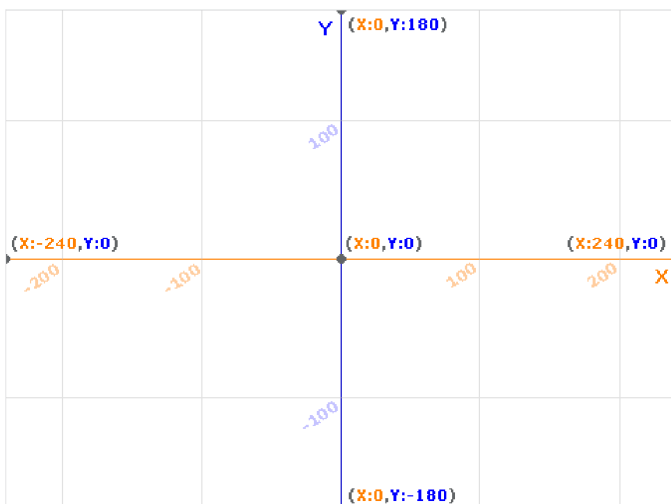
Phase 2.

Se repérer sur la grille.

Projeter à la classe la grille de la 1^{ère} étape de la fiche Annexe 2.2.

Cette grille est celle du logiciel Scratch, elle y décrit l'espace où évoluent les personnages et les objets dans les programmes.

Décrire l'image, que voit-on ?



- des droites
- des nombres
- des lettres (x et y)
- des carreaux
- des couleurs

La grille a pour dimensions 480 en largeur et 360 en hauteur. L'unité est le pixel, comme sur une image numérique (pixel est la contraction de l'anglais « picture element » qui se traduit par « élément d'image »).

Comment se repère-t-on sur la grille ?

La position d'un point sur la grille est donnée par un couple de nombres :

- le premier nombre correspond à la position de gauche à droite de la grille (on l'appelle x) ;
- le deuxième nombre correspond à la position de bas en haut de la grille (on l'appelle y).

On dit que ces nombres pour x et y sont les coordonnées du point.

Le point noir au milieu de la grille, à l'intersection de la droite orange et de la droite bleue, est repéré par la position $(x:0, y:0)$. On peut simplifier cette écriture en $(0, 0)$ en n'écrivant pas les lettres x et y .

Il y a des nombres négatifs sur la grille : -240, -200, -180 et -100. L'utilisation de coordonnées négatives ne devrait pas poser de problème, les enfants connaissent d'autres échelles graduées/situations avec des nombres négatifs :

- Les températures hivernales peuvent descendre sous les 0°C et être négatives ;
- Les commandes d'un ascenseur peuvent avoir des boutons indiquant -1, ou -2 pour des niveaux souterrains (sous-sol, cave, parking).

Demander aux élèves où se situent sur la grille les points pour lesquels y vaut 0.

→ La droite centrale horizontale de couleur orange rassemble tous les points pour lesquels y vaut 0 ; on l'appelle l'axe des x .

En dessous de la droite orange, les valeurs de y sont négatives, au-dessus elles sont positives.

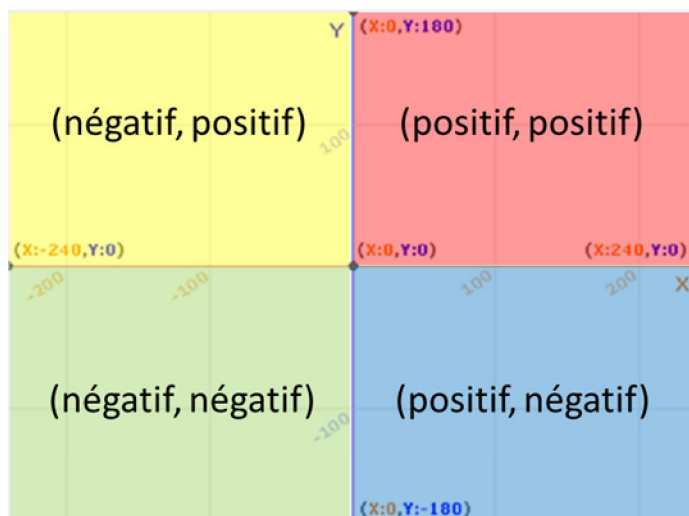
Demander aux élèves où se situent sur la grille les points pour lesquels x vaut 0.

→ La droite centrale verticale de couleur bleue rassemble tous les points pour lesquels x vaut 0 ; on l'appelle l'axe des y .

À gauche de la droite bleue, les valeurs de x sont négatives, à droite elles sont positives.

La grille peut se décomposer en quartiers où les coordonnées sont données par un couple de nombres :

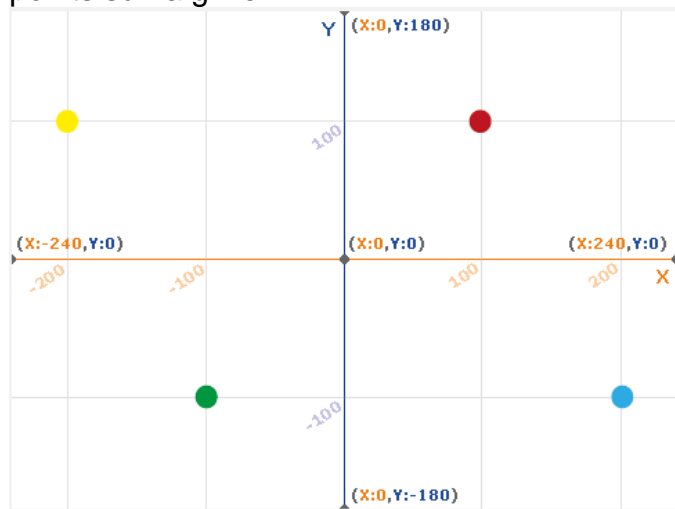
- (positif, positif) en haut à droite (zone rouge)
- (positif, négatif) en bas à droite (zone bleue)
- (négatif, positif) en haut à gauche (zone jaune)
- (négatif, négatif) en bas à gauche (zone verte)



Un quadrillage gris clair large (tous les 100 pixels) aide au repérage sur la grille.

Trouver et noter la position de points.

Les élèves doivent relever les valeurs de x et y (les coordonnées) pour chacun des quatre points sur la grille.



Faire le premier point ensemble.

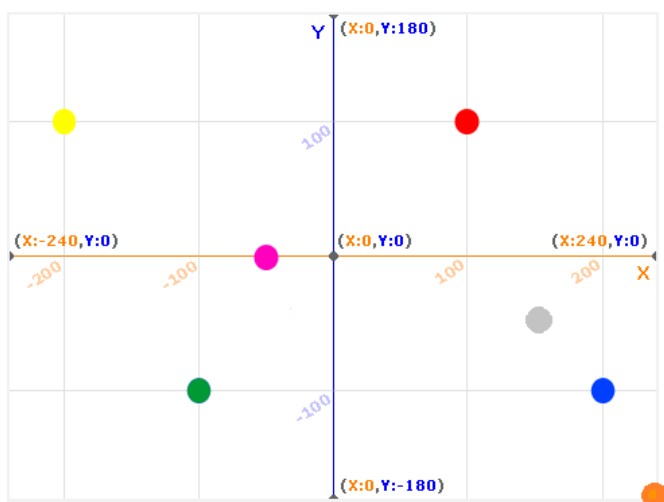
Commencer par les valeurs de x et y, puis donner la position sous la forme (x, y).

Point noir	x : 0	y : 0	(0, 0)
Point rouge	x : 100	y : 100	(100, 100)
Point jaune	x : -200	y : 100	(-200, 100)
Point vert	x : -100	y : -100	(-100, -100)
Point bleu	x : 200	y : -100	(200, -100)

S'assurer que les élèves ont compris comment se repérer sur la grille.

Placer des points connaissant leurs coordonnées.

- rose à (- 50, 0)
- gris à (150, - 50)
- orange à (240, - 180)
- violet à (- 300, - 100)



Le point rose et le point gris ne sont pas sur une intersection du quadrillage gris clair ; les élèves peuvent envisager de tracer un quadrillage plus serré, tous les 50 pixels, ou bien utiliser un outil de mesure (règle) pour placer ces points (proportion, fraction).

Le point orange est dans le coin inférieur droit, à la limite de la grille.

Le point violet ne peut pas être placé sur la grille puisque sa valeur de x est au-delà de -240 qui est la limite de la grille.

Déplacer un point.

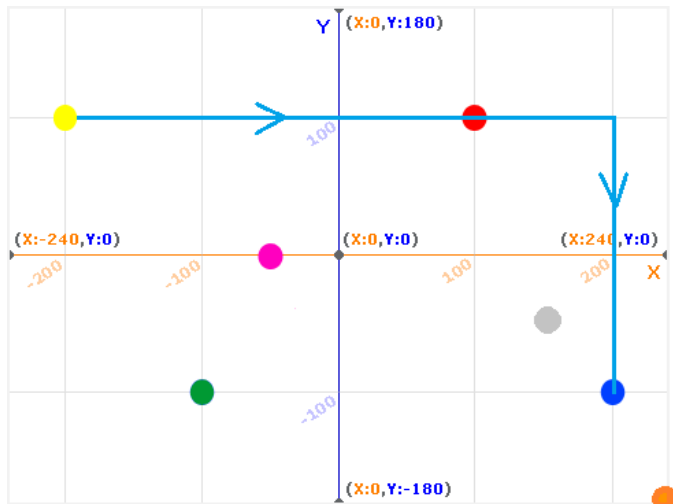
Je veux déplacer le point jaune de 400 en x et de -200 en y.

Où ce point va-t-il se retrouver ?

Se déplacer de 400 en x signifie se déplacer vers les valeurs positives de x (vers la droite) de 400 pixels.

Se déplacer de -200 en y signifie se déplacer vers les valeurs négatives de y (vers le bas) de 200 pixels.

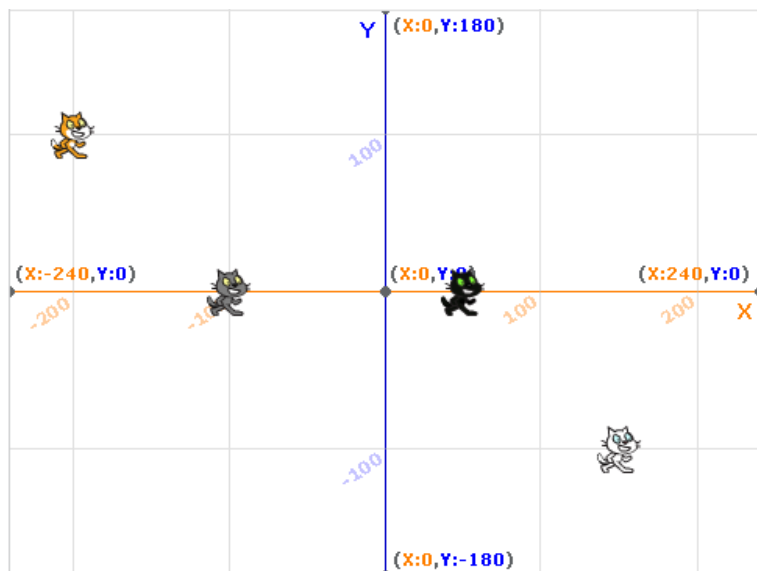
Remarque : ces instructions de déplacement de 400 ou -200 en x ou en y seront utilisées plus tard dans Scratch.



Le point jaune se retrouve à la position du point bleu.

Proposer aux élèves ou demander aux élèves de proposer d'autres déplacements de ce type, par exemple comment doit-on déplacer le point rose pour qu'il se retrouve à la position du point rouge ?

Aux élèves de jouer ! plusieurs solutions possibles.



	Coordonnées (x,y)
Chat orange	(-200,100)
Chat gris	(-100,0)
Chat noir	(50,0)
Chat blanc	(150,-100)

Remarque : la position d'un personnage est donnée pour son centre.

Pour le chat, le centre est situé au niveau de la bouche.

Exemple de déplacement du chat orange pour qu'il rejoigne le chat blanc, en passant par le chat gris et le chat noir :

- Se déplacer de 100 en x (vers la droite)
- Se déplacer de -100 en y (vers le bas)
- Se déplacer de 150 en x (vers la droite)
- Se déplacer de -100 en y (vers le bas)
- Se déplacer de 100 en x (vers la droite)

On peut commencer avec les élèves par décrire les déplacements en donnant les directions (tant de pixels vers le haut, ou la gauche) puis passer ensuite aux instructions déplacer de tant de pixels en y ou de . tant de pixels en x.

Les questions peuvent être simplifiées si besoin.

Conclusion

Sur une grille, on repère la position d'un objet ou d'un personnage par deux nombres (ses coordonnées). Un déplacement consiste à passer d'une position sur la grille à une autre position repérée par des coordonnées différentes.

Pour réaliser des animations avec Scratch, il faut savoir repérer la position de personnages à l'écran et décrire des déplacements pour pouvoir ensuite les programmer.

Retour **SOMMAIRE**

Séance 3 en classe

Découvrir le langage de programmation de Scratch

Objectifs

Découvrir le langage de programmation de Scratch.
Découvrir l'interface de Scratch.

Notions

« Langages »

- Un langage de programmation permet de donner des instructions à une machine ; ce langage est compréhensible par l'homme et par la machine.
- Un programme est un algorithme exprimé dans un langage de programmation.

Matériel

Par groupe de 3 ou 4 élèves :

Annexe 3.1 : blocs d'instructions Scratch en noir et blanc

Annexe 3.3 : Blocs d'instructions Scratch par catégories (noir et blanc)

Pour l'enseignant : à projeter

Annexe 2.1 (vidéo Annexe 2.1_fond blanc)

Annexe 3.2 : Blocs d'instructions Scratch en couleurs

Annexe 3.3 : Blocs d'instructions Scratch par catégories

Annexe 3.4 : l'interface de Scratch (diaporama ou pdf)

Phases de déroulement de l'activité

Phase 1 :

Visionner à nouveau la vidéo de l'activité précédente (Annexe 2.1_fond blanc)

A présent que l'on sait repérer les personnages et décrire leur déplacement, comment faire pour programmer ?

Laisser réfléchir les élèves et faire des propositions.

Il faut dire à l'ordinateur comment faire : d'où part le chat, où il arrive, quel chemin il emprunte.

Il faut écrire un algorithme qui résout ce problème et le transmettre à l'ordinateur.

Sous quelle forme doit-on transmettre les instructions ? Quel langage est compris par Scratch ? Est-ce le même langage que celui que nous utilisons pour parler et écrire ?

Scratch est utilisé partout dans le monde. Il possède son propre langage.

Parce que c'est un logiciel de programmation visuelle (ou graphique), sa mise en œuvre est simple ; en effet il n'est pas nécessaire de connaître ni d'apprendre un vocabulaire et une grammaire spécifiques car les instructions du langage de programmation sont représentées par des blocs. L'utilisateur manipule et empile des blocs pour former des algorithmes. Les instructions seront exécutées successivement, dans une logique séquentielle.

Expérimentation :

Distribuer à chaque groupe d'élèves une fiche annexe 3.2 avec un jeu de blocs d'instructions (impression en noir et blanc).

Expliquer que les instructions dans Scratch se présentent sous la forme de blocs et qu'ils ont sous les yeux une partie de ces blocs (il en existe plus d'une centaine).

Demander aux élèves de bien les regarder et de les classer en catégories, selon des critères

qu'ils auront établis.

Phase 2 : mise en commun

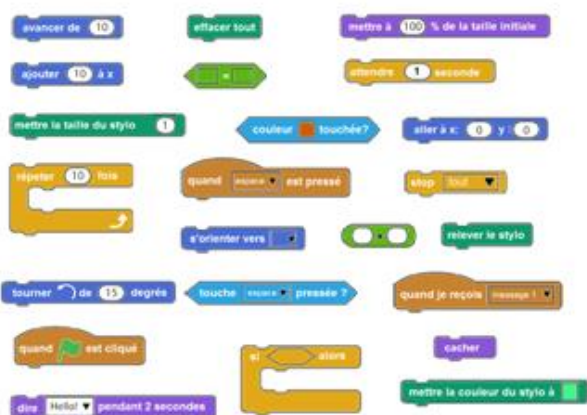
Les élèves présentent leur classement en donnant les critères qu'ils ont retenus.

- Forme des blocs
- Taille des blocs
- Mots communs dans les instructions
- Opérateurs mathématiques présents dans les instructions
- Autres

Les propositions des uns et des autres seront discutées.

Cet exercice est difficile et les élèves ne pourront sans doute ne le réussir qu'en partie.

Pour les aider à aller plus loin, l'enseignant projette au tableau la même série de blocs mais en couleurs (Annexe 3.2), tels qu'ils les trouveront dans Scratch.

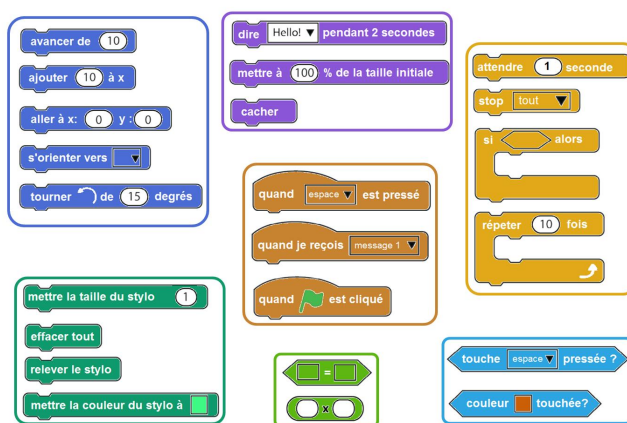


Que peut-on en dire à présent ?

Il y a un code de couleurs qui indique comment classer les blocs en catégories.

7 couleurs différentes dans la sélection de blocs proposée → 7 catégories

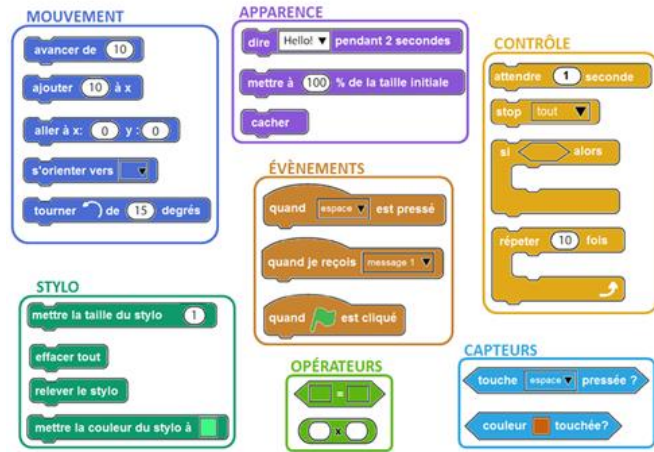
L'enseignant projette la fiche de l'annexe 3.3, montrant les blocs par catégories et distribue aux élèves une version noir et blanc de cette fiche.



Inviter les groupes d'élèves à réfléchir au(x) point(s) commun(s) qui existe entre les blocs d'une même catégorie.

Nouvelle mise en commun

Le ou les points communs entre les blocs d'une même catégorie permettent de présenter les noms des catégories.



Les blocs « **mouvement** » concernent la position, l'orientation et le mouvement des personnages et objets.

Les blocs « **Apparence** » permettent de créer des bulles de parole, de modifier l'apparence des personnages ou objets.

Les blocs « **stylo** » permettent de faire des tracés.

Les blocs « **événements** » permettent de déclencher les instructions placées en dessous.

Les blocs « **contrôle** » permettent d'attendre, de faire des boucles et des répétitions, de stopper un programme.

Les blocs « **opérateurs** » permettent de faire du calcul. Les blocs « **capteurs** » permettent de donner une action lorsqu'on appuie sur une touche du clavier ou si un personnage touche une couleur. Les blocs de ces deux catégories s'insèrent dans d'autres blocs.

