Cahier D'activités Scratch

Enseignant :
Mahjoubi BILEL

Niveau 8ème année de base

Année scolaire 2019-2020

Nom : .....
Prenom : .....
Classe : .....
Groupe : .....

J'approfondis mes compétences

- J'analyse le code d'un programme simple.
- Je comprends et j'utilise une simulation informatique.
- Je teste ma solution et la corrige si necessaire.

Objectifs

- Notion d'algorithme, de programme, de variable informatique.
- Tester et corriger un programme.
- Écrire un programme dans lequel des actions sont déclenchées par des évènements extérieurs.

SÉQUENCES

1. Instructions et algorithme
2. Utilisation des boucles
3. Utilisation des variables
4. Utilisation des instructions conditionnelles
5. Utilisation d'un bloc d'instructions

PROJETS

1. Le chiffre de César
2. Le crabe aux pinces magiques
3. Jeu de Nim
4. Le mage et la grenouille
5. Le nombre mystère

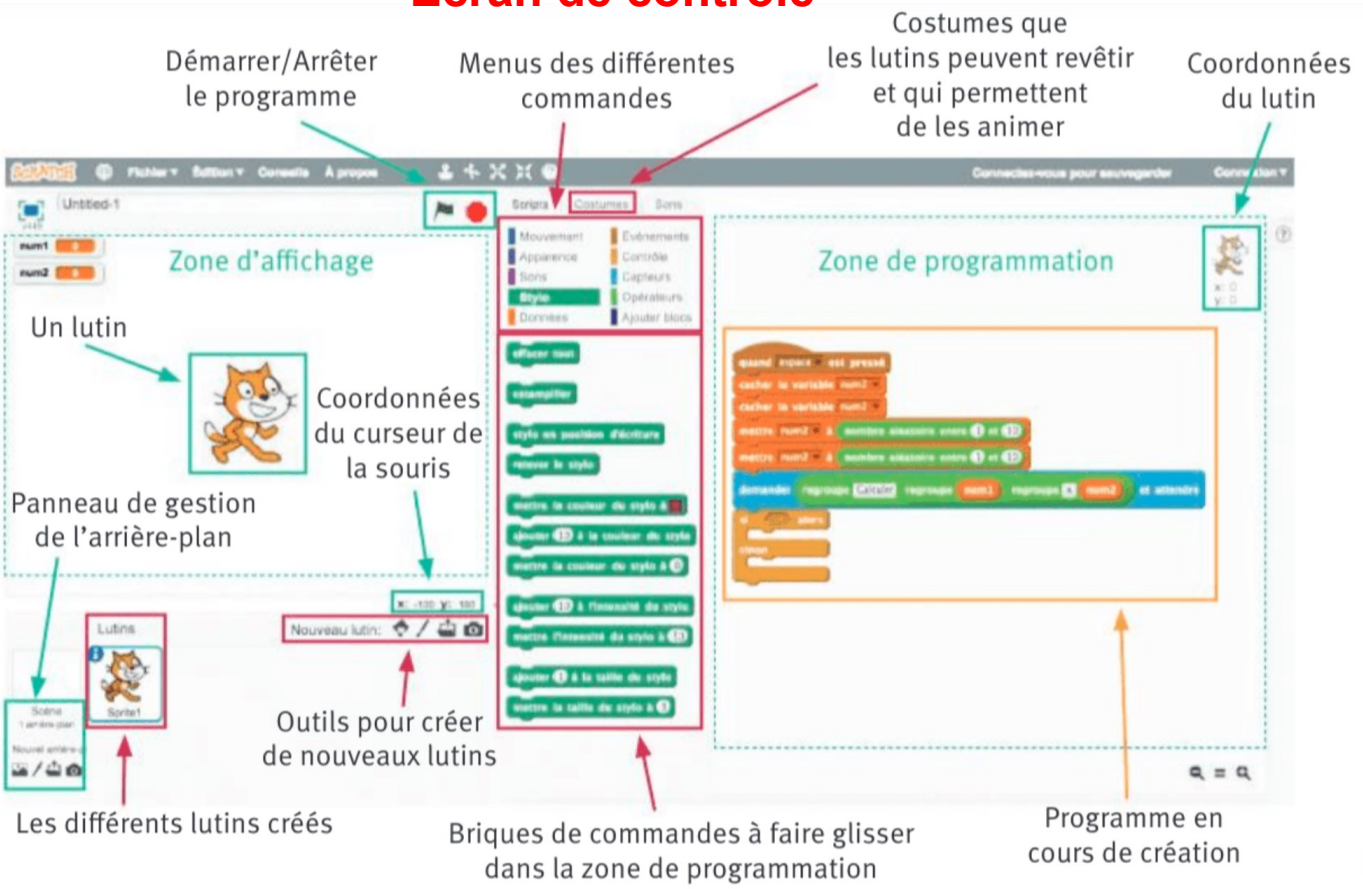
Découvrir Scratch

Je découvre

Dans une histoire, on décrit la scène et les personnages avant de rentrer dans l'action.
Avec Scratch, c'est pareil. Pour chaque **projet** (jeu, animation…) :
– on choisit la **scène** initialement « vide » ; on peut la modifier avec le bouton .
– on ajoute des **lutins** ; ce sont des personnages, des animaux ou des objets.
Chaque nouvelle scène contient le lutin Sprite (chat mascotte de Scratch) que l'on peut supprimer.

Voici **l'interface de Scratch**. On peut y accéder en ligne ou installer l'application.

Écran de contrôle



Menus des différentes commandes

Principales instructions dans Scratch

Mouvement

Ce menu sert principalement à déplacer les lutins dans la scène et à leur donner des directions de déplacement.

Apparence

Ce menu permet de changer l'apparence des lutins et surtout de leur faire dire des choses ce qui permet de communiquer les résultats d'un calcul par exemple.

Sons

Ce menu permet de faire jouer des sons courts ou une musique de fond et de gérer ces sons.

Stylos

Ce menu sert à laisser une trace, ou non, lors des déplacements des lutins ce qui permet par exemple de tracer des figures.

Données

Ce menu permet de définir des variables et ensuite d'opérer sur celles-ci en ajoutant ou affectant des valeurs.

Événements

Ce menu permet de définir l'événement qui déclenchera le programme : un appui sur le drapeau vert ou sur une touche du clavier par exemple.

Contrôle

C'est dans ce menu que se trouvent les différentes boucles et les instructions conditionnelles.