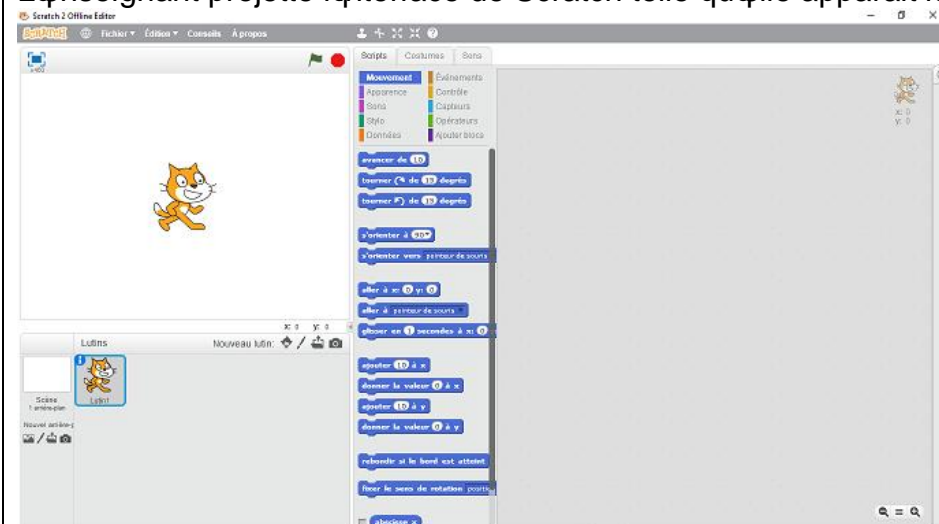
La version noir et blanc de l'annexe 3.3 peut être coloriée par les élèves.

Trace écrite

Dans Scratch, les instructions sont représentées par des blocs. Ils sont classés par catégories, et les blocs d'une catégorie ont tous la même couleur.

Phase 3 : présentation rapide de l'Interface de Scratch

L'enseignant projette l'interface de Scratch telle qu'elle apparaît lorsqu'on ouvre le logiciel.



Les différentes zones sont présentées. Voir fiche annexe 3.4.

LA SCENE

C'est l'endroit où les créations Scratch prennent vie

ZONE DES LUTINS ET DES ARRIERES-PLAN

Liste des personnages et objets (lutins) et liste des arrière-plans
Accès à leur gestion

PALETTES DES BLOCS

Blocs de programmation disponibles

ZONE DES SCRIPTS

Zone où seront assemblés les blocs pour construire les programmes

Il y a quelques mots de vocabulaire spécifiques à Scratch qui sera utile de préciser :

Lutin : c'est un personnage ou objet pour lequel on écrit un ou plusieurs programmes

Scène : c'est là où les créations prennent vie

Arrière-plan : c'est le décor de la scène

Script : c'est un ensemble de blocs formant un programme

Par défaut, à l'ouverture de Scratch, le lutin est un chat orange (c'est la mascotte de Scratch) et l'arrière-plan est blanc.

Montrer les catégories de blocs.

La découverte de Scratch se poursuivra au centre pilote où les élèves s'emploieront à programmer un jeu de labyrinthe.

Retour **SOMMAIRE**

Séance 4 au Centre Pilote Initiation à la programmation avec Scratch

Prérequis

- Utiliser le clavier et la souris.
- Savoir lancer un programme en double-cliquant sur son icône.
- Savoir enregistrer son travail dans un fichier, et ce fichier dans un dossier (ou répertoire).
- Savoir récupérer le travail que l'on a enregistré auparavant.

Objectifs

Découvrir et utiliser *Scratch*, un environnement de programmation graphique simple d'utilisation.

Notions

« Machines » et « Langages »

- On peut donner des instructions à une machine en utilisant un langage spécial, appelé langage de programmation, compréhensible par l'homme et la machine.
- Un "algorithme" est une méthode permettant de résoudre un problème
- Un programme est un algorithme exprimé dans un langage de programmation

Matériel

- Travailler en demi-groupe, un ordinateur pour 2 élèves
- Scratch doit être installé sur toutes les machines
- Les fichiers Scratch élèves doivent être installés sur toutes les machines
- par élève : Annexe 4.1 aide-mémoire Scratch
- pour l'animateur : annexes 4.2 (vidéo et fichier Scratch .sb2)

Phases de déroulement de l'activité

L'animateur explique aux enfants qu'ils vont utiliser un ordinateur pour créer un jeu simple de labyrinthe. Pour cela, ils vont devoir programmer l'ordinateur, c'est-à-dire lui dire quoi faire. Il faudra utiliser un langage spécial, un langage de programmation, compréhensible à la fois par les enfants et par l'ordinateur. Le langage qu'ils vont utiliser s'appelle *Scratch*.

Phase de découverte

L'interface de Scratch a été succinctement présentée en classe.

Reprendre une présentation de *Scratch* en interrogeant les élèves (les zones, les blocs).

L'animateur montre les actions élémentaires :

- comment déplacer un bloc en cliquant dessus et le glissant dans la zone de script ;
- comment assembler les blocs entre eux ;
- comment passer en mode plein écran ; comment en sortir ;
- comment déclencher les actions.

Spécifier pour cela de la fiche Annexe 3.4.

L'animateur distribue la fiche Annexe 4.1 (aide-mémoire Scratch) à laquelle les élèves pourront se reporter au besoin.

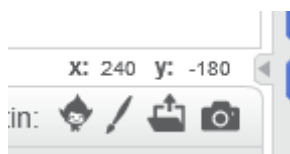
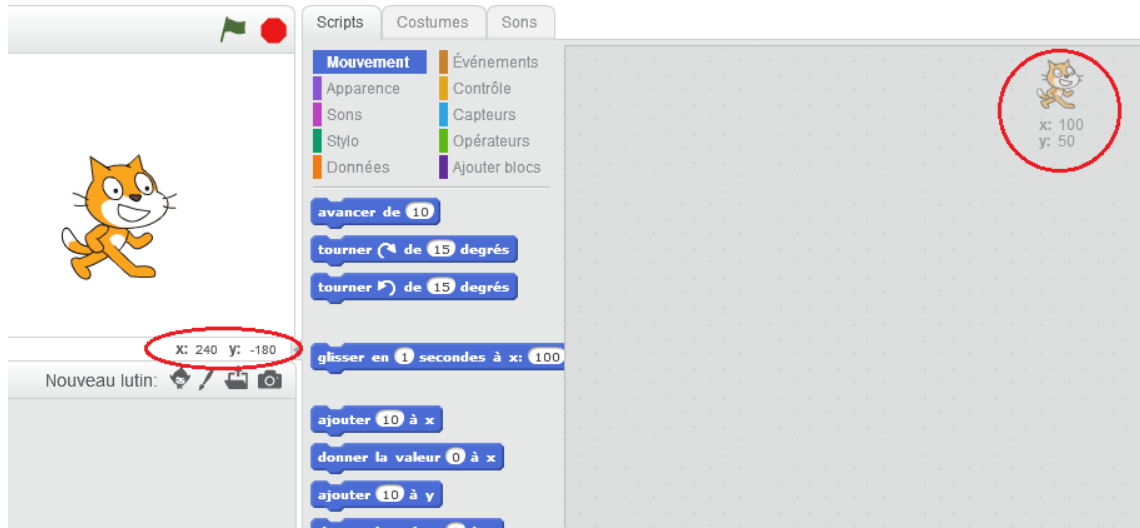
Laisser les élèves explorer librement le logiciel, puis exécuter des exercices simples :

Tâche 1 : les élèves lancent *Scratch* et se familiarisent avec son interface.

Les élèves regardent les différentes catégories d'instructions (« mouvement », « apparence », « événement », « contrôle »)

Tâche 2 : l'animateur demande aux élèves comment se repérer sur la scène.

L'utilisation des coordonnées x et y et de la grille Scratch a été vue lors de la séance 2 en classe.



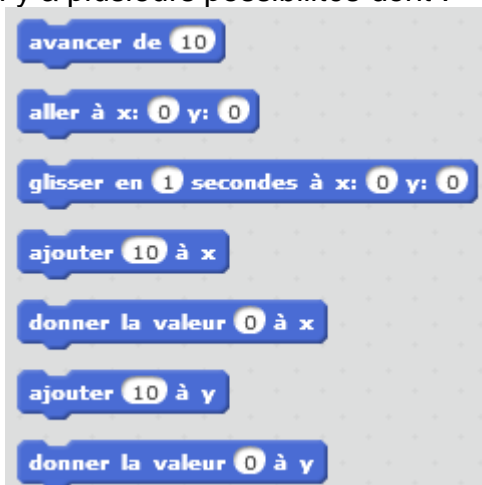
en bas à droite de la scène, **la position du pointeur de la souris** est indiquée.



en haut à droite dans la fenêtre de script, **la position du lutin** est indiquée.

Tâche 3 : l'animateur demande aux élèves quel(s) blocs utiliser pour déplacer le lutin chat sur la scène.

Il y a plusieurs possibilités dont :



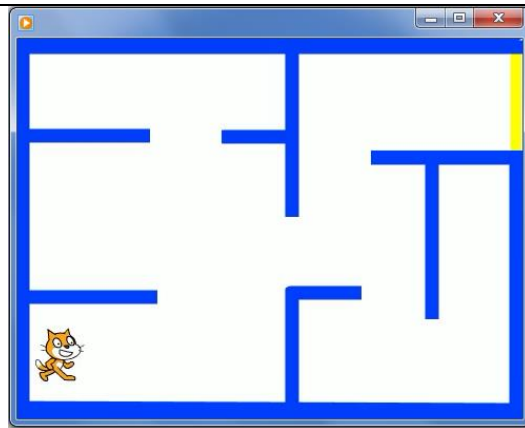
On retiendra pour la suite les instructions :



et

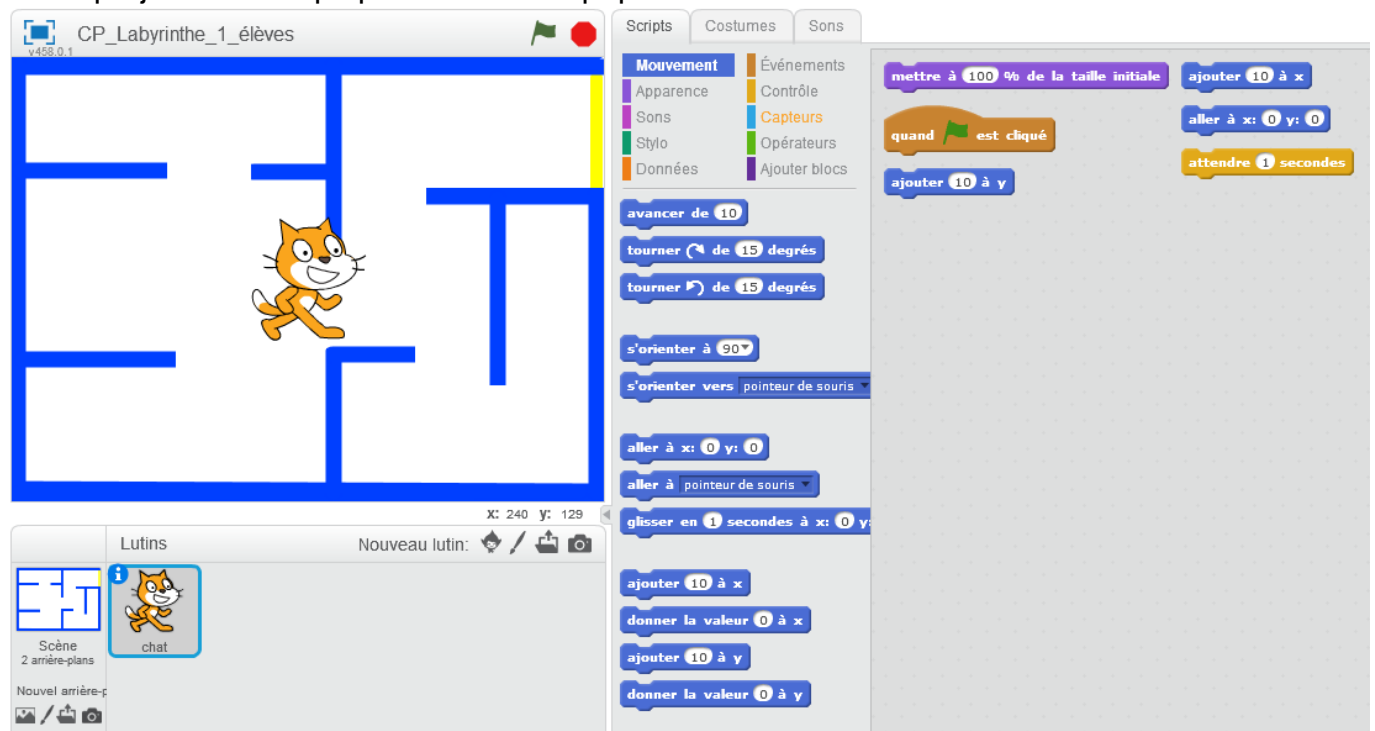


Tâche 4 : Montrer aux élèves la vidéo **Labyrinthe_1** (format mp4). Leur demander de décrire ce qu'il se passe : le chat orange se déplace dans un labyrinthe ; il part du bas à gauche pour arriver en haut à droite.



Les élèves vont devoir programmer ce déplacement.

L'animateur ouvre le fichier Scratch **CP_labyrinthe_1_eleves.sb2** sur l'ordinateur relié au vidéo projecteur et explique aux élèves qu'ils vont travailler avec ce fichier.



Que voit-on ?

Il y a un lutin, le chat orange.

L'arrière-plan est un labyrinthe, murs bleus et jaune sur fond blanc.

Des blocs sont déjà présents dans la fenêtre de scripts : ce sont les blocs utiles pour le programme. *Certains blocs seront à utiliser plusieurs fois.*

Que faire ? Avant de se lancer tête baissée, on réfléchit aux différentes actions qu'il va falloir programmer et à leur enchaînement. Bref on pense algorithme.

On voit pour commencer que le chat est mal dimensionné (trop gros) : il faut réduire sa taille.

On voit que le chat est au centre alors que la position de départ sur la vidéo est en bas à gauche.

Ensuite il faut programmer le déplacement du chat jusqu'au mur jaune.

Une fois ces informations données, les élèves peuvent ouvrir le fichier Scratch **CP_labyrinthe_1_eleves.sb2** et travailler au programme.

Pour réduire la taille du lutin, on utilise le bloc **Apparence** **mettre à 100 % de la taille initiale**.

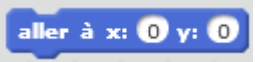

- 100% correspond à la taille normale du lutin

- avec une valeur plus petite que 100, la taille du lutin est réduite
- avec une valeur plus grande que 100, la taille du lutin est augmentée



Laisser les élèves tester différentes valeurs (inférieures à 100) pour trouver la taille adéquate. Pour exécuter l'instruction, il suffit de cliquer sur le bloc dans la fenêtre de script. On visualise immédiatement le résultat.

Une valeur autour de 50% est adaptée.

Ensuite, le chat doit au départ être positionné en bas à gauche de la scène. Utiliser le bloc



Mouvement  et modifier les valeurs pour x et y.  par exemple convient.

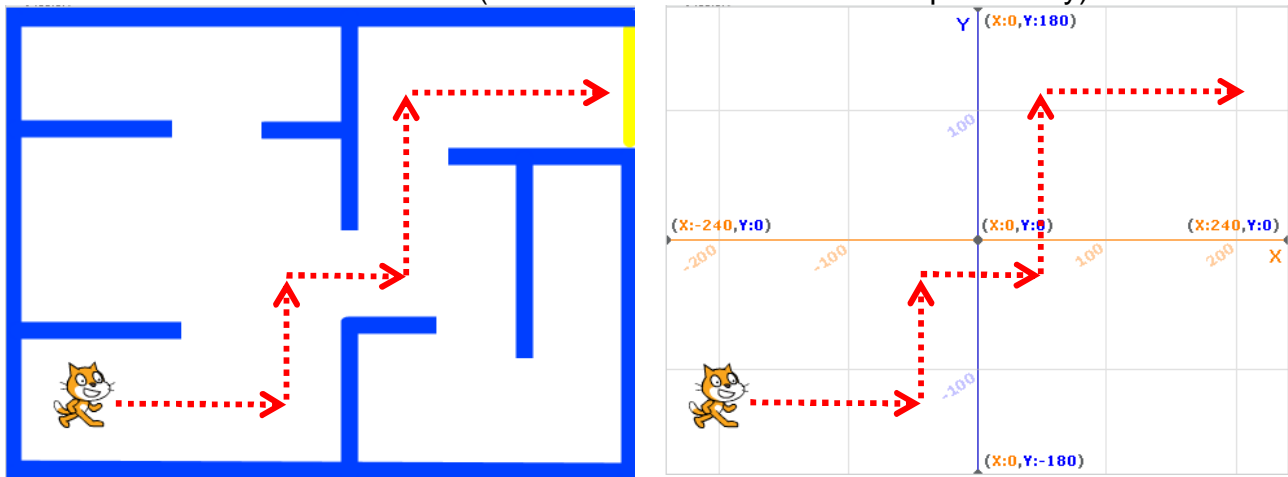
Pour exécuter l'instruction, il suffit de cliquer sur le bloc dans la fenêtre de script. On peut aussi


insérer le bloc **Evènement**  ce qui donne  et déclencher le script en cliquant sur le drapeau vert.

Remarque : les élèves peuvent proposer des valeurs légèrement différentes dans les champs des blocs, cela n'a pas d'importance tant que cela répond à la problématique.

Pour programmer le déplacement lui-même, on utilise les blocs

  (dont il faut modifier les valeurs pour x et y).



Entre ces blocs, il est utile d'insérer un bloc  pour qu'il y ait des pauses dans le déplacement du lutin, sinon on aura l'impression qu'il ne bouge pas ! Il enchaîne en fait trop rapidement les déplacements.

On n'oublie pas le bloc  en début de script si on ne l'a pas encore utilisé.

Une solution possible est :

```

quand est cliqué
mettre à 50 % de la taille initiale
aller à x: -200 y: -120
attendre 0.5 secondes
ajouter 170 à x
attendre 0.5 secondes
ajouter 100 à y
attendre 0.5 secondes
ajouter 90 à x
attendre 0.5 secondes
ajouter 140 à y
attendre 0.5 secondes
ajouter 150 à x

```

Remarque :
 si la taille d'un lutin n'est pas modifiée au cours du programme, on peut ne pas insérer le bloc
 mettre à 50 % de la taille initiale dans le script principal comme ici.

La classe synthétise collectivement ce qui a été appris au cours de cette séance :
 Avant de programmer, on réfléchit aux différentes actions du lutin et à leur enchainement.
 On peut tester le résultat du script au fur et à mesure de sa construction.
 Dans un programme si un lutin se déplace, il ne faut pas oublier de donner sa position de départ au début du script.

Remarques

Labyrinthe 1 : déplacements du chat

```

quand est cliqué
aller à x: -200 y: -120
attendre 0.5 secondes
ajouter 170 à x
attendre 0.5 secondes
ajouter 100 à y
attendre 0.5 secondes
ajouter 90 à x
attendre 0.5 secondes
ajouter 140 à y
attendre 0.5 secondes
ajouter 150 à x

```

Autres blocs de mouvement pouvant être utilisés pour déplacer le chat :

```

aller à x: y:
glisser en secondes à x: y:
donner la valeur à x
donner la valeur à y

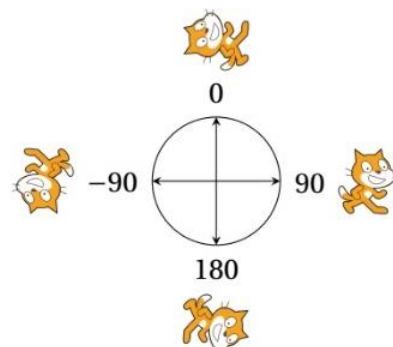
```

Le bloc « avancer de » nécessite d'utiliser le bloc « s'orienter à » car le lutin avance dans le sens dans lequel il est orienté (vers lequel il regarde). Par défaut le chat regarde vers la droite, il est orienté à 90 (degrés).

```

avancer de 10
s'orienter à 90
(90) à droite
(-90) à gauche
(0) vers le haut
(180) vers le bas

```



le lutin regarde vers le haut s'il est orienté à 0 (degrés), vers la droite s'il est orienté à 90 (degrés), vers le bas s'il est orienté à 180 (degrés), vers la gauche la tête en bas s'il est orienté à -90 (degrés) : **pour cette raison, préférez les autres blocs mouvement avec x et y.**

De nombreuses séances sont nécessaires pour utiliser et comprendre le logiciel Scratch.

Des ressources avec des activités complémentaires proposées à la fin de ce document, permettant de poursuivre le travail sur ce logiciel, en classe.

Les exercices sont progressifs, à réaliser en autonomie ou de façon accompagnée.

Par ailleurs, d'autres jeux sont proposés en ligne :

- <http://castor-informatique.fr>
- <http://code.org>

Séance 5 au Centre Pilote Programmer un jeu de labyrinthe partie 1

Objectifs

Utiliser *Scratch*, un environnement de programmation graphique simple d'utilisation.