Capteurs

Par ce menu, on aura la possibilité de faire poser une question aux lutins et de mémoriser temporairement la réponse.
On trouvera également différents tests à insérer dans les boucles ou instructions conditionnelles.

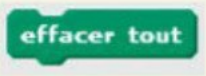
Opérateurs

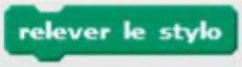
Ce menu permet d'effectuer des calculs, de regrouper des chaines de caractères et de créer des tests à insérer dans les boucles ou instructions conditionnelles.

Trucs et astuces

Le lutin (*sprite* en anglais) prend beaucoup de place à l'écran.
On peut réduire sa taille :

Le stylo





– Il est nécessaire de demander d'effacer ce qui a été tracé, ce n'est pas fait automatiquement d'une utilisation à l'autre.

– Le crayon doit être mis en position d'écriture si on souhaite écrire.

– Et bien sûr, il ne faut pas oublier de le relever quand on veut déplacer un lutin sans qu'il ne laisse de traces.

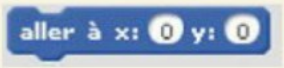
Les événements

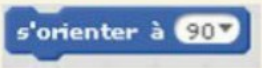
– Cet élément permet de démarrer les actions pour chaque programme.

– Il suffit ensuite de cliquer sur le drapeau vert en haut à droite de la zone d'affichage pour lancer le programme et sur le bouton rouge pour le stopper.

– Il y a d'autres façons de démarrer un programme, en appuyant sur une touche par exemple.

Les mouvements

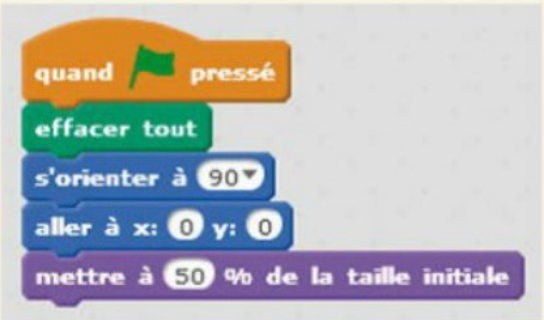




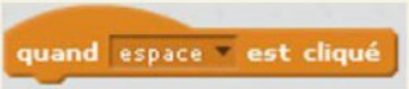
– Cet élément permet de démarrer à l'endroit que l'on souhaite dans la zone d'affichage.
Ici, on débute au centre de la zone qui va de –240 à +240 pour x et de –180 à +180 pour y.

– On peut orienter les déplacements du lutin.
Dans cette situation, il démarre verticalement tourné vers la droite.

Un programme pratique pour réinitialiser l'affichage



Cet algorithme peut démarrer aussi avec :



On remet **le lutin** en situation initiale dès que l'on appuie sur le drapeau.

Instructions et algorithme

Je découvre

• *Les algorithmes*

Dans la vie, on réagit tous les jours à des événements. Mais chacun y réagit différemment. Par exemple ; certains se bouchent les oreilles quand on les appelle, d'autres répondent.

Un **algorithme** décrit, plus ou moins précisément, les **actions** qui s'exécutent à la suite d'un **événement**.

Voici l'algorithme de Zara quand on l'appelle :

Événement → *Quand on m'appelle :*
1re action → *Je réponds.*
2e action → *Je me déplace.*

• *Les scripts*

Le **codage** ou traduction d'un algorithme en langage informatique s'appelle un **script**.

Pour créer le script en **langage Scratch**, qui **code** la réaction de Zara quand on l'appelle, il suffit d'emboîter trois **briques**.

À toi de jouer !

Exercice 1 — Premiers algorithmes

1. Complète cet algorithme avec deux actions logiques :

Quand mon réveil sonne :
.....
.....

2. Complète cet algorithme avec un événement et une action logiques :

Quand ..... :
Je décroche.
.....

Exercice 2

1. Complète ce script pour faire reculer Sprite de 10 pixels si la flèche gauche ← est appuyée :

2. Comment faire reculer Sprite et Abby en même temps de 10 pixels avec le script ci-dessus ?

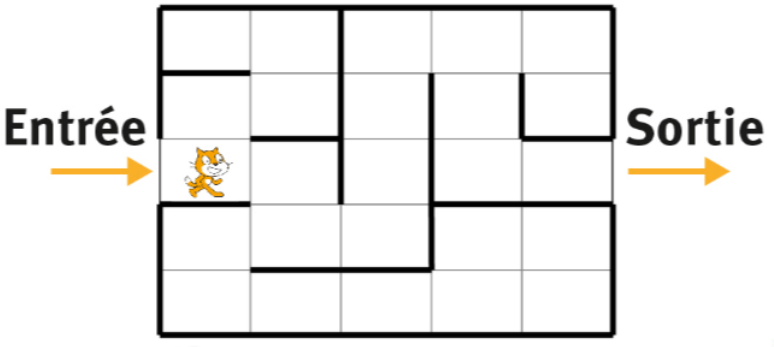
.....
.....
.....

J'applique : Séquences d'instructions

Labyrinthe.

Créez un algorithme qui permet de traverser le labyrinthe à partir des consignes :

- « Avancer d'une case » ;
- « Tourner à gauche » ;
- « Tourner à droite ».



Pour s'entraîner

Instructions et algorithmes

Une **instruction** est une action à exécuter.

> **Exemple :** Sur certaines consoles de jeux, en appuyant sur →, le personnage va aller à droite. Il s'agit d'une instruction.

Une **séquence d'instructions** est une suite d'instructions dans un ordre précis.

> **Exemple :** On peut faire → puis ↑ pour aller à droite puis en haut. L'ordre ici est très important.

Un **algorithme** est une séquence d'instructions finie qui résout un problème.

Projet 1 fin Promenade

1. Ouvre ton projet Promenade.

Utilise le menu Fichier pour ouvrir le document enregistré.

• Déplacement de Sprite

2. a. Place-toi sur Sprite pour créer ce script. Teste-le en appuyant sur la flèche droite ➡.

b. Modifie-le ainsi puis teste-le.

c. Que se passe-t-il de surprenant ?

3. Où ajouter l'action 

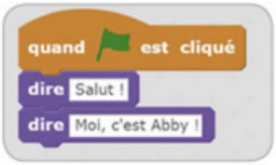
pour un déplacement pas à pas ?