Notions

- Les scripts sont déclenchés par des événements.
- Certaines boucles sont répétées jusqu'à ce qu'une condition soit remplie.

Matériel

- Travailler en demi-groupe, un ordinateur pour 2 élèves
- Scratch doit être installé sur toutes les machines
- Les fichiers Scratch élèves doivent être installés sur toutes les machines
- par élève : Annexe 4.1 aide-mémoire Scratch
- pour l'animateur : annexes 5.1 (vidéo et fichier Scratch .sb2)

Phases de déroulement de l'activité

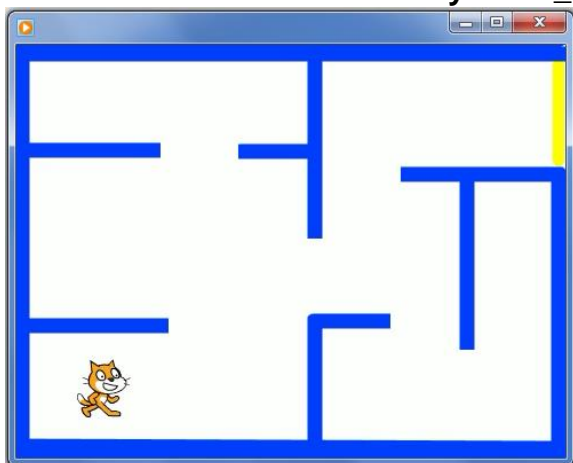
Phase 1

On veut programmer **un jeu** de labyrinthe. Il faut que le joueur puisse contrôler les déplacements du chat.

Demander aux élèves comment faire ?

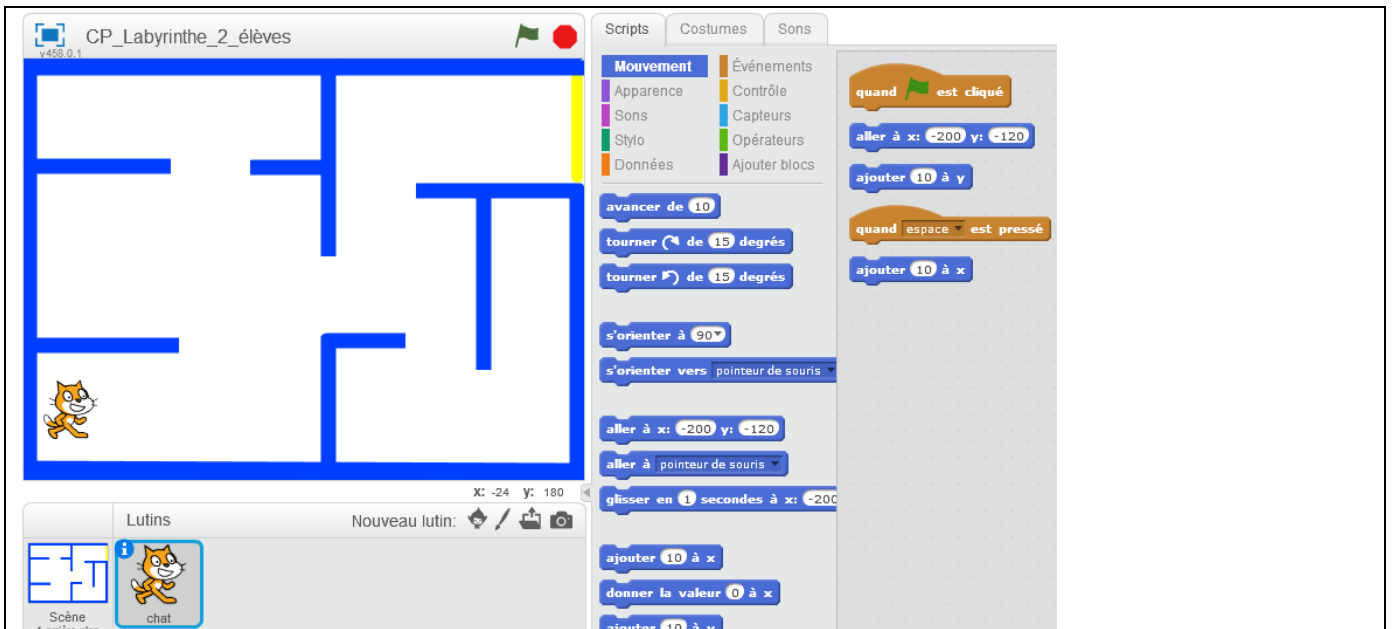
- Ils vont certainement suggérer d'utiliser des touches du clavier pour commander le lutin (ici pas de manette de jeu comme sur les consoles, ni écran tactile).

Montrer aux élèves la vidéo **Labyrinthe_2** (format mp4).



Le lutin effectue de petits déplacements à droite, à gauche, vers le haut et vers le bas. Ces déplacements sont contrôlés par des touches du clavier de l'ordinateur.

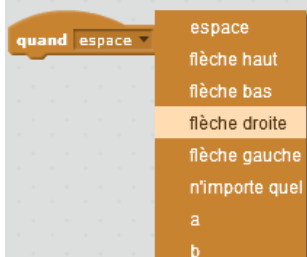
L'animateur ouvre le fichier Scratch **CP_Labyrinthe_2_eleves.sb2** sur l'ordinateur relié au vidéo projecteur et dit aux élèves qu'ils vont travailler avec ce fichier.



Un nouveau bloc **Evènement** est proposé :

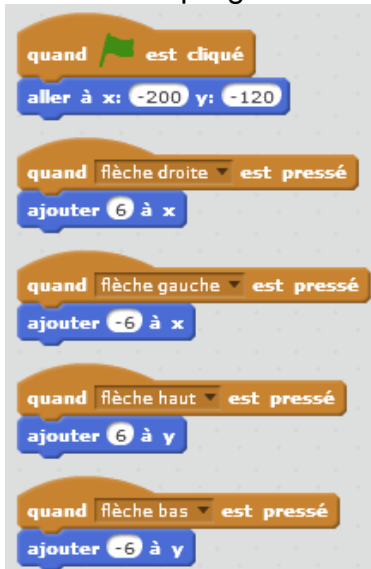


En déroulant le menu du bloc, on peut choisir la touche du clavier qui déclenchera l'action.



On va choisir ici les touches flèches du clavier, mais on aurait pu en choisir d'autres

Les élèves ouvrent à leur tour le fichier **CP_Labyrinthe_2_eleves.sb2** sur leur ordinateur et travaillent au programme.



Solution possible :

La valeur du déplacement unitaire en x et y (correspondant à un appui sur la touche) est à ajuster. Elle ne doit pas être trop grande (jeu trop facile) ni trop petite ! La valeur de 6 ou -6 est proposée ici.

Rappel : utilisation de nombres négatifs pour aller vers la gauche (-x) et vers le bas (-y) de la scène.

Ne pas oublier de définir la position de départ (ici x = -200 et y = -120) afin que le chat soit au bon endroit à chaque début de partie.

Il y a 5 blocs évènements différents: chaque script correspondant sera exécuté lorsque l'évènement aura lieu (appui sur drapeau vert pour lancer le jeu, appui sur une touche flèche pour contrôler les déplacements du lutin).

Chaque script peut être testé au fur et à mesure de sa construction.

Les élèves vont sans doute se rendre compte que le lutin peut traverser les murs du labyrinthe ! Ils ne s'en rendent pas compte seuls, l'animateur en fait la démonstration.

C'est un problème : comment faire pour que le chat ne puisse pas franchir les murs du labyrinthe. Relever les idées et propositions des élèves.

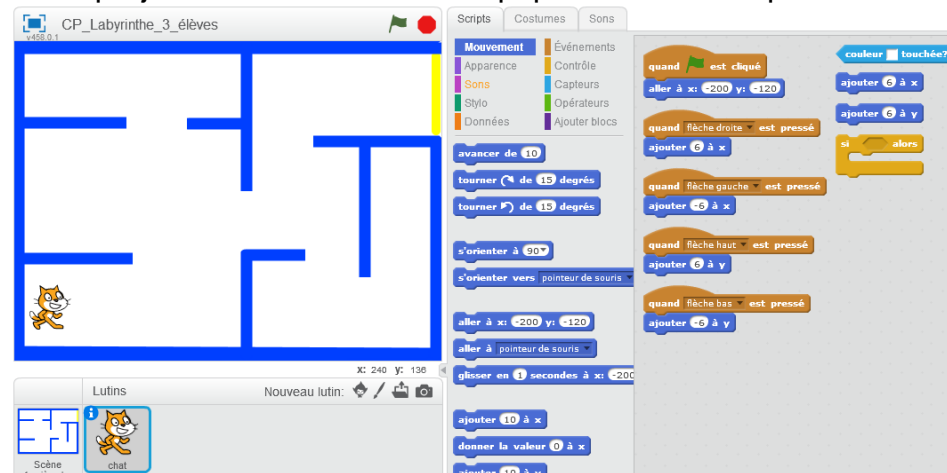
→ Proposition probable des élèves : Si le chat touche un mur, alors il s'arrête.

Notions de test et de condition vues en classe séance 1.

Comment traduire cela dans Scratch ?

Premièrement : quand le chat touche le mur.

L'animateur ouvre le fichier Scratch **CP_Labyrinthe_3_eleves.sb2** sur l'ordinateur relié au vidéo projecteur et dit aux élèves qu'ils vont travailler pour cette nouvelle étape avec ce fichier.



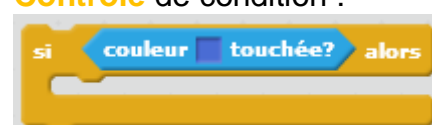
Remarque : la partie du programme construite précédemment est déjà en place dans ce nouveau fichier. Ce sera le cas pour les fichiers suivants lors de chaque nouvelle étape.

Les élèves ouvrent à leur tour le fichier Scratch **CP_labyrinthe_3_eleves.sb2** sur leur ordinateur et travaillent au programme.

Nouveaux blocs



Le bloc **Capteur** avec sa forme pointue s'insère dans le bloc **Contrôle** de condition :



Pour sélectionner la couleur exacte des murs, cliquer sur la zone carrée du bloc (ici blanche)




. Le pointeur de la souris passe de la forme « flèche » à la forme « main » ; cliquer alors sur la couleur de votre choix dans l'écran (ici sur un mur bleu du labyrinthe de la scène).

Que mettre à l'intérieur du bloc condition ?

Deuxièmement : le chat s'arrête

Il n'y a pas de bloc dans la catégorie **Mouvement** pour arrêter un lutin.

Il existe un bloc **Contrôle**  mais sa fonction est de stopper tout le script du lutin : le

jeu s'arrêterait. Il n'est pas proposé ici.

Les autres blocs proposés sont



Pour aider les élèves à trouver la solution, leur faire tester les scripts ci-dessous :

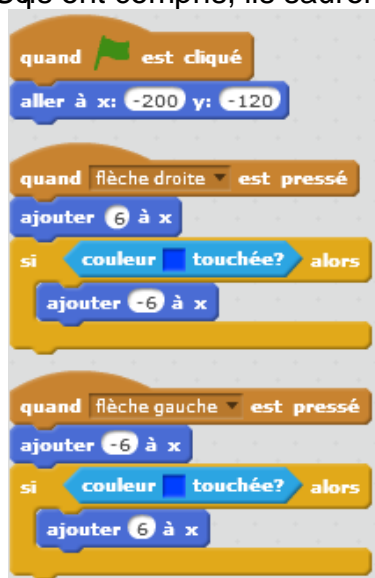
①	②	③	④
<p>Le chat avance à droite, attend puis recule et revient à sa position de départ.</p>	<p>Avec un temps d'attente plus court, le chat va plus vite. (expliquer les chiffres à virgule (ici point !))</p>		<p>Sans bloc d'attente, on ne voit plus le chat bouger, il va trop vite !</p>

La solution est donc ici de programmer un déplacement inverse de celui que le chat a fait précédemment, pour donner l'illusion qu'il n'avance pas, que le mur le bloque !

Si besoin, construisez avec les élèves le script correspondant à l'un des déplacements, vers le haut par exemple :



Si ils ont compris, ils sauront faire la même chose pour les trois autres déplacements.



Conclusions

La classe synthétise collectivement ce qui a été appris au cours de cette séance :
On peut contrôler les déplacements d'un lutin à l'écran à l'aide des touches du clavier.
Pour donner l'illusion qu'un lutin s'arrête devant un obstacle, on le fait avancer puis reculer très vite.

Séance 6 au Centre Pilote Programmer un jeu de labyrinthe partie 2

Objectifs

Utiliser *Scratch*, un environnement de programmation graphique simple d'utilisation.

Notions

- Des instructions peuvent démarrer en même temps, si leur exécution est déclenchée par un même événement.
- Certaines boucles sont répétées jusqu'à ce qu'une condition soit remplie, et certaines boucles sont répétées indéfiniment.

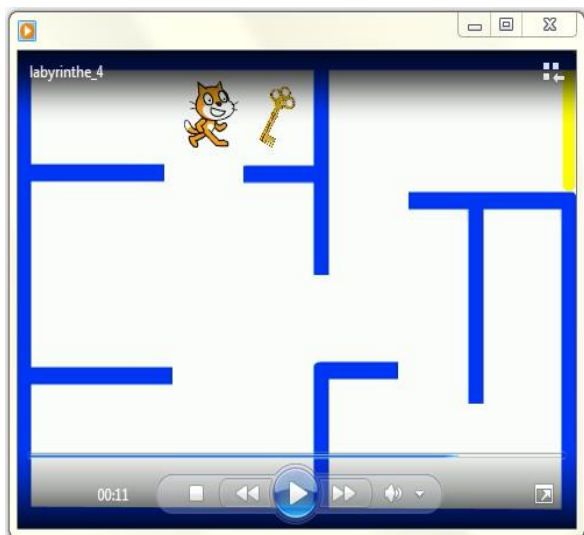
Matériel

- Travailler en demi-groupe, un ordinateur pour 2 élèves
- Scratch doit être installé sur toutes les machines
- par élève : Annexe 4.1 aide-mémoire Scratch
- pour l'animateur : annexes 6.1 (vidéo et fichier Scratch .sb2)

Phases de déroulement de l'activité

Phase 1

Montrer aux élèves la vidéo **Labyrinthe_4** (format mp4).

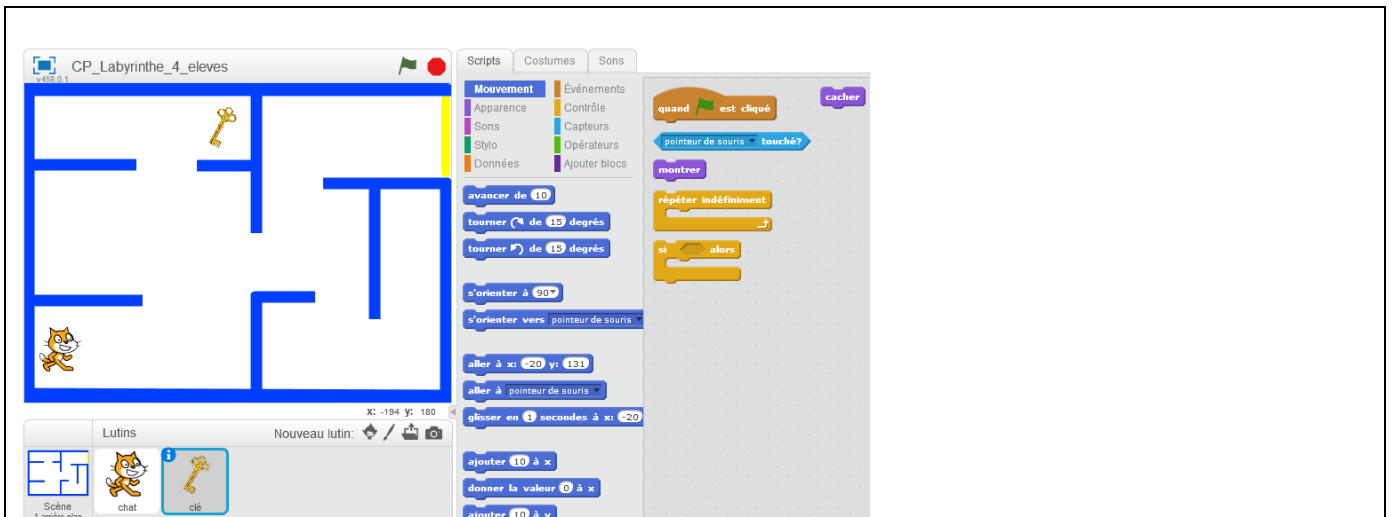


Faire le point avec les élèves : dans la vidéo, on voit le chat se déplacer jusqu'à la clé ; lorsqu'il touche la clé, celle-ci disparaît.

Etape suivante dans le jeu : le chat doit ramasser un objet avant de sortir du labyrinthe.

L'animateur ouvre le fichier Scratch **CP_Labyrinthe_4_eleves.sb2** sur l'ordinateur relié au vidéo projecteur et dit aux élèves qu'ils vont travailler pour cette nouvelle étape avec ce fichier.


Un nouveau lutin est présent, on voit sa vignette dans la zone des lutins sous la scène.



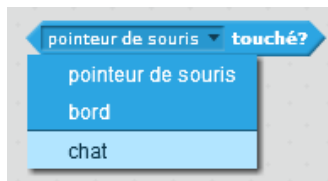
Les élèves ouvrent à leur tour le fichier **CP_Labyrinthe_4_eleves.sb2** sur leur ordinateur et travaillent au programme.



Pour le script du lutin clé, les blocs ci-contre vont être utilisés. Laisser réfléchir les élèves à leur utilisation.

Blocs **Apparence**   : la clé doit disparaître (illusion que le chat la ramasse).

Blocs **Contrôle**  



Bloc **Capteur** dont le menu déroulant permet de poser la question « le lutin chat est touché ? »

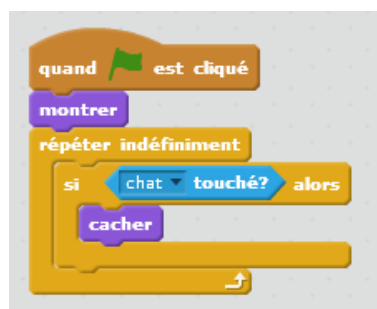
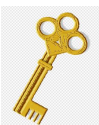
Pensons algorithmes : faire le point avec les élèves.

- au début du jeu, la clé doit être visible dans le labyrinthe



- le chat se déplace, contrôlé par les touches flèches du clavier : les scripts pour le lutin chat ne sont pas modifiés.
- Lorsque le chat arrive à la clé, il la touche (la clé est aussi touchée par le chat) et elle doit disparaître.

Script pour le lutin clé



Il y a action réciproque : si le chat touche la clé, alors la clé touche aussi le chat. Le script est écrit pour la clé car c'est elle qui disparaît (cela donne l'illusion qu'elle est prise par le chat).

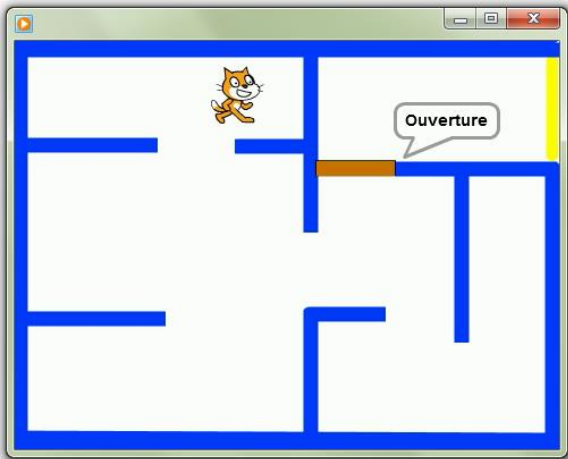
Il faut un bloc de répétition infinie, sinon, le test (si le lutin chat est touché) ne serait effectué

qu'une seule fois et le script s'arrêterait. Le test doit être effectué en permanence.

La nécessité du bloc de répétition infinie n'est pas facile à comprendre, mais le fait de pouvoir tester rapidement les scripts dans Scratch permet de se rendre compte que sans ce bloc, cela ne fonctionne pas, le script ne donne pas le résultat attendu.

Remarque : il n'est pas nécessaire de préciser la position initiale du lutin clé (il ne se déplace pas dans le jeu).

Passons à l'étape suivante dans la création du jeu :
Montrer aux élèves la vidéo **Labyrinthe_5** (format mp4).