4. Que devient l'abscisse *x* du lutin si tu appuies sur ➡ ?

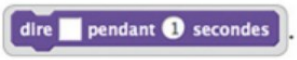
Les coordonnées *x* et *y* de la souris s'affichent en bas à droite de la scène.

• Réaction d'Abby

5. a. Crée et teste ce script d'Abby :



b. Que se passe-t-il d'étrange ?

c. Corrige ce problème avec 

.

6. Quelle action ajouter initialement pour qu'Abby salue une seconde après Sprite ?

7. Teste et enregistre ton projet.

Utilisation des boucles

Dans un algorithme, une **boucle** consiste à faire répéter un certain nombre de fois (connu à l'avance ou non) une même séquence d'instructions. Il existe principalement deux types de boucles : la boucle « Répéter *x* fois » et la boucle « Répéter jusqu'à »

Exemple : Pour aider Teddy à rejoindre le poisson, on peut donner comme instructions ▶ ▶ ▶ ▶.

On peut aussi utiliser des boucles :

« Répéter 4 fois ▶ »

On utilise la boucle « Répéter *x* fois » quand on sait déjà combien de fois on doit faire répéter les instructions.

« Répéter ▶ jusqu'à *Le poisson est atteint* »

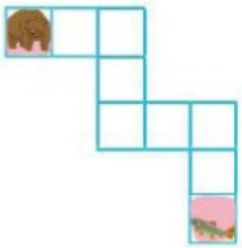
On utilise la boucle « Répéter jusqu'à » quand on ne sait pas combien de fois on doit répéter les instructions mais quand on sait à quel moment on doit s'arrêter.

À toi de jouer !

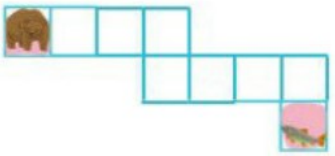
Exercice 1 À la pêche !

Dans chacun des cas ci-dessous, écrire des algorithmes permettant à Ted d'atteindre le poisson à l'aide des quatre instructions ◀ ▶ ▲ ▼ et d'une boucle.

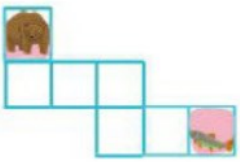
1.



2.

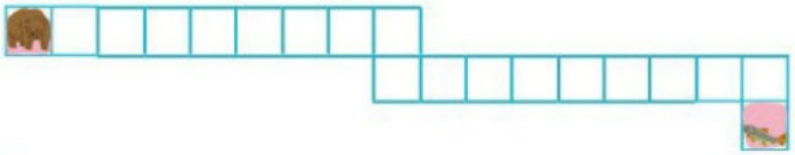


3.



4.

5.



Aide

On peut utiliser plusieurs boucles. De plus, une boucle peut contenir plusieurs instructions.

Exercice 2 Des polygones en boucles

1. Reprendre le programme ci-contre en le simplifiant à l'aide d'une boucle afin de construire un hexagone, puis enregistrer ce programme.
2. Faire évoluer le programme précédent afin d'obtenir un octogone de côté 50.
3. Faire évoluer à nouveau le programme afin d'obtenir un polygone dont le nombre de côtés est demandé à l'utilisateur.

Pour s'entraîner

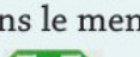
Les différents types de **boucles** de Scratch sont dans le menu Contrôle .

répéter 10 fois

répéter jusqu'à

répéter indéfiniment

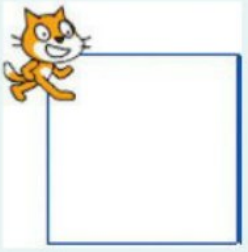
• On peut indiquer le nombre de répétitions souhaitées : ce nombre peut être une variable.

• La répétition est ici effectuée jusqu'à ce qu'un test soit validé. Ces tests sont dans le menu Opérateurs . On a, par exemple :  ,  ou encore  .

• Cette boucle est utilisée par exemple quand on attend une réponse au clavier. Cela nécessitera généralement l'usage d'une instruction conditionnelle : elle **sera placée dans ce bloc**.

Exemple

Ce programme va permettre de faire tracer un carré de côté 100 à chaque fois que l'utilisateur appuie sur  .



Projet 1 début Rebonds

1. Crée ce projet avec le lutin « Basketball ».

2. Crée et teste ce script :

3. Le script ne fonctionne pas. Dee a commencé à réécrire l'algorithme :

Quand le jeu commence :
Initialisation.
Répéter indéfiniment :
.....
.....

a. Complète l'algorithme ci-dessus.

b. Que contient la partie « Initialisation » ?

.....
.....

c. Teste le script corrigé.

4. On voudrait réduire la taille de la balle et la placer initialement au hasard sur la scène.

Où doit-on placer ces briques ?

.....
.....

5. Teste et enregistre ton projet.

Projet 2 début Les 100 pas

1. Crée ce projet avec le lutin « Jaime Walking ».

2. Crée le script qui fait avancer sans arrêt le lutin de 5 pixels et le fait rebondir sur le bord.

3. a. Que se passe-t-il d'étrange quand le lutin rebondit ?

.....
.....

b. Pour éviter ce problème, Dee utilise 

. Teste les trois options proposées. Laquelle permet de retourner correctement le lutin ? Que fait-il avec les deux autres ?

.....
.....
.....
.....
.....

4. Teste et enregistre ton projet.

Utilisation des variables

Une **variable** est une boite dans laquelle on stocke une information pour l'utiliser plus tard. On désigne une variable par un nom.

Exemple : Le score d'un joueur est une variable qui pourra évoluer tout au long du jeu, une nouvelle valeur efface la précédente.

À toi de jouer !

1 On compte

1. Créer une variable « COMPTEUR » et écrire un programme qui fasse augmenter de 1 cette variable à chaque clic sur le drapeau vert. Enregistrer ce programme.
2. Améliorer le programme en faisant dire au chat : « Le compteur est arrivé à ... »

Aide

Utiliser l'opérateur 

2 Qui es-tu ?

Programme un algorithme demandant ton prénom, puis ton âge et enregistre ce programme. Fais alors dire au chat : **Bonjour** *ton prénom*, puis **Tu as** *ton âge* **ans.**

Aide

Utiliser l'instruction 

.

On attribue alors la réponse à la variable créée 

Compter des points

Objectif Gérer des variables.