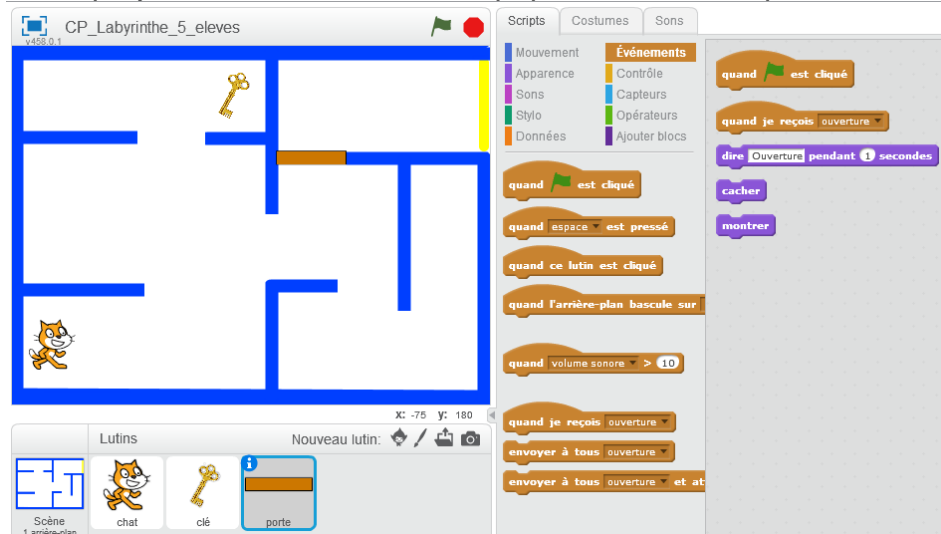


Qu'y a-t-il de nouveau ?

Il y a une porte brune qui bloque l'accès à l'arrivée.

Lorsque le chat ramasse la clé, un message s'affiche et cette porte s'ouvre, libérant l'accès au couloir de l'arrivée.

L'animateur ouvre le fichier Scratch **CP_Labyrinthe_5_eleves.sb2** sur l'ordinateur relié au vidéo projecteur et dit aux élèves qu'ils vont travailler pour cette nouvelle étape avec ce fichier.



Le nouveau lutin-porte est présent, on voit sa vignette dans la zone des lutins sous la scène.

Pensons algorithmique : mener une réflexion commune avec les élèves.

Le script du lutin porte est lié à celui du lutin clé : c'est parce que la clé est ramassée (touchée) que la porte s'ouvre. Le lutin clé doit communiquer avec le lutin porte.

Les élèves ouvrent à leur tour le fichier **CP_Labyrinthe_5_eleves.sb2** sur leur ordinateur et travaillent au programme.

Regardons de plus près les blocs pour le lutin porte :

On retrouve les blocs **Apparence** **montrer** **cacher** comme pour la clé.

Il y a un bloc **Apparence** **dire Ouverture pendant 1 secondes**

Il y a un bloc **Evènement**



Regardons également de plus près les blocs pour le lutin clé :



En plus du script déjà construit précédemment, il y a un bloc **Evènement**



C'est grâce aux blocs **Evènement**



et



que les deux lutins communiquent.

Le lutin clé envoie un message (qui s'appelle « ouverture ») à tous les lutins.

Quand le lutin porte reçoit le message « ouverture », alors une action peut avoir lieu.

Script pour le lutin

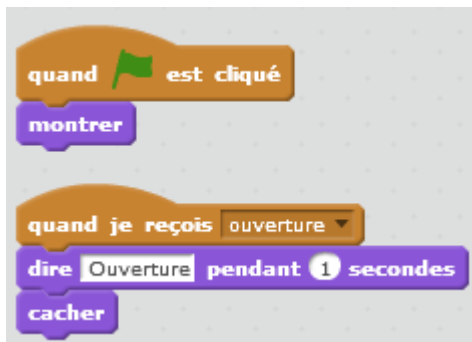


clé



*Le lutin clé communique avec le lutin porte : il lui envoie un message (bloc **Evènement**) pour lui signifier qu'il va devoir faire une action, ici se cacher, donnant l'illusion de s'ouvrir. Ce bloc d'envoi de message est inséré dans le bloc de condition.*

Script pour le lutin porte



*Quand le lutin porte reçoit le message du lutin clé (bloc **Evènement**) lui demandant de s'ouvrir (de se cacher), alors le lutin porte obéit.*

*Ne pas oublier le bloc **Apparence** « montrer » en début de script, sinon le lutin porte est caché à la fin de la première exécution du jeu et on ne le voit plus dans les exécutions suivantes !*

Attention, la porte n'est pas infranchissable pour le chat !

Ajouter dans le script du chat les instructions nécessaires pour que le lutin ne puisse pas franchir la porte.

Les élèves ont fait ce travail précédemment avec les murs du labyrinthe.

En fonction du temps et de l'avancement du projet, cette partie peut être ou non travaillée avec les élèves.



Partie du script du lutin chat :

Ces blocs sont sous le script de déplacement vers le haut, la porte n'est en effet accessible que par un déplacement vers le haut.

Conclusions

La classe synthétise collectivement ce qui a été appris au cours de cette séance :

- Cacher un lutin donne l'illusion qu'il disparaît.
- Il est parfois nécessaire de répéter des conditions indéfiniment.
- Des lutins peuvent communiquer entre eux : en envoyant un message, un lutin en informe un autre qui aura une ou plusieurs actions à accomplir.

Séance 7 au Centre Pilote Programmer un jeu de labyrinthe partie 3

Objectifs

Utiliser *Scratch*, un environnement de programmation graphique simple d'utilisation.

Notions

Des instructions peuvent démarrer en même temps, si leur exécution est déclenchée par un même événement.

Certaines boucles sont répétées jusqu'à ce qu'une condition soit remplie. D'autres sont répétées indéfiniment.

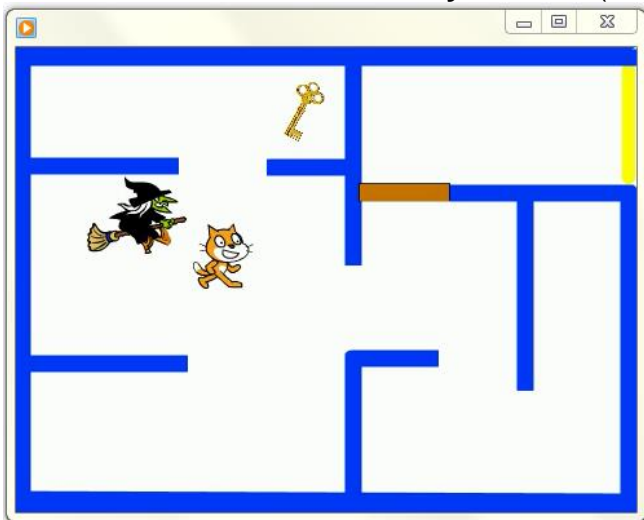
Matériel

- Travailler en demi-groupe, un ordinateur pour 2 élèves
- Scratch doit être installé sur toutes les machines
- pour l'animateur : annexes 7.1 (vidéo et fichier Scratch .sb2)

Phases de déroulement de l'activité

Etape suivante dans la création du jeu :

Montrer aux élèves la vidéo **Labyrinthe_6** (format mp4).

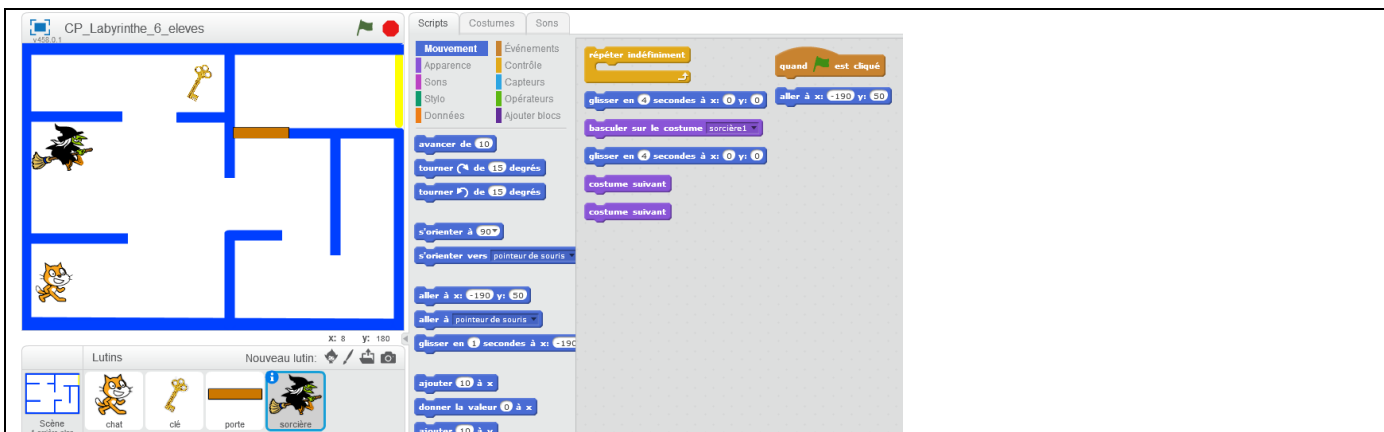


Qu'y a-t-il de nouveau ?

Les choses se compliquent : une sorcière garde la clé !

Si le chat touche la sorcière, il retourne à son point de départ.

L'animateur ouvre le fichier Scratch **CP_Labyrinthe_6_eleves.sb2** sur l'ordinateur relié au vidéo projecteur et dit aux élèves qu'ils vont travailler pour cette nouvelle étape avec ce fichier.



Le nouveau lutin-sorcière est présent, on voit sa vignette dans la zone des lutins sous la scène.

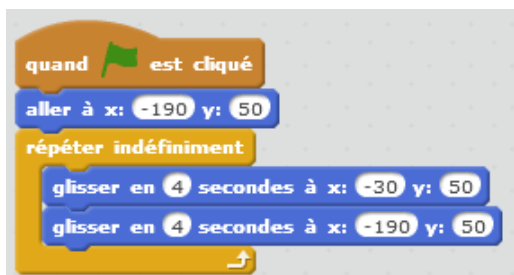
Les élèves ouvrent à leur tour le fichier **CP_Labyrinthe_6_eleves.sb2** sur leur ordinateur et travaillent au programme.

Pensons algorithmes : mener une réflexion commune avec les élèves.

La sorcière se déplace devant le passage menant à la clé. Elle fait demi-tour à chaque aller et chaque retour.

Si le chat touche la sorcière, il retourne à son point de départ.

Script pour le lutin-sorcière



On ne oublie pas de définir la position de départ de la sorcière au début du script, afin qu'elle reparte bien toujours du même endroit à chaque début de partie.

On utilise le bloc de répétition infinie.

Nouveaux blocs **Apparence** :

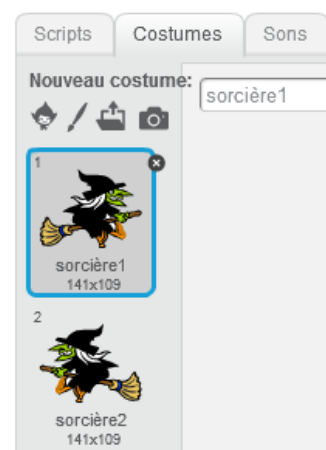


Avec le script ci-dessus, la sorcière avance et recule devant l'accès à la clé ; faisons-lui faire des allers-retours pour un rendu plus naturel.

Le lutin-sorcière a deux costumes symétriques : le premier regarde vers la droite, le second vers la gauche.

On a ajouté les blocs **Apparence** « costume suivant » après chaque déplacement pour avoir l'impression que la sorcière fait des allers-retours.

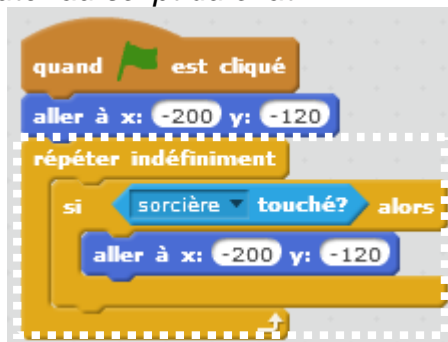
Il ne faut pas oublier d'initialiser le costume de départ.





Maintenant que la sorcière se déplace, il faut qu'elle joue son rôle de gardienne de la clé. Si le chat est touché par la sorcière, celui-ci retourne au départ et doit recommencer. Les blocs sont donc à ajouter au script du chat !

Début du script pour le lutin chat



On n'oublie pas la répétition infinie afin que le test soit effectué en permanence et non une seule fois !

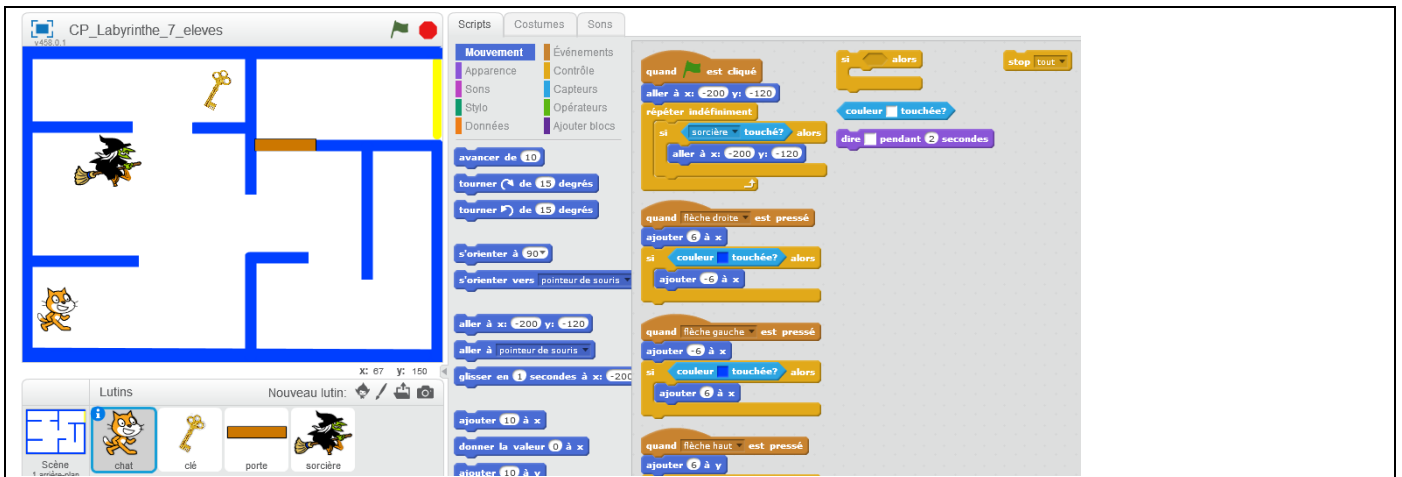
Dernière étape dans la création du jeu : (si le temps le permet)
Montrer aux élèves la vidéo **Labyrinthe_7** (format mp4).



Qu'y a-t-il de nouveau ?

Quand le chat touche le mur jaune, il dit qu'il a gagné, le jeu s'arrête.

L'animateur ouvre le fichier Scratch **CP_Labyrinthe_7_eleves.sb2** sur l'ordinateur relié au vidéo projecteur et dit aux élèves qu'ils vont travailler pour cette nouvelle étape avec ce fichier.



Les élèves ouvrent à leur tour le fichier **CP_Labyrinthe_7_eleves.sb2** sur leur ordinateur et travaillent au programme.

Les élèves ont déjà travaillé à ce type de script avec le lutin-clé (si la couleur jaune est touchée alors 0).

Un seul bloc nouveau : . Ce bloc **Contrôle** met fin à tous les scripts commençant par le bloc .

Script pour le lutin chat



```

quand flèche droite est pressé
ajouter 6 à x
si couleur touchée? alors
ajouter -6 à x
si couleur touchée? alors
dire Gagné! pendant 2 secondes
stop tout
  
```

Ces blocs sont sous le script de déplacement à droite, le mur jaune n'est en effet accessible que par un déplacement vers la droite.

Conclusions

La classe synthétise collectivement ce qui a été appris au cours de cette séance :

Un lutin peut avoir plusieurs costumes et en changer au cours d'un script.

Il faut parfois utiliser le bloc de répétition infinie pour qu'une condition soit testée tout au long du script.

Remarques

Améliorations possibles du jeu :

Plusieurs objets à ramasser, plusieurs gardiens (les élèves devraient savoir programmer seuls)

Un temps limité pour accomplir la mission

Un nombre de vies limité (une vie en moins quand la sorcière touche le chat)

Plusieurs niveaux

etc.

Récapitulatif des Scripts pour le jeu du labyrinthe



Scripts pour le lutin chat

Position de départ

Si le chat est touché par la sorcière, il retourne au départ

Le chat se déplace vers la droite si la touche flèche droite du clavier est pressée

Le chat recule s'il touche un mur bleu : on a l'impression qu'il ne bouge pas (avance + recule)

Si le chat touche le mur jaune, il dit « Gagné ! » et le script s'arrête

Le chat se déplace vers la gauche si la touche flèche gauche du clavier est pressée

Le chat avance s'il touche un mur bleu : on a l'impression qu'il ne bouge pas (recule + avance)

Le chat se déplace vers le haut si la touche flèche haut du clavier est pressée

Le chat descend s'il touche un mur bleu : on a l'impression qu'il ne bouge pas (monte + descend)

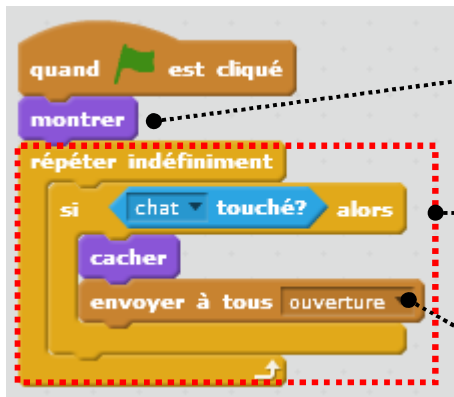
Le chat descend s'il touche la porte : on a l'impression qu'il ne bouge pas (monte + descend)

Le chat se déplace vers le bas si la touche flèche bas du clavier est pressée

Le chat monte s'il touche un mur bleu : on a l'impression qu'il ne bouge pas (descend + monte)

The image displays a Scratch script for a maze game, organized into several sections separated by red dashed lines. Each section contains code blocks with corresponding annotations on the right side, connected by dotted lines. The script starts with a 'when green flag is clicked' event, followed by a 'go to x: -200 y: -120' block. A 'repeat indefinitely' loop contains a 'if witch touched?' condition that triggers a 'go to x: -200 y: -120' block. The main movement logic consists of 'when arrow key is pressed' events for right, left, up, and down. Each event block includes an 'add 6 to x' or 'add 6 to y' block, followed by 'if blue wall touched?' or 'if yellow wall touched?' conditions. These conditions trigger 'add -6 to x' or 'add -6 to y' blocks to prevent movement through walls. The yellow wall condition triggers a 'say Gagné! for 2 seconds' block and a 'stop all' block. The script is annotated with 12 text descriptions explaining the purpose of each part of the code.

Script pour le lutin clé

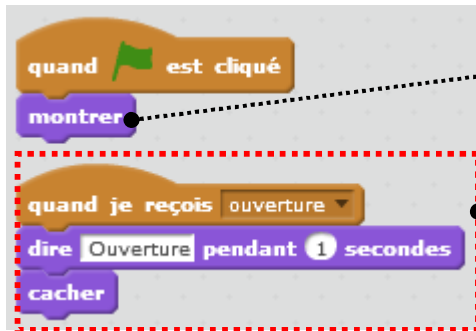


Au départ, la clé est visible (montrée)

Si la clé est touchée par le chat elle disparaît (est cachée)

Lorsque la clé est touchée, envoi d'un message aux autres lutins, dont la porte

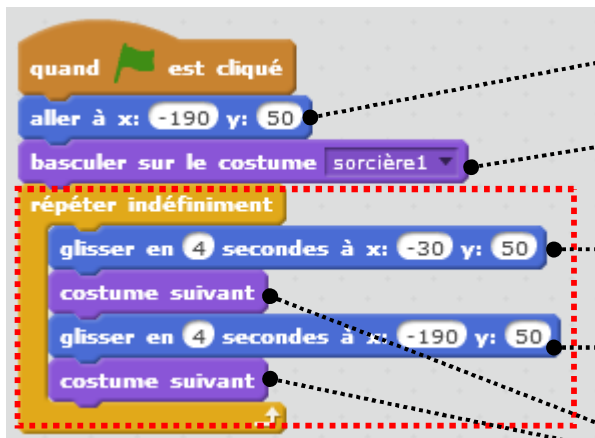
Scripts pour le lutin porte



Au départ, la porte est visible (montrée)

Lorsque la porte reçoit le message envoyé par la clé, la porte affiche une bulle pour dire quelle s'ouvre puis elle disparaît (est cachée)

Script pour le lutin sorcière



Position de départ

Costume de départ

Déplacement aller

Déplacement retour

Changement de costume pour changement de direction de la sorcière

ANNEXES

Annexe 1.1 : des algorithmes au quotidien

Exercice 1

Tous les matins, au moment de se lever pour aller à l'école Lucas a du mal à se réveiller. Il sort du lit encore endormi et ne sait plus ce qu'il doit faire pour se préparer.