Les variables sont très utiles en programmation. On peut considérer une **variable** comme une « boîte » contenant une valeur. Chaque boîte a une étiquette : c'est le nom de la variable.

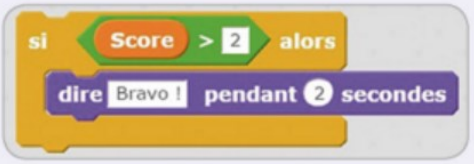
On parle de « variable » car elle peut varier au cours du temps. Au début d'un jeu, une variable « score » peut valoir 0, puis être augmentée par la suite. On dit qu'elle est **incrémentée**.

Pour créer une variable, clique sur « Créer une variable » dans la catégorie **Données** et choisis « Pour tous les lutins ».

- Visualiser (ou non) la variable sur la scène.
- Initialiser la variable avec un nombre ou du texte.
- Augmenter ou incrémenter la variable score d'une valeur numérique k (ici $k = 1$).
- Faire apparaître ou cacher la variable sur la scène lors du déroulement d'un script.

Pour certains projets, on peut utiliser  pour comparer des variable ou  pour effectuer des opérations. Ces briques sont proposées dans la catégorie **Opérateurs**.

Par exemple, on peut ainsi tester si une variable dépasse une certaine valeur pour féliciter le joueur.



Les valeurs des variables sont conservées entre deux jeux, sauf si on les remet à 0 !

Exemple

Ce programme va demander un nombre et donner son double à chaque fois que l'utilisateur appuiera sur 

Entre ce programme dans Scratch et exécute-le afin de vérifier qu'il produit bien le résultat attendu.

Pour s'entraîner

J'applique

Attribuer une phrase à une variable

Salutations.

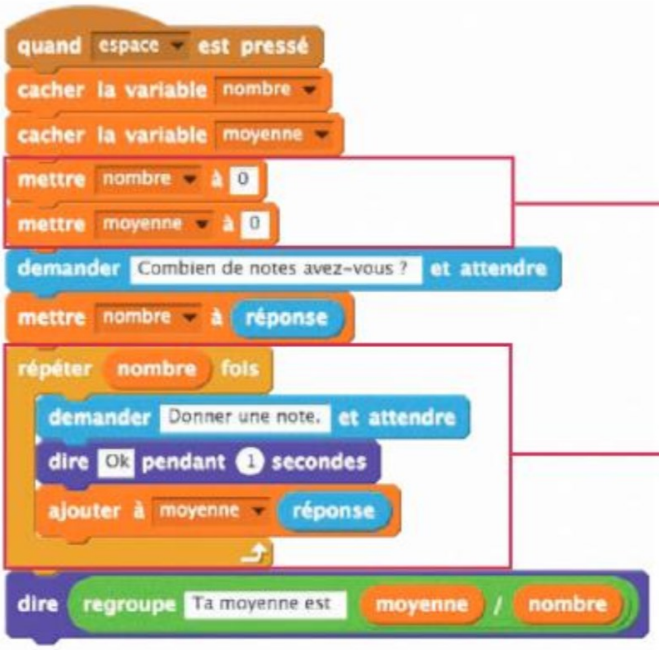
Programmez un algorithme qui demande son nom à l'utilisateur et le salue alors de manière personnalisée.

Attribuer une phrase à une variable

Le jeu du cadavre exquis.

Programmez le jeu du cadavre exquis qui crée des phrases en combinant des termes de manière aléatoire. Pensez à cacher la liste au début et à la vider pour effacer la partie précédente.

Exemple :



Remet à zéro les variables utilisées par le programme.

Ajoute au fur et à mesure chaque note de l'élève autant de fois que la valeur de la variable « nombre ».

Utilisation des instructions conditionnelles

Une **instruction conditionnelle** est de la forme :

Si « Condition »

Alors « Instruction(s) »

Dans ce cas, si la **« Condition »** est réalisée, alors les **« Instruction(s) »** seront effectuées.

Exemple
Si le soleil brille cet après-midi, alors j'irai à la piscine.

ou

Si « Condition »

Alors « Instruction(s) 1 »

Sinon « Instruction(s) 2 »

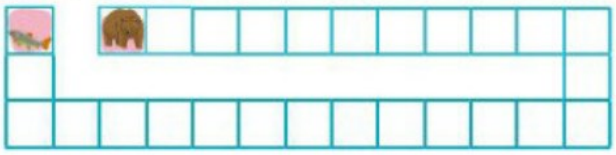
Dans ce cas, si la **« Condition »** est réalisée, alors les **« Instruction(s) 1»** seront effectuées, sinon ce sont les **« Instruction(s) 2 »** qui seront effectuées.

Exemple
Si le soleil brille cet après-midi, alors j'irai à la piscine, sinon j'irai au cinéma.

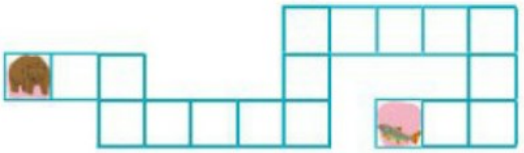
1 Case après case

Écrire une ligne d'algorithme permettant à l'ours Ted d'attraper le poisson dans chaque cas :

1. en utilisant une instruction conditionnelle de type « **Si** ... **Alors** ... »



2. en utilisant une instruction conditionnelle de type « **Si** ... **Alors** ... **Sinon**... »



2 Les tirs au but

1. Durant la Coupe du monde de football, à la fin du temps réglementaire d'un match, quand les équipes sont à égalité, le match continue durant une période que l'on appelle prolongations.
   Écrire un algorithme utilisant une instruction conditionnelle « **Si** ... **Alors** ... » pour décrire cette situation.

2. À la fin des prolongations, si les équipes n'ont pas réussi à se départager, elles entament une séance de tirs aux but. Chaque équipe tire cinq fois.
   Si à la fin de cette séance, les équipes sont toujours à égalité, vient la « mort subite ». Chaque équipe tire au but une fois jusqu'à ce que l'une marque et l'autre non.
   Écrire un algorithme utilisant une instruction conditionnelle « **Si** ... **Alors** ... **Sinon** ... . » pour décrire la phase de « mort subite ».

Pour s'entraîner

Les différents types d'instructions conditionnelles de Scratch sont dans le menu **Contrôle**

Les conditions sont à construire à partir des éléments du menu **Opérateurs** comme  ,  ou  .

Exemple : Ce programme donne la nature d'un nombre saisi par l'utilisateur.

Entre ce programme dans Scratch et exécute-le afin de vérifier qu'il produit bien le résultat attendu.

À toi de jouer !

Allez l'aléa !

1. Programmer un algorithme mettant dans une variable  une valeur aléatoire de 1 à 6, et faire cacher cette variable à l'écran. Recommencer avec une variable 

Aide

Utiliser l'instruction  et 

2. a. Compléter le programme pour qu'il demande ensuite à l'utilisateur de choisir un nombre entre 2 et 12.
   La demande doit être répétée tant que la valeur donnée n'est pas comprise entre 2 et 12.
   b. Comparer la valeur choisie par l'utilisateur à la somme des deux dés :
   – si les deux valeurs sont égales, dire : « Gagné ! » ;
   – si les deux valeurs sont différentes, indiquer la valeur la plus grande (somme des deux dés ou valeur de l'utilisateur).

La balade du chat

Programmer un algorithme faisant déplacer le chat vers la droite ou vers la gauche à l'aide des flèches du clavier.

Aide

Utiliser l'instruction 

ainsi que la boucle  .

Utilisation d'un bloc d'instructions paramétré

Un groupement d'instructions paramétré (un bloc) est constitué d'une suite d'instructions dont l'une au moins utilise une information précisée dans le bloc lui-même.

Exemple

On pourra utiliser ce groupement en commandant : Réveil matin(8 ; 2 ; croissants) ce qui donnera un **réveil** à **8h** avec **2 croissants**.
Les paramètres sont ici les nombres **8**, **2** et **croissants**.

Réveil matin
Mettre le réveil à **paramètre n°1** h
Prendre un petit-déjeuner avec **paramètre n°2** **paramètre n°3**

1 Travaux ménagers

En utilisant les instructions suivantes, écris deux groupements : l'un correspondant à « mettre la table pour une personne » et l'autre correspondant à « préparer une lessive ».

Mettre un verre

Mettre le linge dans le tambour

Préparer un pichet d'eau

Mettre les couverts

Mettre la lessive

Poser une assiette sur la table

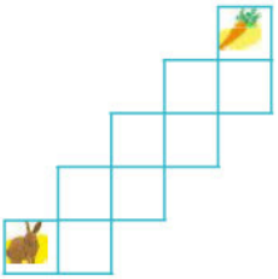
Mettre un adoucisseur si besoin

Choisir le programme de lavage

2 Le repas du lapin

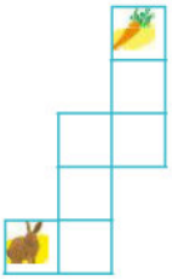
Dans chacun des cas ci-dessous, écrire le groupement d'instructions (GI) permettant au lapin de retrouver sa carotte à l'aide des quatre instructions ◀ ▶ ▲ ▼

1.



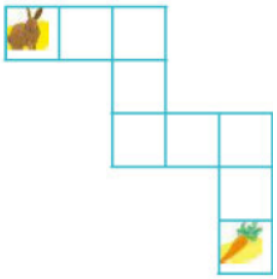
Répéter 4 fois
{ Groupement d'instructions GI1
Fin

2.



Répéter 2 fois
{ Groupement d'instructions GI2
Fin

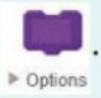
3.



Répéter 2 fois
{ Groupement d'instructions GI3
Fin

Utiliser un bloc d'instructions ici permet d'éviter d'avoir à mettre une boucle dans une boucle.

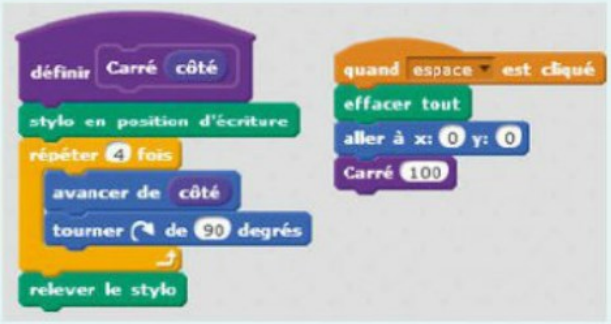
Pour s'entraîner

Lorsque l'on crée un bloc, on doit le nommer et on peut aussi choisir des options : 

Exemples

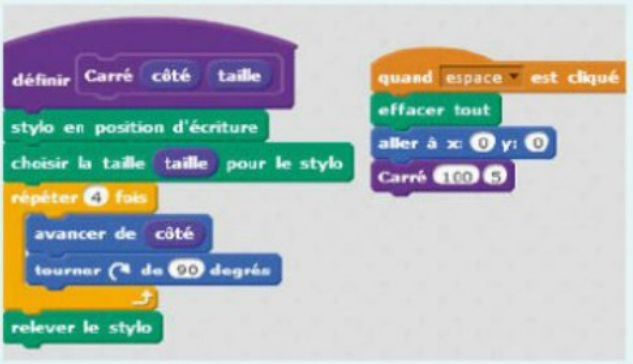
Ici, le bloc **Carré** a été créé avec un paramètre : *côté*.
Avec 

, on trace un *carré* de *côté* 100.

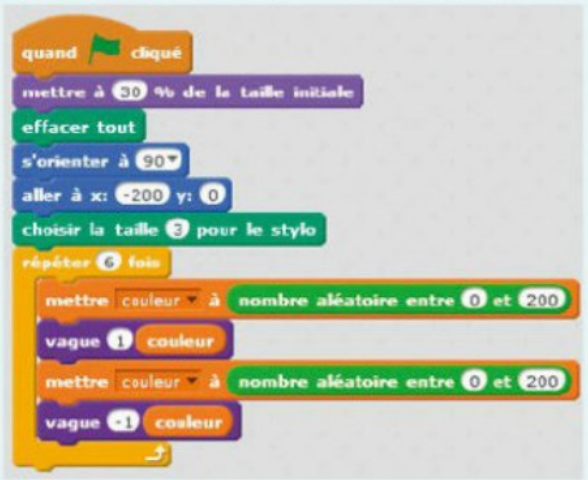


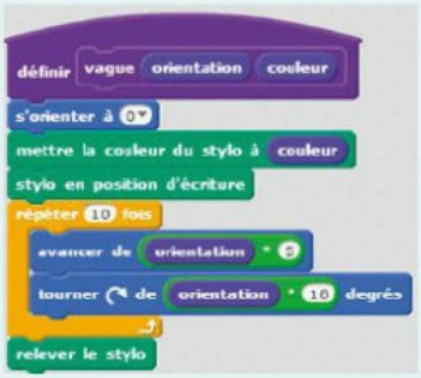
Ici, le bloc **Carré** a été créé avec deux paramètres : *côté* et *taille*.
Avec 

, on trace un carré de *côté* 100 avec un stylo de *taille* 5.