Aide Lucas en découpant les étiquettes et en les collant dans l'ordre (il y a plusieurs possibilités).

Laver son bol de céréales	Prendre une douche	Mettre ses chaussures
Se laver les dents	Mettre son manteau	Mettre ses sous-vêtements
Déjeuner	Enlever son pyjama	Mettre ses vêtements
Se coiffer	Prendre son sac d'école	

Exercice 2

Avant de partir pour l'école, Lucas doit choisir s'il met un manteau ou pas, s'il prend un parapluie ou pas. Son choix dépend bien sûr de la météo.

A l'aide des étiquettes suivantes, reconstitue un algorithme que pourrait suivre Lucas pour choisir sa tenue.

mettre un manteau	Si	il fait froid
prendre un parapluie	Regarder la météo	alors
Si	il pleut	alors

Les deux conditions peuvent être vraies toutes les deux ; alors Lucas met un manteau et prend un parapluie.

Exercice 3

Au petit-déjeuner, Lucas peut manger des tartines jusqu'à ce qu'il n'ait plus faim et boire du jus de fruit tant qu'il a soif.

Reconstitue avec les étiquettes suivantes un algorithme pour le petit-déjeuner de Lucas afin qu'il n'ait ni faim ni soif en partant à l'école.

j'ai soif	Tant que	Tant que
manger une tartine	Laver son verre	Ranger la cuisine
boire du jus de fruit	j'ai faim	

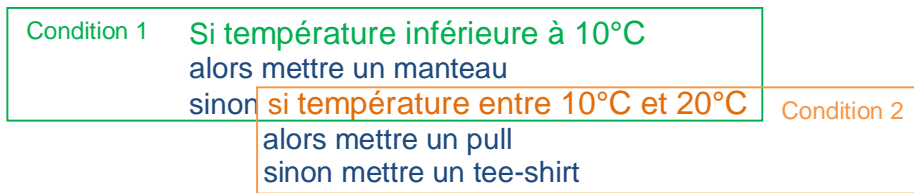
Exercice 4

Lucas choisit ses vêtements en fonction de la météo. Si la température est inférieure à 10°C, il met un manteau ; si la température est comprise entre 10°C et 20°C, il met un pull, et si la température est supérieure à 20°C, il met un tee-shirt.

Propose un algorithme que pourrait suivre Lucas pour choisir sa tenue.

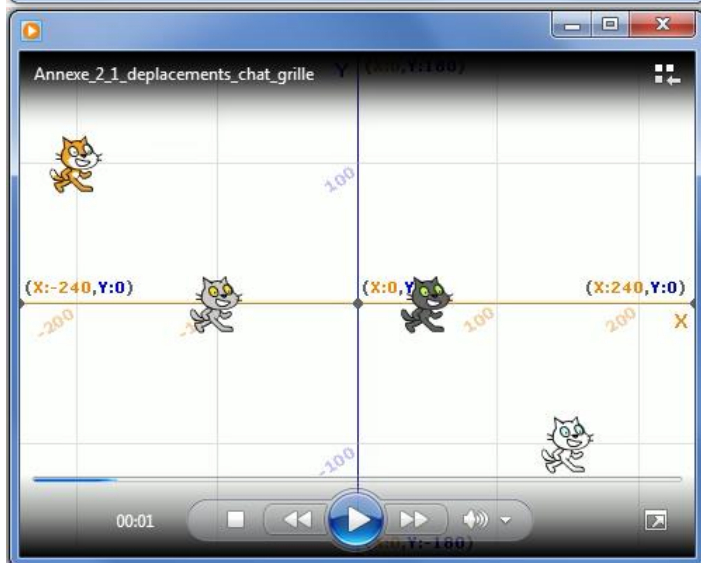
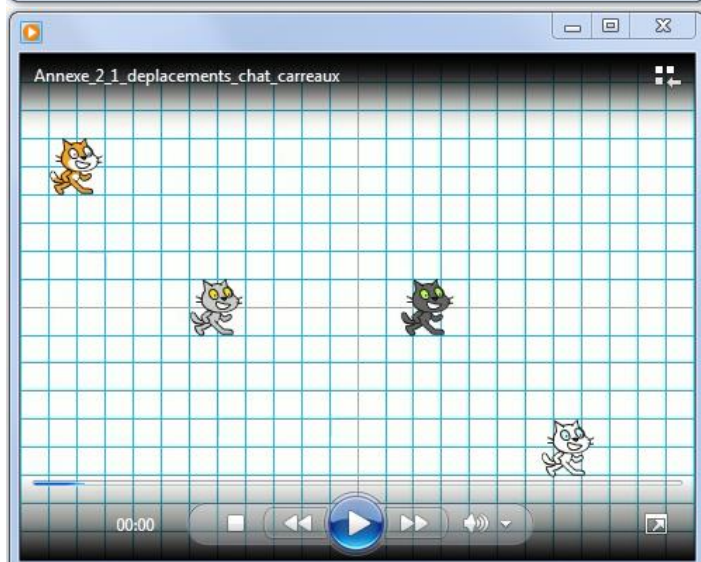
Aide sur les conditions : elles s'imbriquent l'une dans l'autre ; c'est le même paramètre (la température) qui est testé.

Pour distinguer les conditions et rendre plus lisible l'algorithme, on décale (indente) l'alignement de la condition suivante

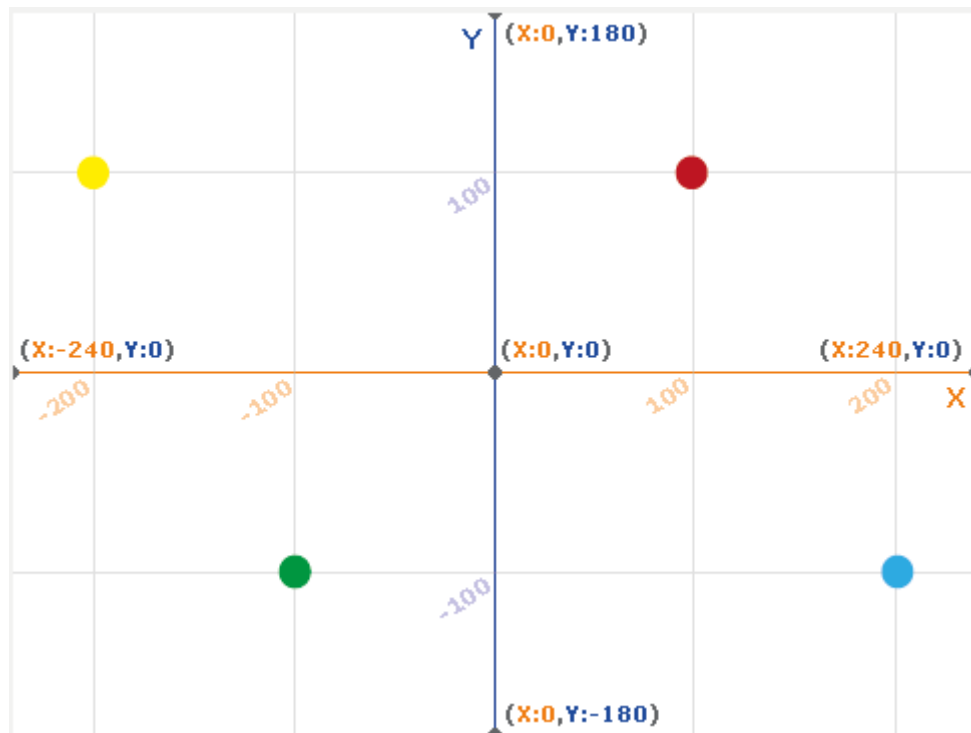


La troisième condition (si température supérieure à 20°C) n'a pas besoin d'être posée : si la température n'est pas inférieure à 10°C, qu'elle n'est pas non plus comprise entre 10°C et 20°C, c'est qu'elle est supérieure à 20°C.

Annexe 2.1



2^{ème} étape : trouve et note la position (les coordonnées) de chaque point.



Point noir	x : 0	y : 0	(0, 0)
Point rouge	x :	y :	(,)
Point jaune	x :	y :	(,)
Point vert	x :	y :	(,)
Point bleu	x :	y :	(,)

3^{ème} étape : positionne des points (sur la grille de la 2^{ème} étape)

Place un point rose à x : - 50 et y : 0 (-50, 0)

Place un point gris à x : 150 et y : - 50 (150, - 50)

Place un point orange à x : 240 et y : - 180 (240, - 180)

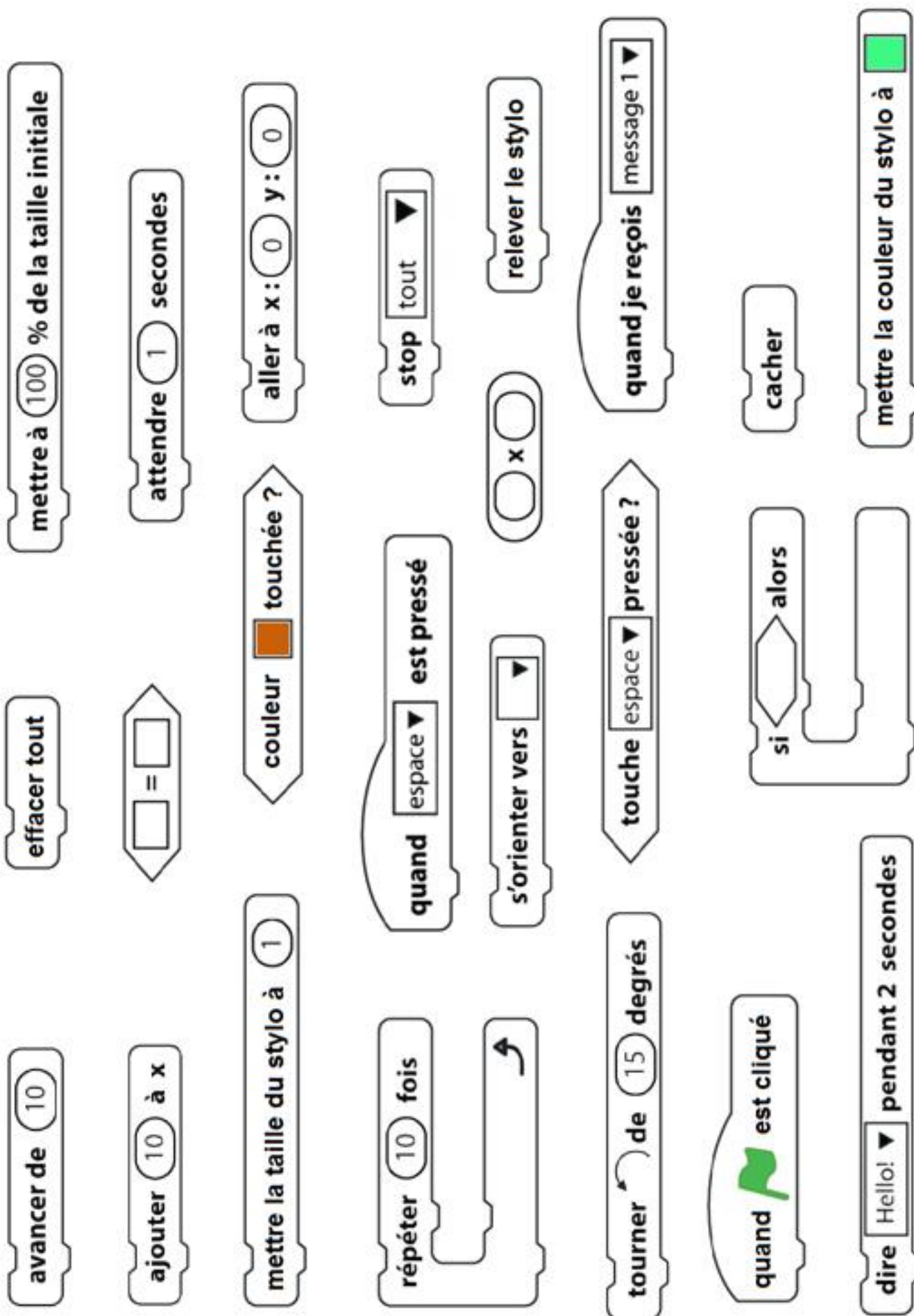
Place un point violet à x : - 300 et y : - 100 (- 300, - 100)

4^{ème} étape : déplace un point (sur la grille de la 2^{ème} étape)

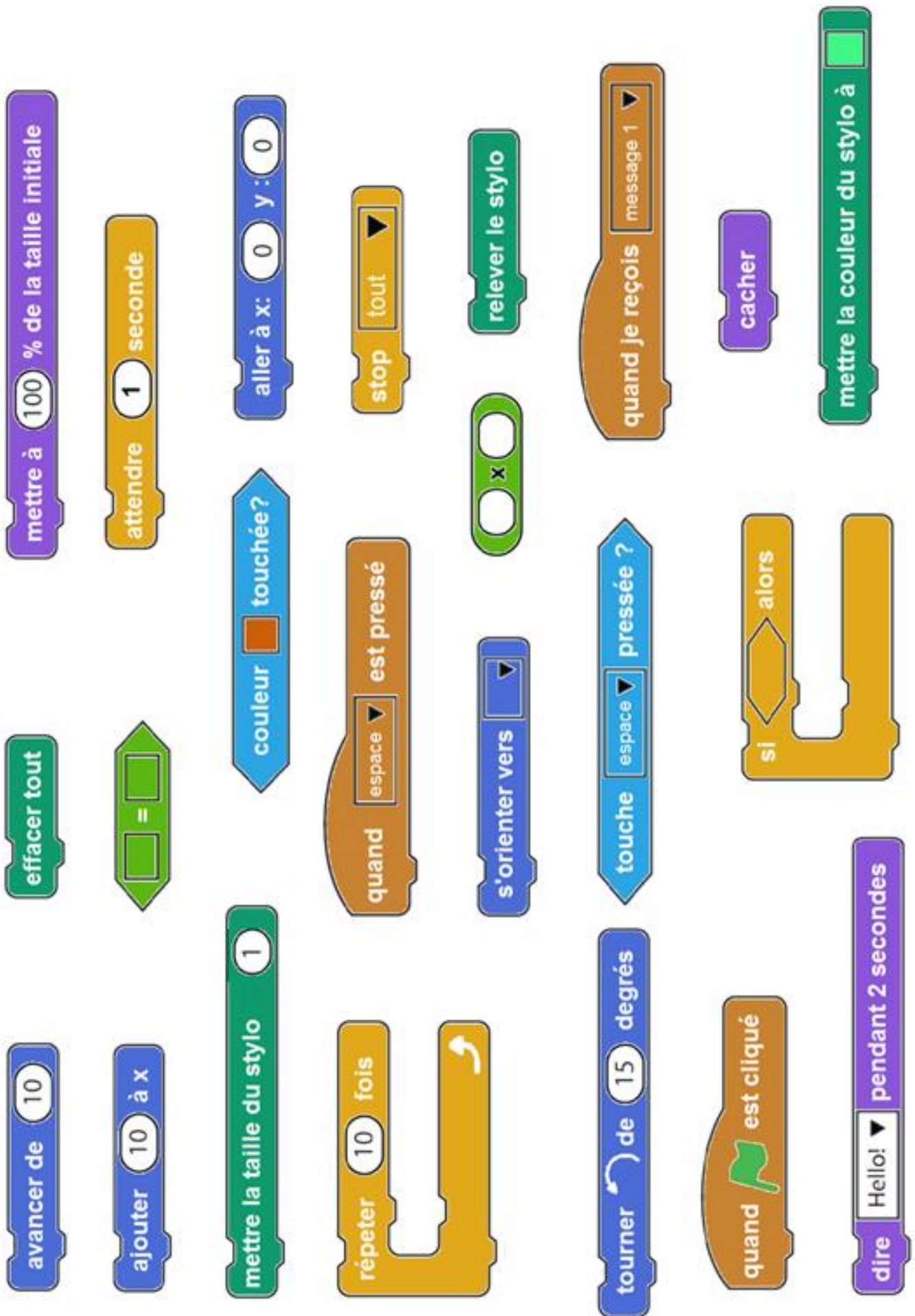
Je veux déplacer le point jaune de 400 en x et de -200 en y.

Où ce point va-t-il se retrouver ?

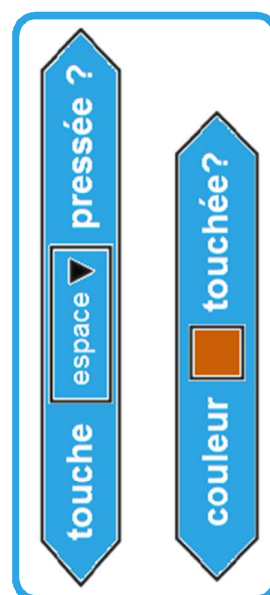
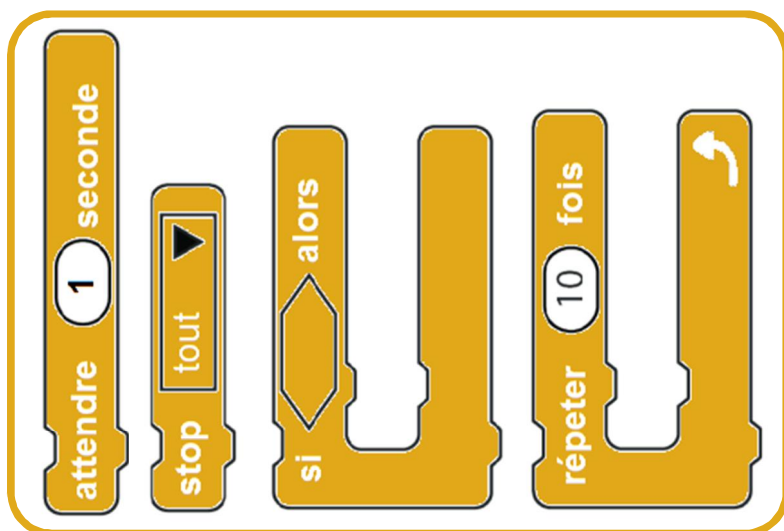
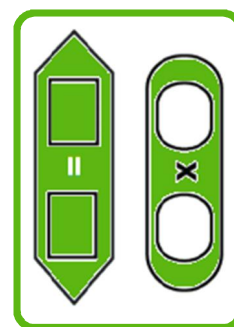
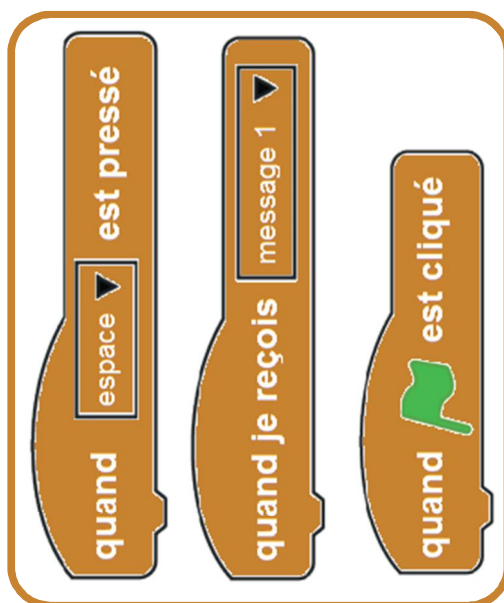
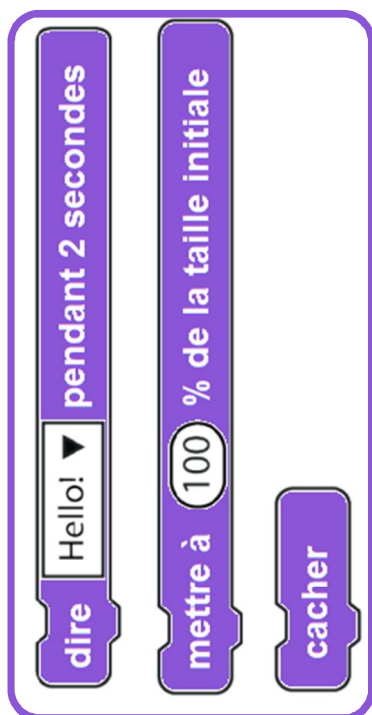
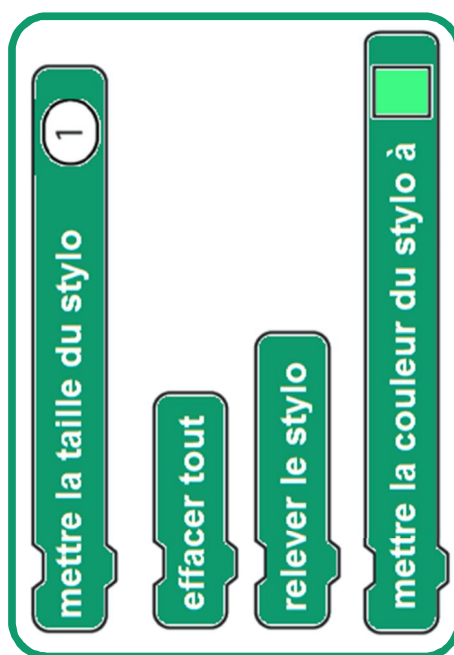
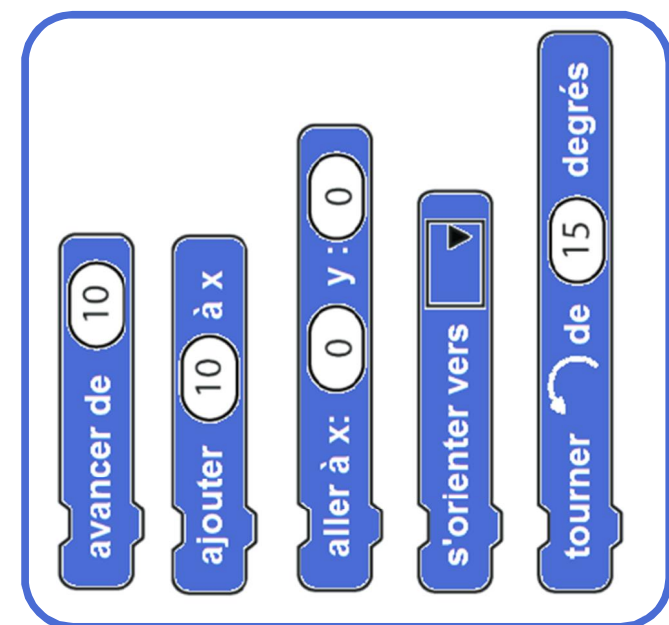
Annexe 3.1 : Blocs d'Instructions de Scratch