Exemple : Ce programme permet, à l'aide du bloc **vague**, d'obtenir un joli dessin.





Des croix

1. Écrire un algorithme contenant un bloc paramétré 

permettant de tracer ces deux croix en changeant les paramètres.
2. Modifier l'algorithme pour obtenir ce type de figure, sans changer le bloc.

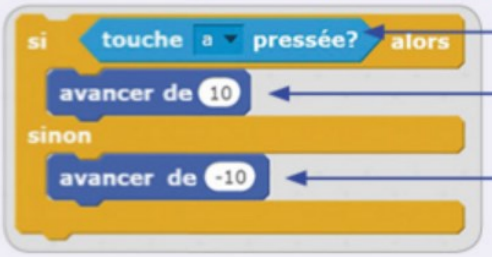
Quel cirque ! Maximus !

Écrire un algorithme faisant appel à un bloc « triangle équilatéral » pour construire un hexagone.

les capteurs

Voici comment utiliser des conditions dans des situations particulières :

- *Faire avancer un lutin de 10 si* A *est pressée, sinon reculer de 10 :*

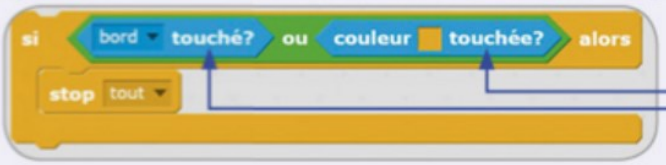


La condition.

Action **si** la condition **est** réalisée.

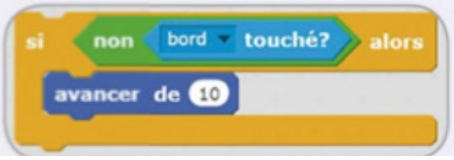
Action **si** la condition **n'est pas** réalisée.

- *Stopper le jeu si le lutin touche le bord ou s'il touche du jaune :*

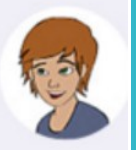


Deux conditions :
soit l'une **ou** soit l'autre.

- *Faire avancer le lutin si le bord est non touché :*



On utilise aussi la brique et pour tester si 2 conditions sont réalisées.



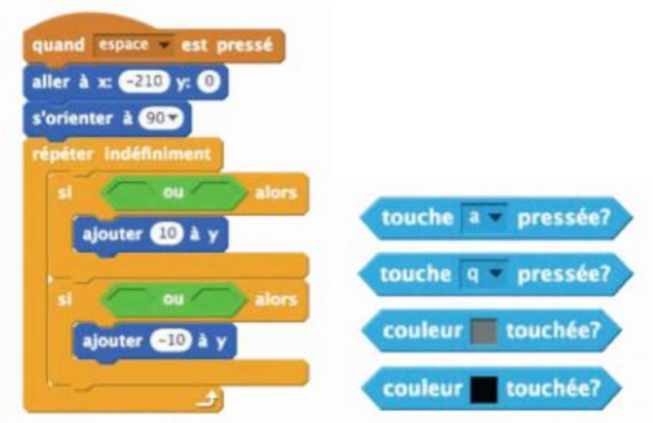
À toi de jouer !

J'applique

Utiliser les capteurs

Puzzle.

Remettez les blocs dans le bon ordre pour pouvoir coder le mouvement de la raquette gauche du jeu de Pong.



▲ Le mouvement de la raquette gauche.

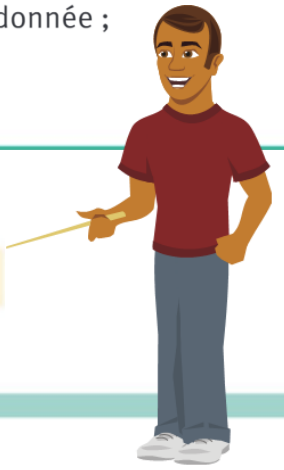
On veut faire monter et descendre verticalement la raquette grâce aux lettres a (monter) et q (descendre). De plus, il faut que la raquette rebondisse quand elle touche les bords haut et bas.

Pour s'entraîner

Types de capteurs

Les programmes peuvent interagir avec le joueur et la page graphique. Il y a plusieurs types de **capteurs** :

- les capteurs « demander », qui demandent à l'utilisateur d'entrer une donnée ;
- ceux qui testent si une touche du clavier est appuyée ;
- ceux qui testent si le lutin touche une couleur ou un autre lutin, etc.



Objectif

Manipuler des conditions avancées.

À toi de jouer !

Projet 1 Poursuite dangereuse

Le pingouin doit échapper au scarabée qui le poursuit sans toucher les obstacles verts.

1. Crée ce projet avec les lutins « Penguin1 » et « Beetle ».

2. Dessine, sur la scène, des disques verts de taille différente et suffisamment espacés comme ci-dessus.

3. Crée le script qui déplace indéfiniment le pingouin à la position de la souris.

4. Crée un autre script tel que, quand le pingouin est cliqué :

a. Il envoie le message « C'est parti ! »

b. Il teste indéfiniment s'il touche « Beetle » ou la couleur verte. Si oui : le jeu s'arrête.

5. a. Complète cet algorithme en suivant ces instructions :

Quand « Beetle » reçoit le message « C'est parti ! », il répète sans arrêt :

- *Il avance de 10.*
- *S'il touche du vert, il recule de 10, tourne à gauche de 60° et avance de 10.*
- *Sinon, il s'oriente vers le pingouin.*

Quand « Beetle » reçoit « C'est parti ! » :

Répéter indéfiniment :

Si .....

.....

.....

.....

Sinon :

.....

b. Crée le script associé à cet algorithme dans Scratch.

6. Teste et enregistre ton projet.

Projet 1 Le chiffre de César

Crypter… décrypter… voilà ce à quoi s'attaque le chiffre de César ! Cette forme simple de cryptage était déjà utilisée dans l'Antiquité. À ton tour de chiffrer un message comme les Romains !

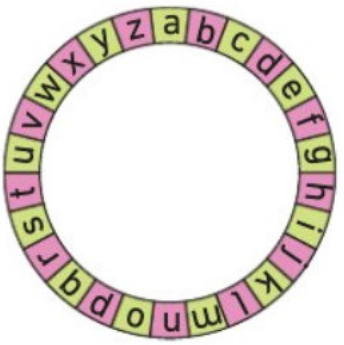
Étape 1 ■ Création et stockage de l'alphabet

– Le chiffre de César est un procédé qui consiste à décaler les lettres de l'alphabet vers la droite ou vers la gauche d'un nombre de crans déterminés.

On choisit ici de décaler vers la droite.

– Créer une variable alphabet contenant les 26 lettres de l'alphabet dans l'ordre.

Il s'agira de reconstituer le principe du cercle de lettres ci-dessous à partir des 26 lettres de l'alphabet dans l'ordre mises dans la variable.



Étape 2 ■ Saisie du décalage et du message à coder

– Demander le nombre correspondant au décalage souhaité.

– Demander le message à coder.

Attention ! Ce message sera tout d'abord un mot simple, c'est-à-dire sans trait d'union, sans apostrophe, etc.
On dispose désormais du message (mot) à coder et du décalage souhaité : il faudra, à l'étape suivante, créer le message codé.

Étape 3 ■ Création des variables nécessaires au codage

– Créer une variable message codé qui sera initialisée par du vide.

– Créer ensuite une variable lettre qui prendra successivement comme valeur chacune des lettres du mot à coder.

– Créer une variable compteur qui permettra de compter les lettres codées et que l'on initialisera à 1.

Projet 1 Le chiffre de César

Étape 4 ■ Création du bloc permettant le codage

– On affecte à la variable `lettre` la lettre à coder.

– On appelle un bloc `CODER` qui va :

• déterminer la position de cette lettre dans l'alphabet et la stocker dans une variable `position`, en utilisant éventuellement l'instruction `lettre 1 de world` ;

• ajouter le `décalage` à la variable `position` pour trouver la position de la lettre codée ;

• affecter à la variable `lettre` sa nouvelle valeur.

Par exemple avec la lettre **b** et un décalage de **3**, on aura comme valeur successives des variables :

- `lettre b`
- `CODER`
- `position 2`
- `position 5`
- `lettre e`

Attention ! Dans le cas où la somme `position` + `décalage` dépasse 26, il faut trouver un moyen de redémarrer au début de l'alphabet (27 correspond à 1, 28 à 2, etc.)

Pour cela, utiliser l'instruction `( ) modulo ( )` qui donne le reste d'une division euclidienne.

Étape 5 ■ Création et affichage du message codé

– Créer une boucle qui va, pour chacune des lettres du message initial, trouver celle qui doit la remplacer à l'aide du bloc `CODER` et l'ajouter à la suite du message codé.

On pourra utiliser pour la boucle :

`longueur de world` et `répéter ( ) fois`

et pour construire le message codé :

`regroupe message codé lettre`

Vérifier que le codage fonctionne bien, par exemple en tapant l'alphabet entier comme message avec un décalage de 3 pour observer si chaque lettre se code bien.

Étape 6 ■ Amélioration du programme pour coder des phrases entières

– Modifier le bloc `CODER` pour prendre en compte les espaces, les apostrophes ou les traits d'union dans une phrase.

Si le programme ne trouve pas la lettre à coder dans l'alphabet lors de la recherche de position, il doit conserver la lettre d'origine.

Un espace sera donc codé par un espace, une apostrophe par une apostrophe, etc.

On pourra pour cela utiliser `( ou )` dans la boucle.

Projet 2 Le crabe aux pinces magiques

Dans un univers maritime, l'antre d'un crabe est attaqué par des ballons ! Heureusement, ses pinces magiques lui permettent de détruire ces ballons avant qu'ils n'atteignent leur but !

Étape 1 ■ Positionner et déplacer le crabe

- Prendre comme lutin un crabe et le positionner en bas de l'écran.
- Programmer le déplacement horizontal (gauche et droite) du crabe à l'aide des flèches du clavier.

Pense à réduire la taille du crabe si besoin.

Étape 2 ■ Positionner et déplacer un ballon

- Ajouter un ballon qui descend automatiquement en démarrant toujours de la même abscisse –170. Cette abscisse sera modifiée par la suite.

Pense aussi à réduire la taille du ballon si besoin.

Étape 3 ■ Programmer les actions

- Modifier le programme pour que lorsque le crabe touche le ballon, celui-ci réapparaisse en haut à son point de départ.
- Prévoir aussi un retour au point de départ si le ballon arrive en bas de la scène, sans être touché par le crabe.

Étape 4 ■ Ajouter un décor et un compteur

– Ajouter un compteur permettant de savoir combien de fois le crabe a touché le ballon.

– Ajouter aussi un arrière-plan adapté.

Un double-clic sur

permet d'obtenir cet affichage

. On obtient le même résultat en faisant un clic droit et en choisissant dans le menu déroulant.

Étape 5 ■ Faire apparaitre le ballon n'importe où

– Faire en sorte que le ballon puisse surgir de n'importe quelle abscisse entre –200 et +200.

C'est l'occasion d'utiliser l'élément

Étape 6 ■ Créer une fin pour le jeu

– Terminer le jeu avec les indications suivantes :

- lorsque le compteur atteint 5, augmenter la vitesse de déplacement du crabe ainsi que la vitesse de descente des ballons ;
- lorsque le compteur atteint 10, augmenter encore la vitesse de descente des ballons ;
- lorsque le compteur atteint 20, faire dire au crabe : « Gagné ! » et arrêter le jeu.

Projet 3 Jeu de Nim

Ce jeu très classique peut être programmé...
Il peut même être programmé pour que l'ordinateur gagne à chaque fois ! Joli défi !
Il s'agit de retirer un, deux ou trois crayons à chaque tour.
Le vainqueur est celui qui retire le dernier crayon !

Étape 1 ■ Choix du premier crayon

– Choisir un crayon comme lutin. Le nommer ***crayon1***.

L'objectif est de programmer au maximum l'algorithme de ce crayon, pour le dupliquer ensuite et en avoir au total 16.
Il restera alors juste quelques modifications à apporter crayon par crayon.