Annexe 3.2 : Blocs d'instructions de Scratch en couleurs



Annexe 3.3 : Blocs d'Instructions de Scratch par catégories



avancer de 10

ajouter 10 à x

aller à x: 0 y: 0

s'orienter vers ▼

tourner ↻ de 15 degrés

mettre la taille du stylo à 1

effacer tout

relever le stylo

mettre la couleur du stylo à 

dire Hello! ▼ pendant 2 secondes

mettre à 100 % de la taille initiale

cacher

quand espace ▼ est pressé

quand je reçois message 1 ▼

quand  est cliqué

attendre 1 secondes

stop tout ▼

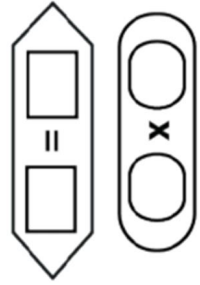
si  alors

répéter 10 fois

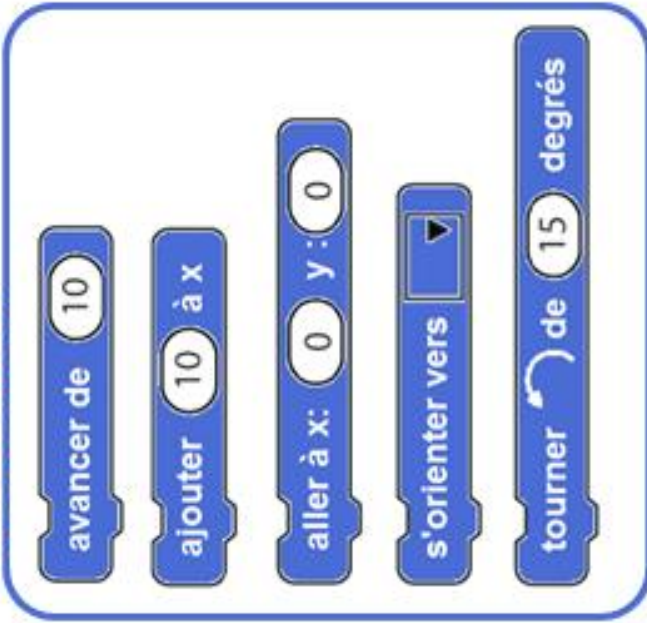


touche espace ▼ pressée ?

couleur  touchée ?



MOUVEMENT



avancer de 10

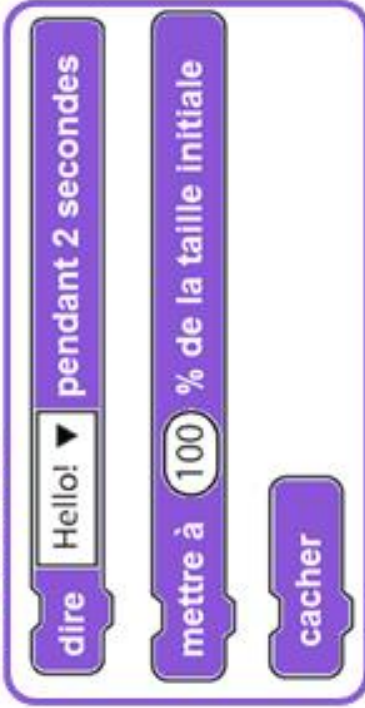
ajouter 10 à x

aller à x: 0 y: 0

s'orienter vers

tourner de 15 degrés

APPARENCE

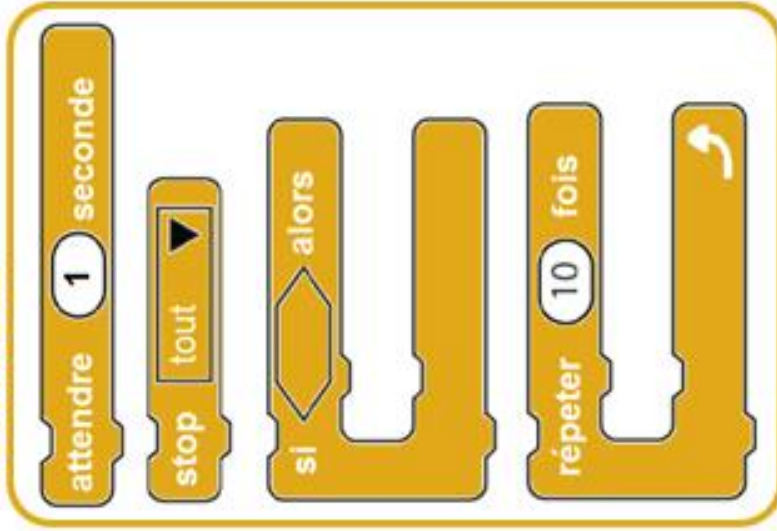


dire Hello! pendant 2 secondes

mettre à 100 % de la taille initiale

cacher

CONTRÔLE



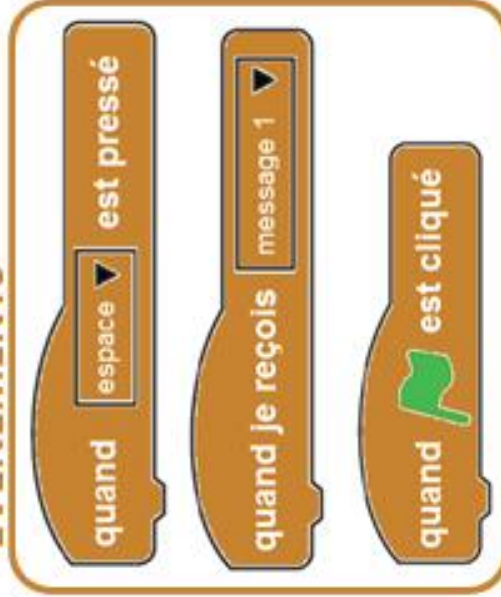
attendre 1 seconde

stop tout

si alors

répéter 10 fois

ÉVÈNEMENTS

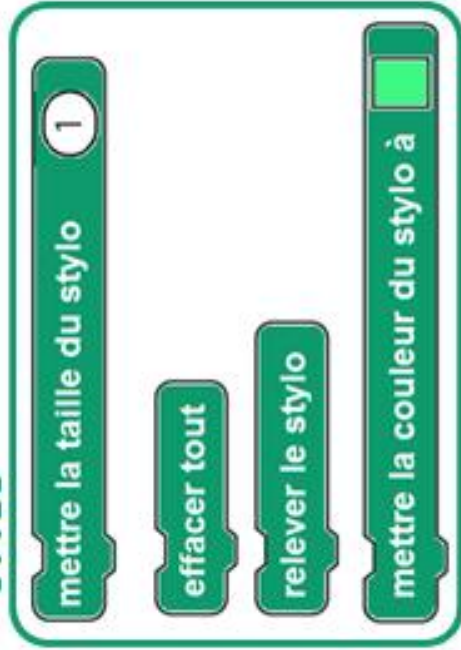


quand espace est pressé

quand je reçois message 1

quand est cliqué

STYLO



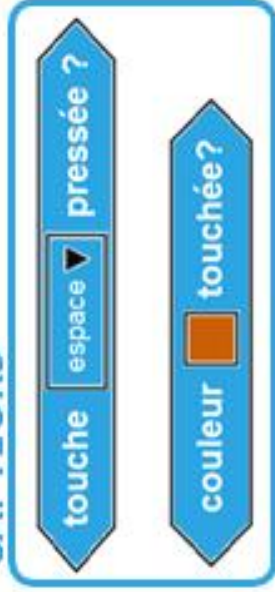
mettre la taille du stylo 1

effacer tout

relever le stylo

mettre la couleur du stylo à

CAPTEURS



touche espace pressée ?

couleur touchée ?

OPÉRATEURS

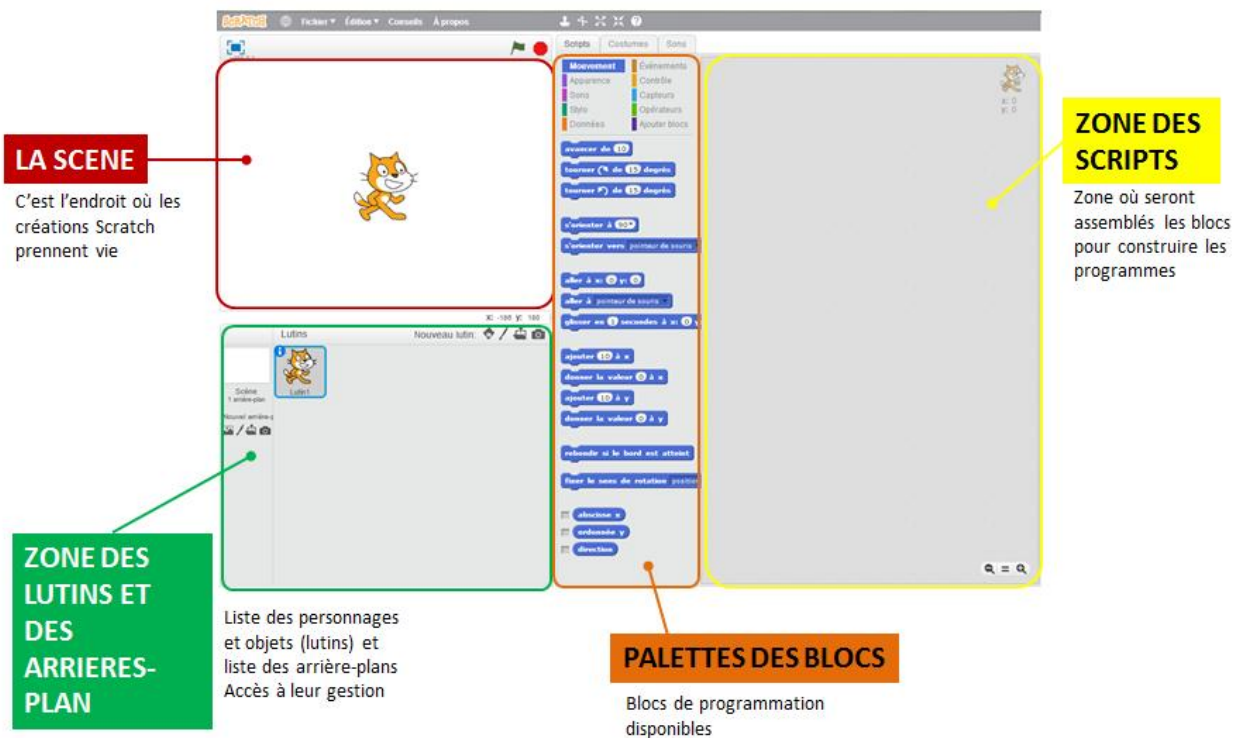


=

x

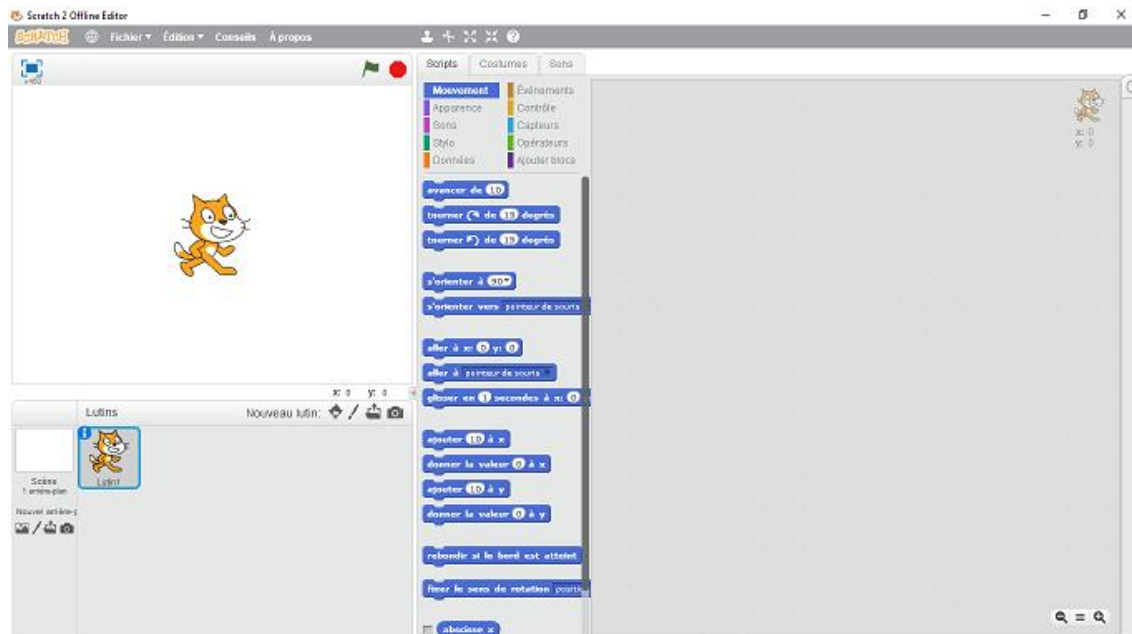
Annexe 3.4 : Interface de Scratch

Utiliser le document powerpoint ou pdf présentant l'interface Scratch
(Annexe_3_4_Interface_Scratch.pptx ou Annexe_3_4_Interface_Scratch.pdf)



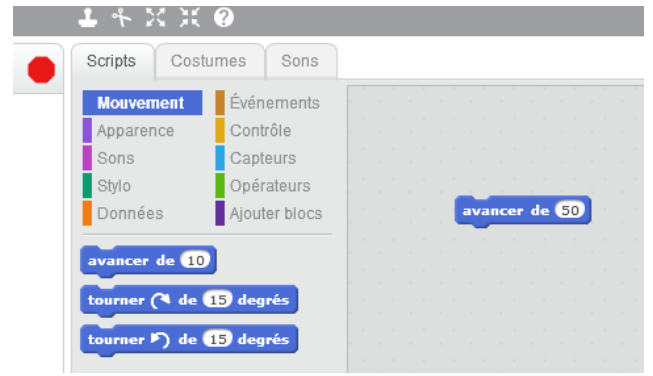
Pour l'enseignant : tous premiers pas avec Scratch




Ouvrir Scratch.




Par défaut, à l'ouverture de Scratch, le lutin est un chat orange et l'arrière-plan est blanc.

Cliquer sur le bloc  et le glisser dans la zone de script.

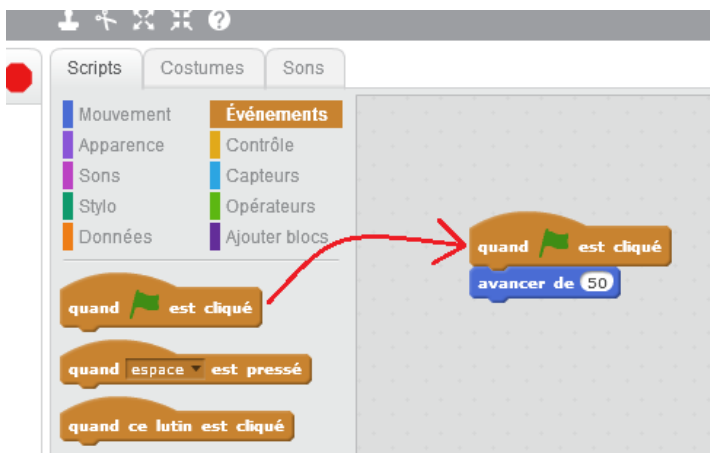


En cliquant sur la zone blanche , on peut modifier le chiffre 10 :  (ou ).

Pour déclencher cette action, cliquer sur le bloc  dans la zone de script. Le lutin avance de 50 pixels (ou recule de 100 pixels si bloc ).

Passer en mode plein écran en cliquant sur . On ne voit que la scène et plus les scripts (assemblages de blocs) à l'écran.


Comment faire pour déclencher les actions ? On voit un drapeau vert et une forme octogonale rouge. Sortir du mode plein écran et ajouter un bloc événement dans le script. Les blocs semblent, comme aimantés.

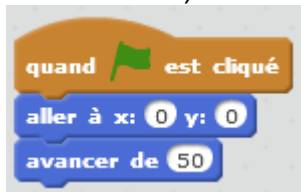


Passer en mode plein écran et cliquer sur le drapeau vert pour déclencher l'action.

Remarque : pour déclencher ce script, on peut également toujours cliquer sur les 2 blocs dans la zone de script quand celle-ci est visible.

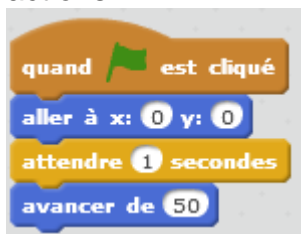
Si on clique 5 fois de suite sur le drapeau vert (ou sur le script quand il est visible), le lutin sort de la scène (il se déplace en tout de 250 pixels vers la droite) il ne revient pas à sa position de départ car cela n'est pas indiqué dans le script !

Pour préciser la position de départ du lutin, on peut utiliser le bloc  (catégorie **Mouvement**).



Mais si on clique plusieurs fois, on ne voit plus le lutin se déplacer.

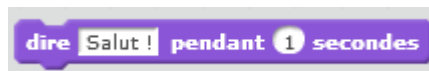
Il faut ajouter un bloc de la catégorie **Contrôle** () pour bien voir toutes les actions.



Faisons faire une autre action au chat : ajoutons un bloc **Apparence** :



que l'on va modifier ainsi



Lors de l'exécution du script, le chat part du centre de la scène, attend 1 seconde, avance de 50 pixels puis dit (bulle de parole) « Salut ! » pendant 1 seconde.

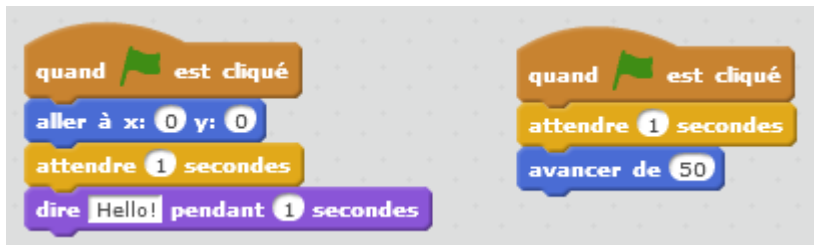
Les actions se réalisent dans l'ordre où elles ont été empilées.

Si on veut que le chat salue puis se déplace, alors il faut inverser l'ordre des blocs.



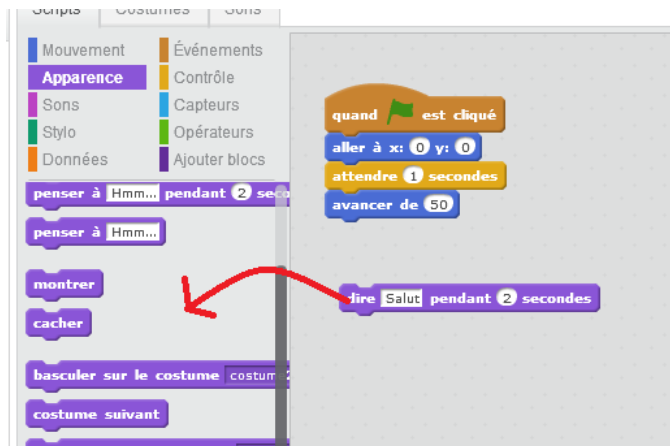
Et si on veut que le chat salue en même temps qu'il se déplace ?

Il faut faire 2 scripts, déclenchés par le même événement (ici appui sur drapeau vert) ; les deux actions seront alors simultanées.

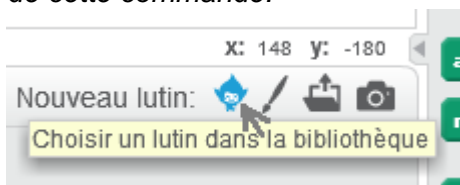


Remarques :

Pour supprimer un ou plusieurs blocs de la zone de script, il suffit de les glisser vers la zone centrale des blocs.



Quand on passe le pointeur de la souris sur les commandes, on voit une étiquette expliquant la fonction de cette commande.



Il existe de nombreux documents et tutoriels pour débiter avec Scratch. Ne pas hésiter à se reporter.

<https://magicmakers.fr/scratch-2-scratch-online-tutoriels>

https://scratch.mit.edu/projects/editor/?tip_bar=getStarted

[http://dsden02.ac-](http://dsden02.ac-amiens.fr/tnp/tnp_internet/Dossiers_tnp/activites_classe_mobile/programmation/utiliser_scratch_a_l_ecole.pdf)

[amiens.fr/tnp/tnp_internet/Dossiers_tnp/activites_classe_mobile/programmation/utiliser_scratch_a_l_ecole.pdf](http://dsden02.ac-amiens.fr/tnp/tnp_internet/Dossiers_tnp/activites_classe_mobile/programmation/utiliser_scratch_a_l_ecole.pdf)

et bien d'autres

La prise en main de Scratch par les élèves est en général rapide.

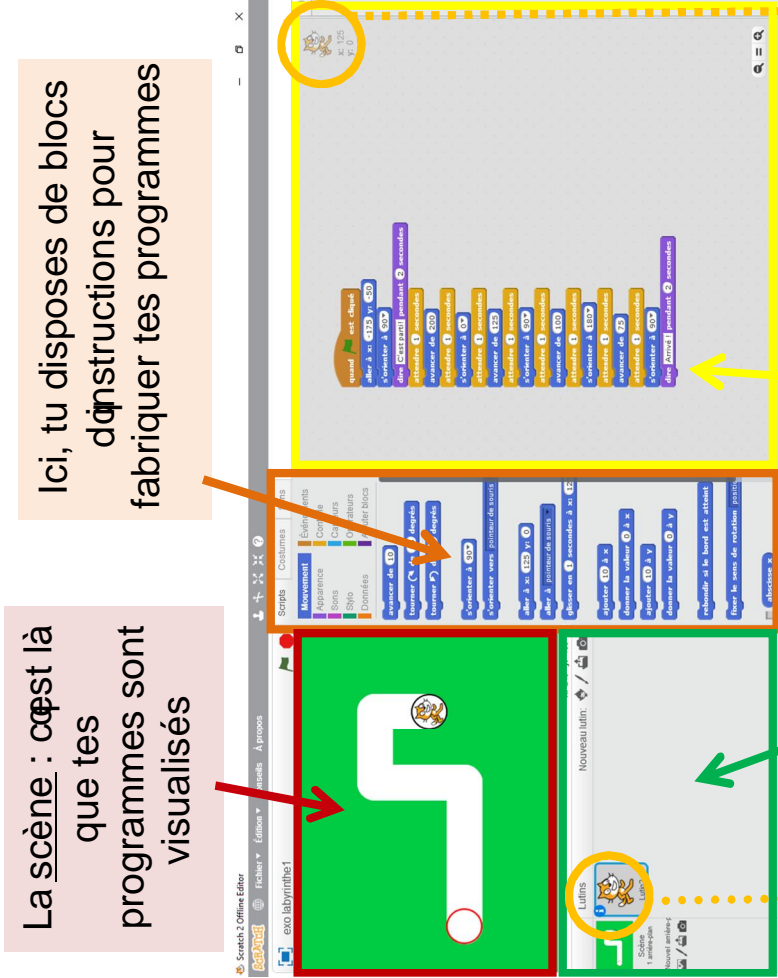
Il n'y a pas de syntaxe à connaître ou à écrire, on déplace et empile des blocs.

L'environnement est simple et efficace : on a à la fois à l'écran les blocs d'instructions, la fenêtre de programmation (zone des scripts) et la fenêtre d'exécution des programmes (scène).

Il est adapté à la programmation événementielle (les scripts sont déclenchés par un événement).

Un simple clic sur un bloc ou un groupe de blocs conduit à leur exécution, ce qui permet de vérifier facilement la bonne programmation d'un personnage ou d'un objet.

Aide-mémoire Scratch : les zones



La scène : c'est là que tes programmes sont visualisés

Ici, tu disposes de blocs d'instructions pour fabriquer tes programmes

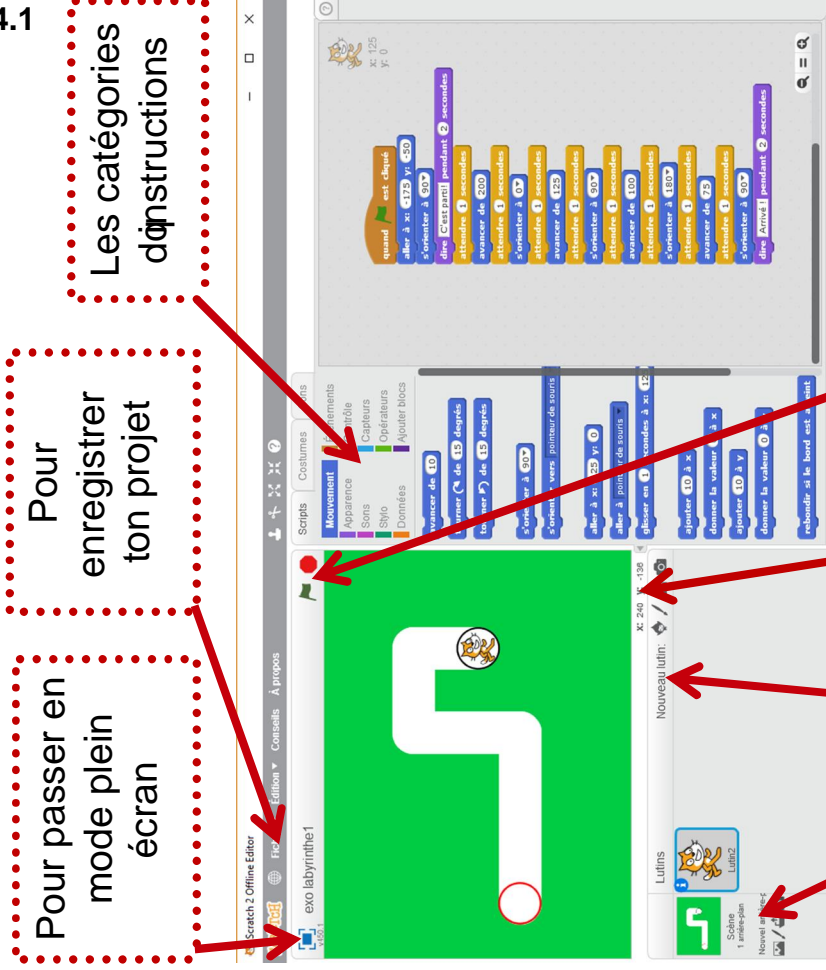
C'est ici qu'apparaissent tes personnages et tes arrière-plans

C'est ici que tu emboites des instructions pour concevoir tes programmes

L'objet pour lequel tu es en train d'écrire un programme

Annexe 4.1

Aide-mémoire Scratch : les commandes



Pour passer en mode plein écran

Pour enregistrer ton projet

Les catégories d'instructions

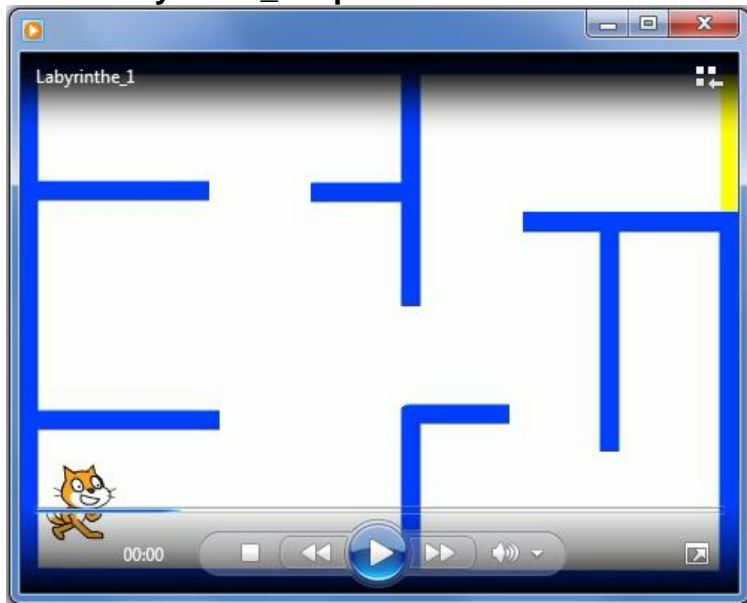
Pour créer des nouveaux personnages et arrière-plans

Pour commencer et arrêter les programmes

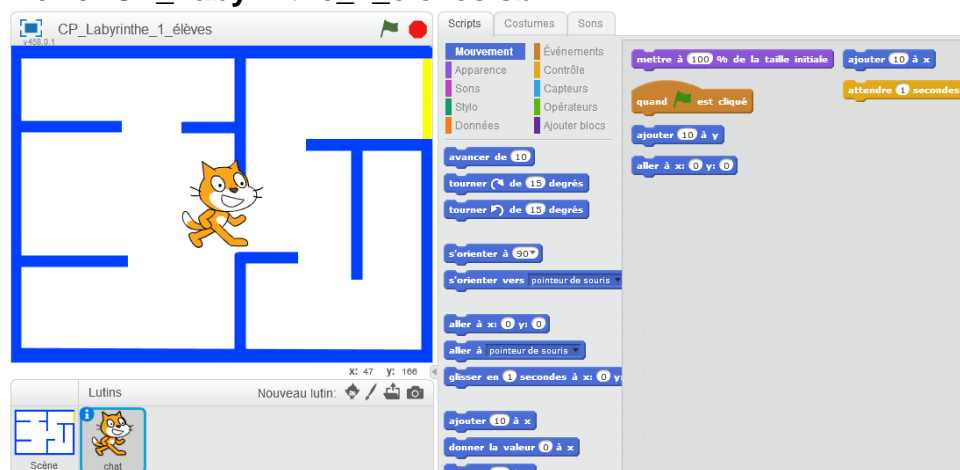
Position du pointeur de la souris

Annexe 4.2

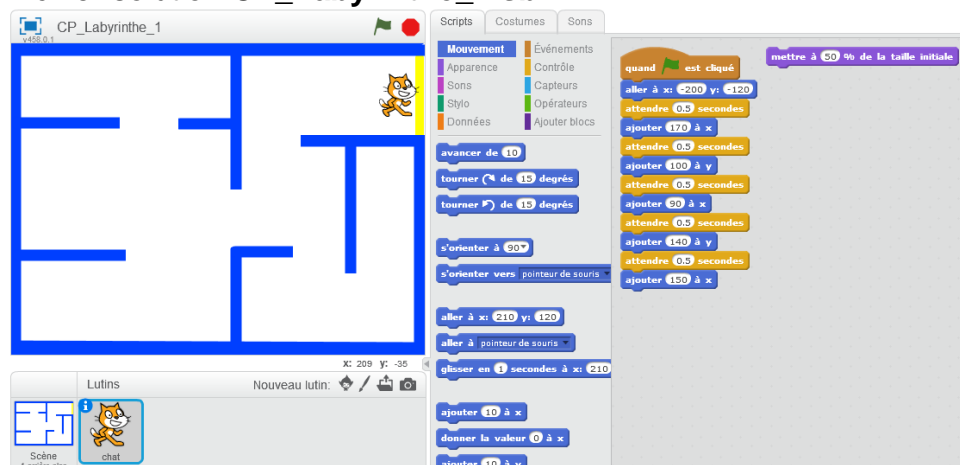
Vidéo Labyrinthe_1.mp4



Fichier CP_Labyrinthe_1_eleves.sb2

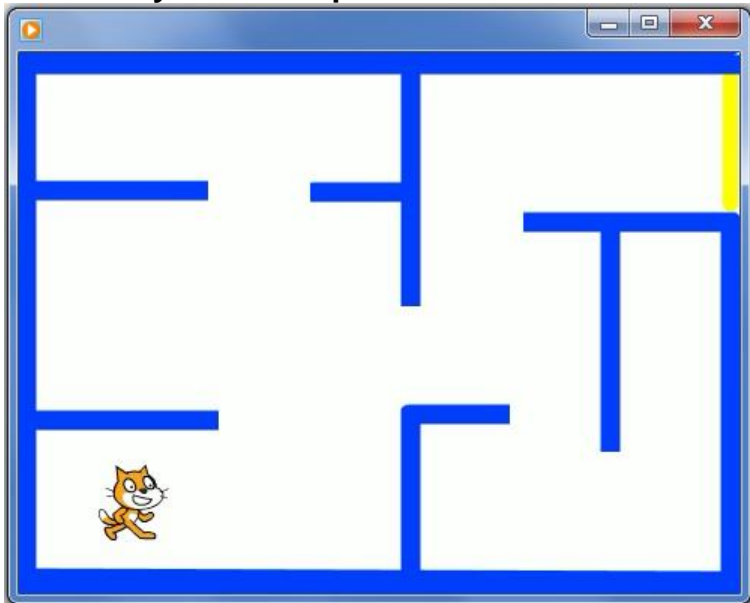


Fichier solution CP_Labyrinthe_1.sb2

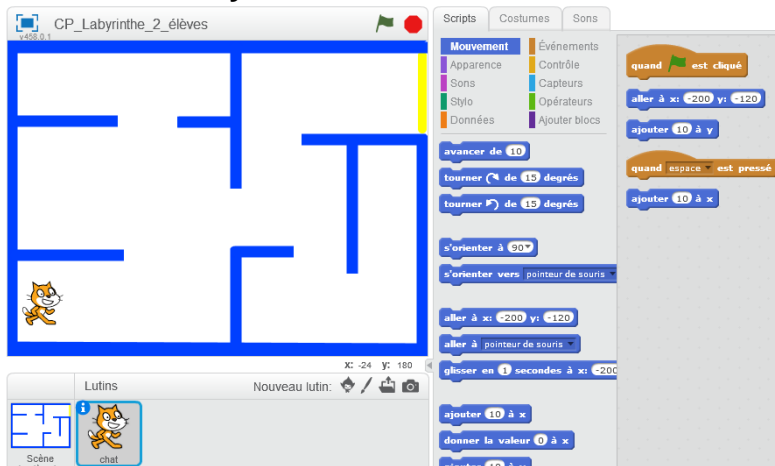


Annexe 5.1

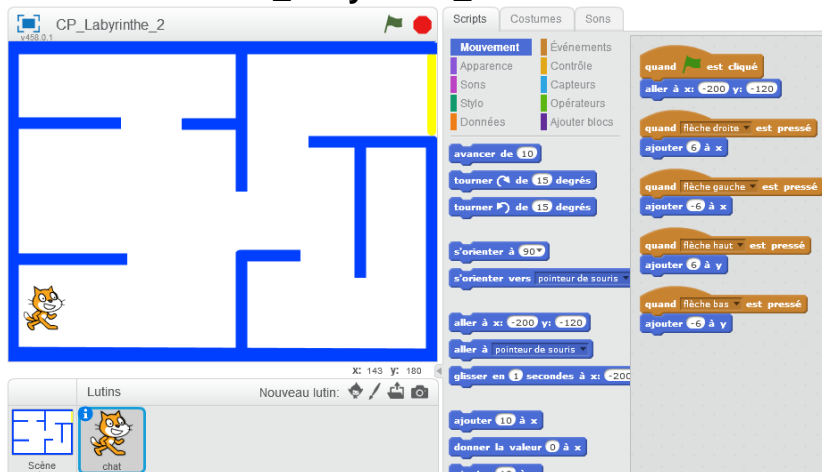
Vidéo Labyrinthe_2.mp4



Fichier CP_Labyrinthe_2_eleves.sb2



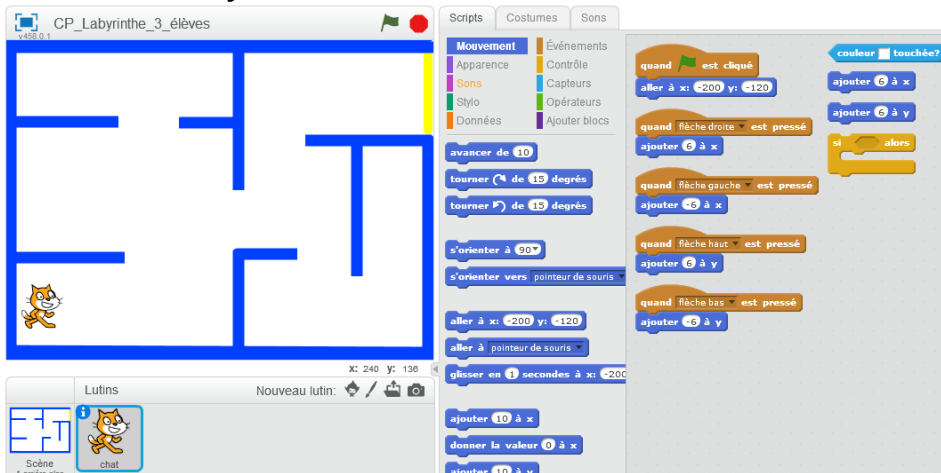
Fichier solution CP_Labyrinthe_2.sb2



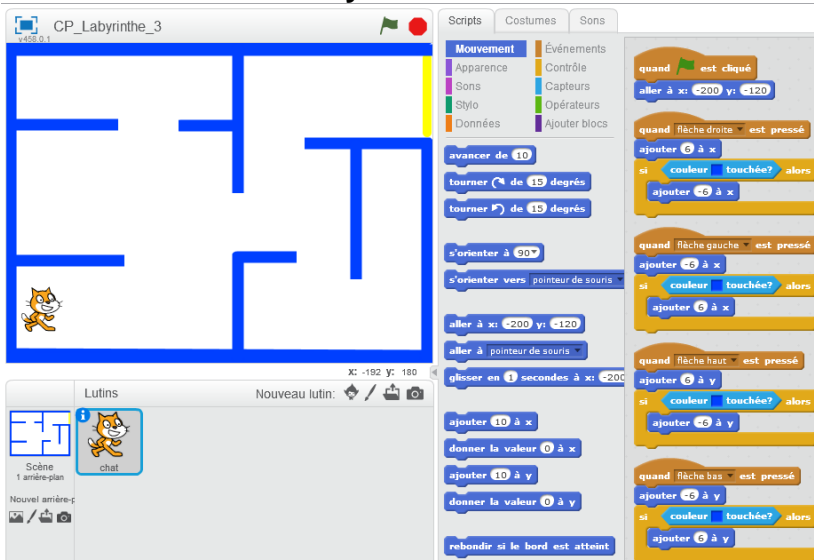
Vidéo Labyrinthe_3.mp4



Fichier CP_Labyrinthe_3_eleves.sb2

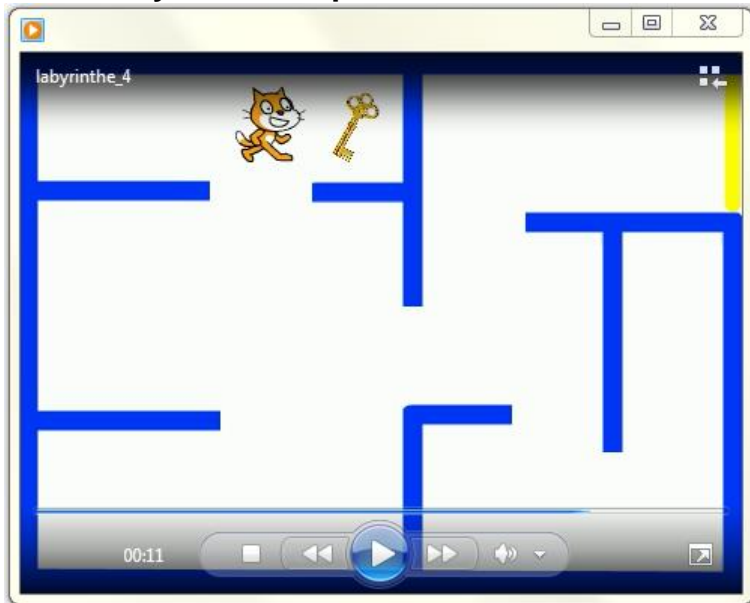


Fichier solution CP_Labyrinthe_3.sb2

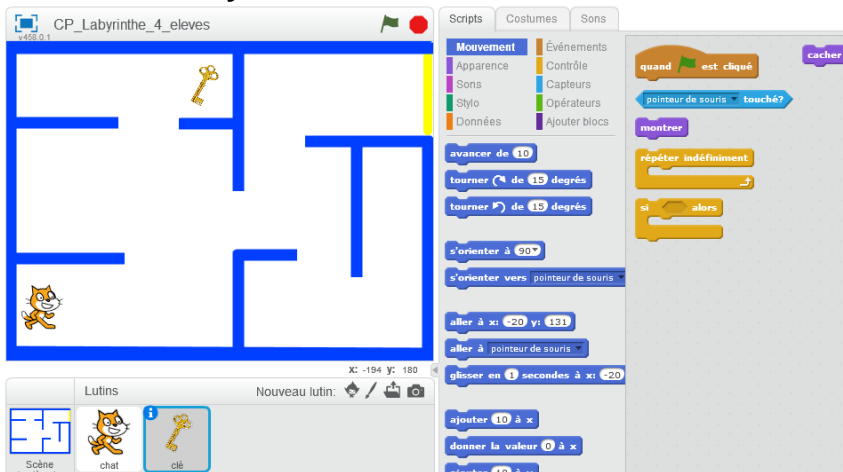


Annexe 6.1

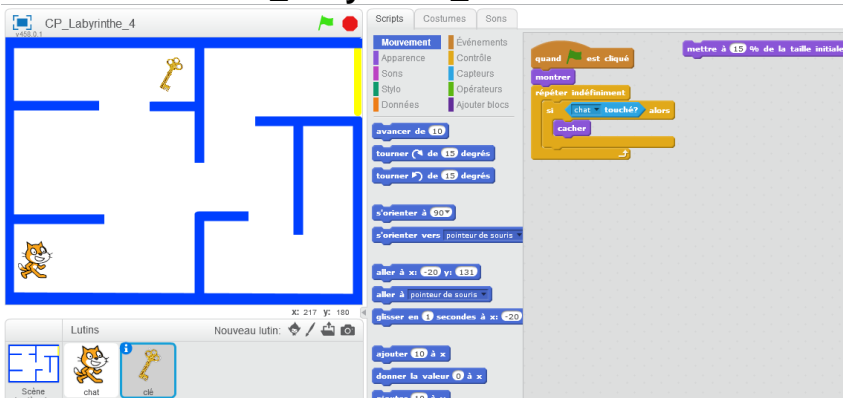
Vidéo Labyrinthe_4.mp4



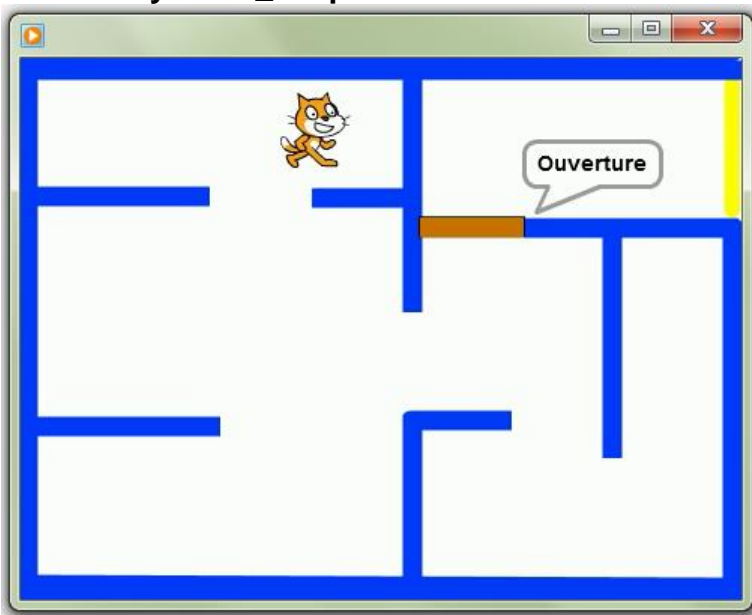
Fichier CP_Labyrinthe_4_eleves.sb2



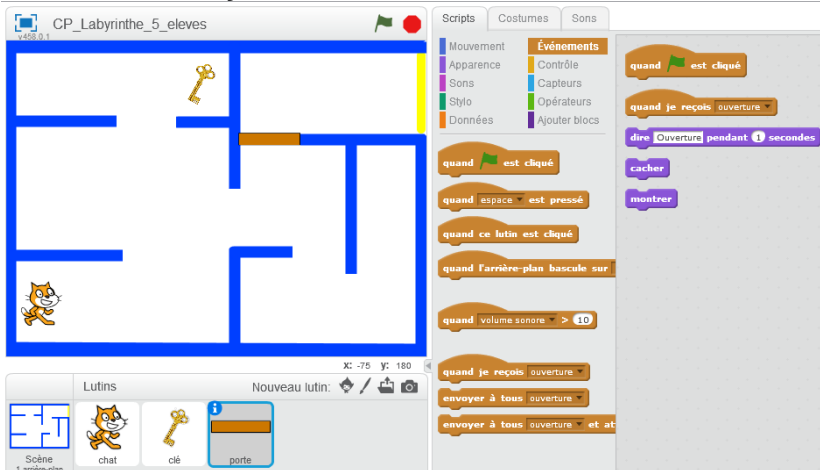
Fichier solution CP_Labyrinthe_4.sb2



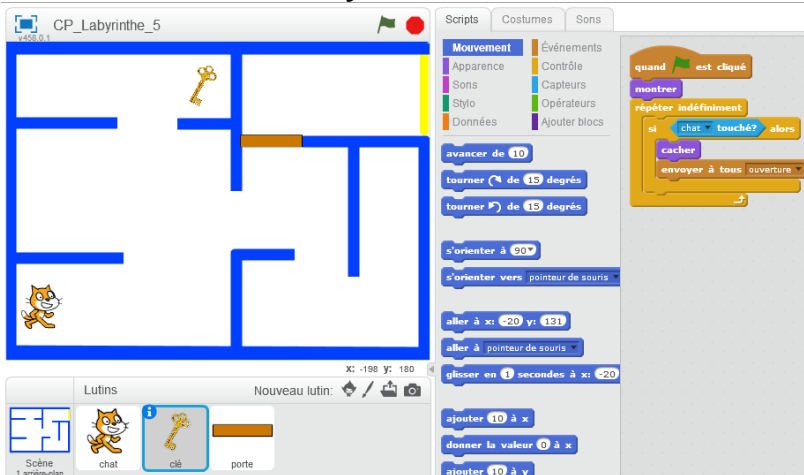
Vidéo Labyrinthe_5.mp4



Fichier CP_Labyrinthe_5_eleves.sb2

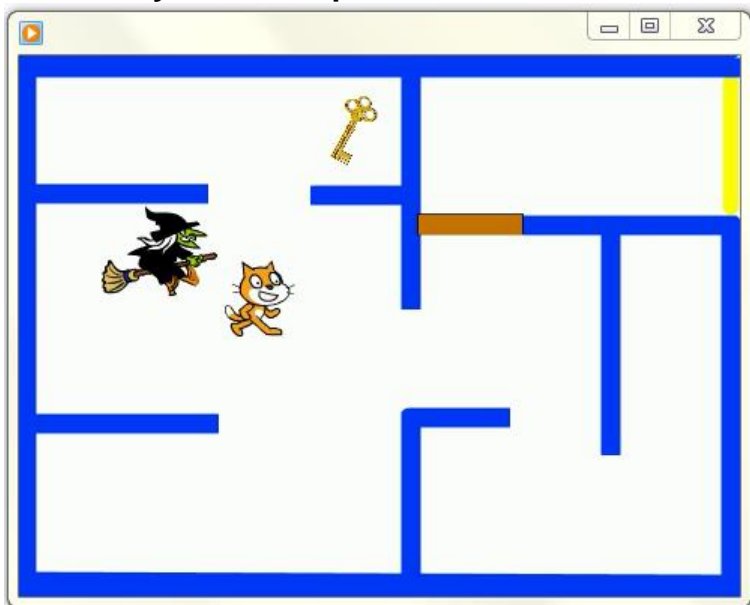


Fichier solution CP_Labyrinthe_5.sb2

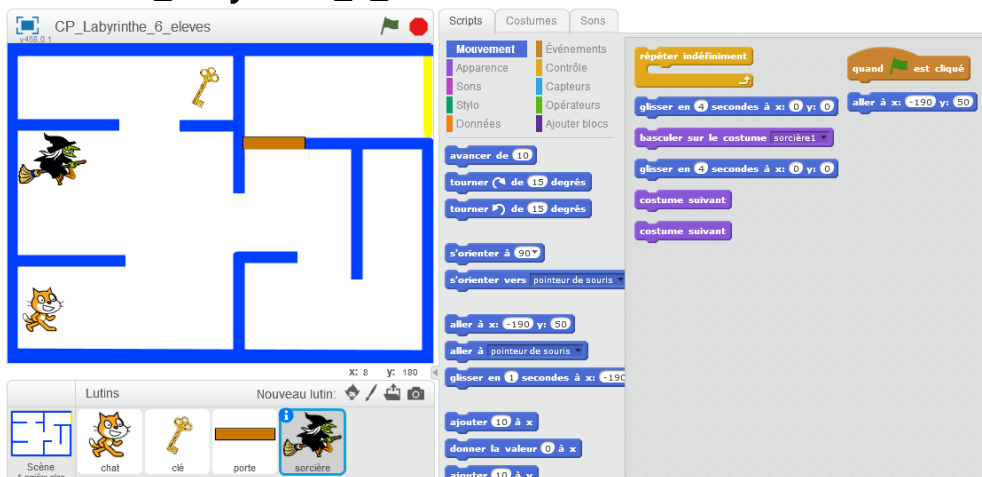


Annexe 7.1

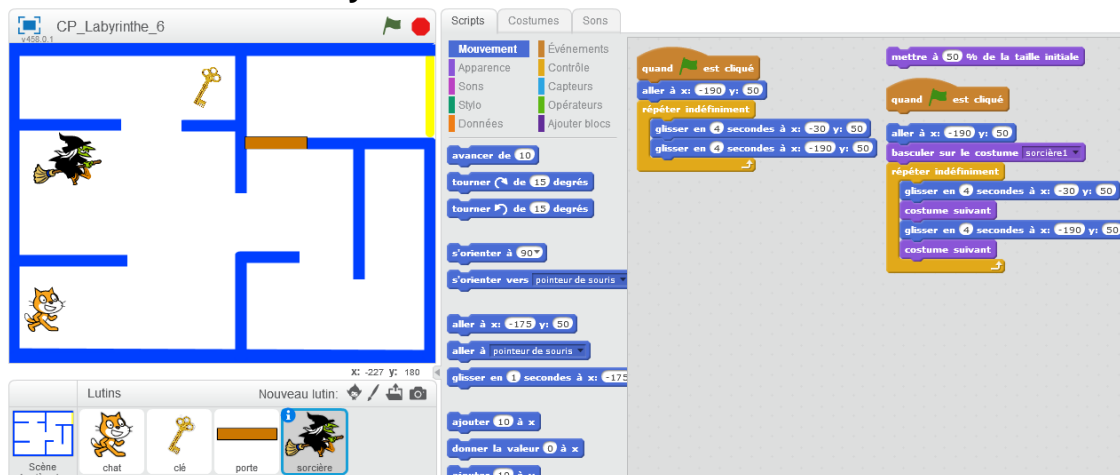
Vidéo Labyrinthe_6.mp4



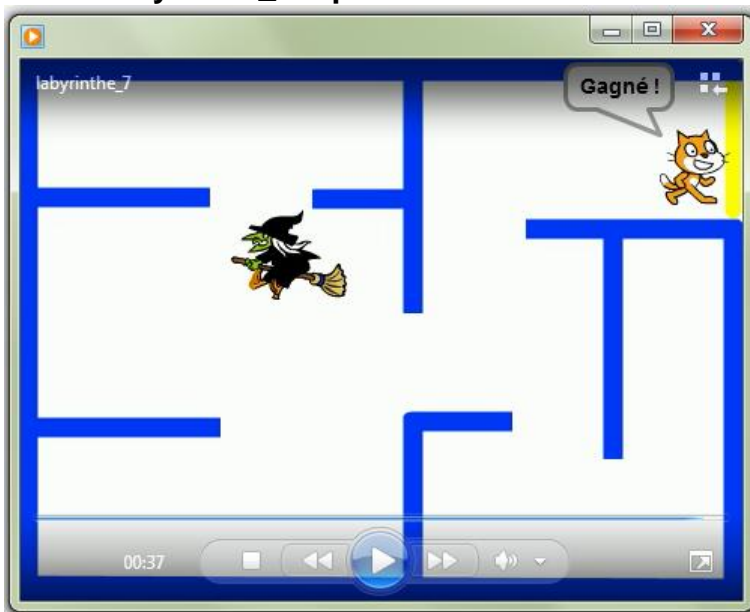
Fichier CP_Labyrinthe_6_eleves.sb2



Fichier solution CP_Labyrinthe_6.sb2



Vidéo Labyrinthe_7.mp4



Fichier CP_Labyrinthe_7_eleves.sb2

Fichier solution CP_Labyrinthe_7.sb2



Ressources

Ressources Scratch explorateur : <https://primabord.eduscol.education.fr/scratch-explorateur>

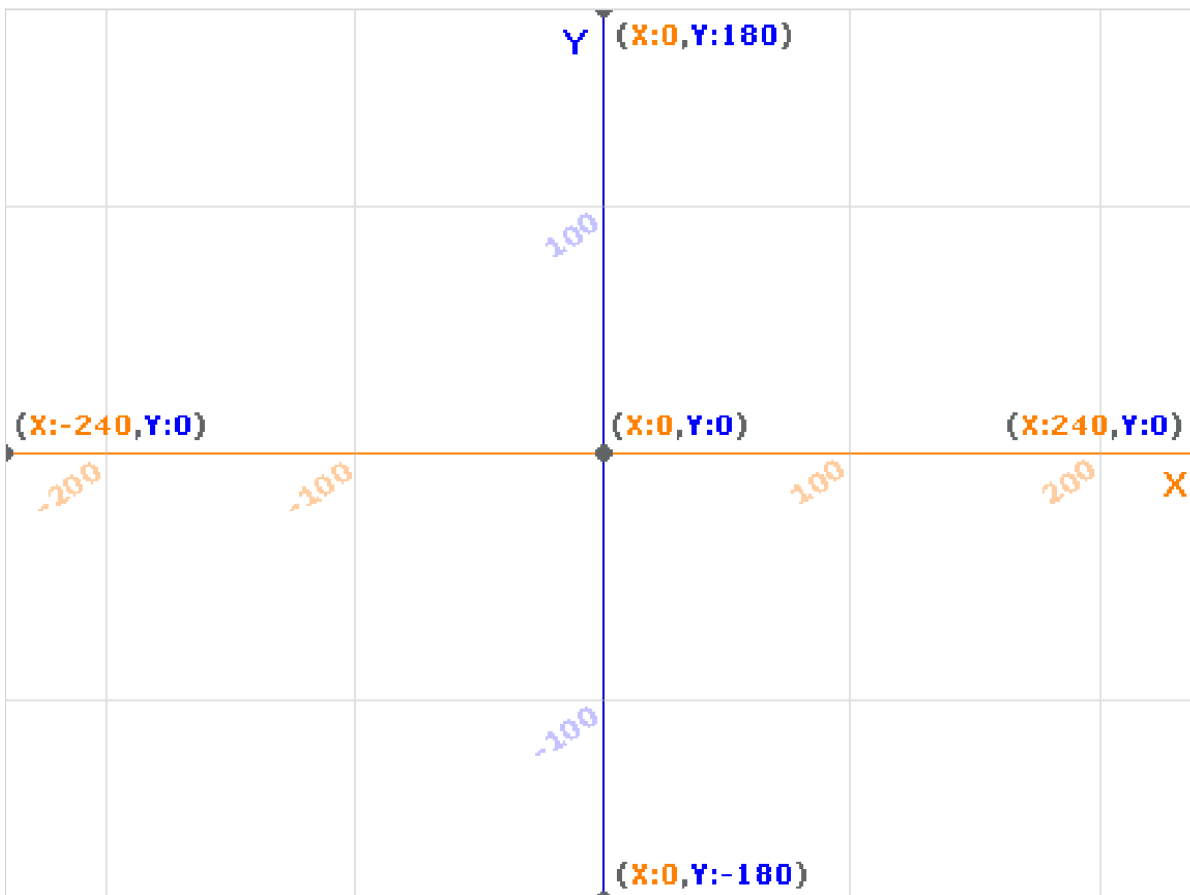
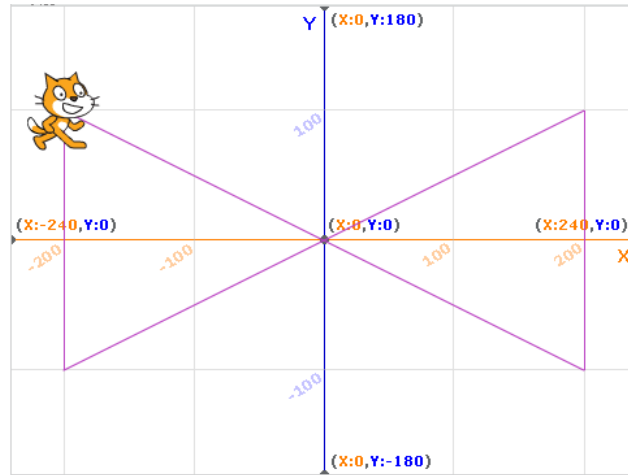
Réseau Canopé, cartes devinettes : <http://www.reseau-canope.fr/atelier-yvelines/spip.php?article1158>

Des exercices à faire en classe au retour du Centre pilote

Quelle figure est obtenue suite à l'exécution de ce programme ?

```
quand  est cliqué  
aller à x: -200 y: 100  
effacer tout  
mettre la couleur du stylo à   
stylo en position d'écriture  
attendre 1 secondes  
aller à x: 200 y: -100  
attendre 1 secondes  
ajouter 200 à y  
attendre 1 secondes  
aller à x: -200 y: -100  
attendre 1 secondes  
donner la valeur 100 à y  
attendre 1 secondes
```

Solution



Amuse-toi à imaginer des figures et à écrire le script pour les tracer dans Scratch.

Quel tracé est obtenu suite à l'exécution de ce programme ?



Au lancement de ce script, que va faire la lionne ?