Étape 2 ■ Programmation du premier crayon

– Lorsque le drapeau vert est pressé, montrer le crayon au bon emplacement.
Par exemple, à ces coordonnées :

– Programmer le crayon pour qu'à la réception du message ***crayon1,*** il se déplace verticalement vers le bas, puis se cache.

On pourra ainsi déclencher ultérieurement cette action par l'instruction :

Tu peux d'ailleurs tester qu'un clic sur cette instruction déclenche bien cette action.

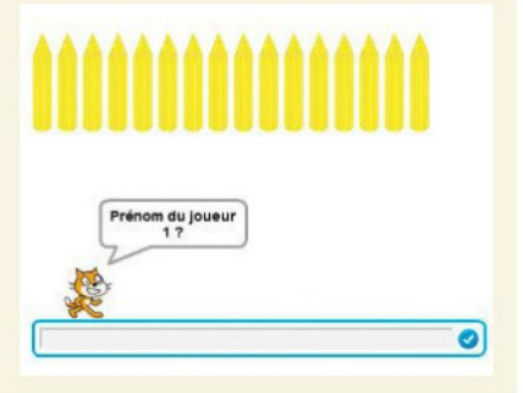
Étape 3 ■ Création des autres crayons

– Dupliquer 15 fois le ***crayon1*** et renommer à chaque fois le nouveau crayon obtenu : ***crayon2***, ***crayon3***, etc.
– Modifier l'endroit de départ où doivent apparaitre ces crayons pour qu'ils soient côte-à-côte et régulièrement espacés comme ci-contre.
– Pour chaque crayon, changer le nom du message qui le fera se déplacer et se cacher : ***crayon2, crayon3, etc.***

Projet 3 Jeu de Nim

Étape 4 ■ Programmation du chat

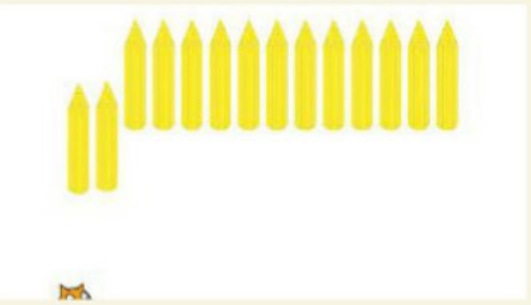
- Lorsque le drapeau vert est pressé, programmer le chat pour qu'il demande le prénom du joueur 1, puis celui du joueur 2 et qu'il les stocke dans des variables « JOUEUR1 » et « JOUEUR2 ».
- Une variable « COMPTEUR » est créée et initialisée afin de déterminer le nombre de crayons déjà enlevés et donc le crayon auquel on est arrivé.
- Le chat doit demander indéfiniment et à tour de rôle à chaque joueur combien de crayons il désire enlever. Ce nombre doit être 1, 2 ou 3, sinon le chat doit reposer la question.
  Stocker à chaque fois ce nombre dans une variable.

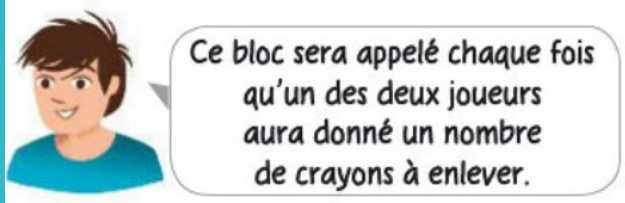


Étape 5 ■ Traitement du nombre de crayons à enlever

- Créer un bloc dans lequel la variable « COMPTEUR » augmente de 1 autant de fois que l'utilisateur a choisi d'enlever des crayons. Le bloc doit envoyer à tous le message *crayon1* si le compteur est égal à 1, le message *crayon2* si le compteur est égal à 2, etc.

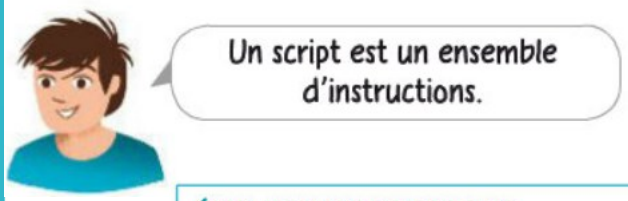
  Ainsi les crayons disparaitront l'un après l'autre.





Étape 6 ■ Fin du jeu

- Modifier le bloc pour qu'il teste à chaque fois si un des joueurs a gagné en fonction de la valeur de la variable « COMPTEUR ». Le joueur qui prend le dernier crayon a gagné.
- Faire afficher le prénom du joueur qui a gagné et stopper tous les scripts.



ÉVOLUTIONS POSSIBLES

- Modifier le jeu pour que le gagnant soit celui qui ne prend pas le dernier crayon.
- Faire évoluer le jeu pour qu'un joueur joue contre l'ordinateur.
- Imaginer un arrière-plan évoluant tout au long de la partie.

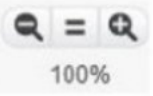
Projet 4 Le mage et la grenouille

Un mage doit viser une grenouille avec sa baguette magique.
Lorsque la grenouille a reçu assez de magie, elle se transforme en prince ou en princesse…

Étape 1 ■ Préparer la grenouille

– Choisir comme lutin la grenouille de Scratch et aller dans l'onglet costumes.
– Dupliquer la grenouille.
– Avec l'outil Gomme, supprimer la langue sur le costume 1.

Pour plus de précision dans l'usage de la gomme, on peut utiliser l'outil loupe en bas à droite de l'écran.

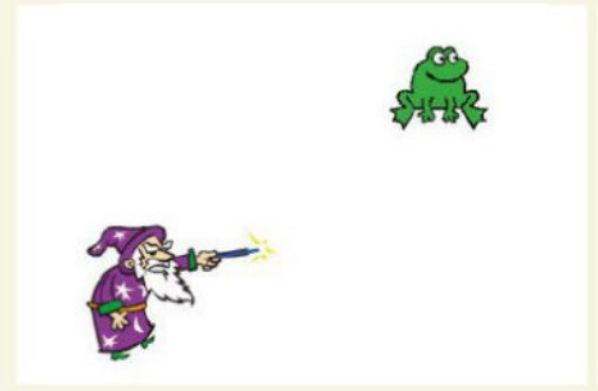


Étape 2 ■ Le mage et sa magie

– Écrire un algorithme de déplacement de la grenouille dans les quatre directions. On pourra utiliser

– Ajouter le lutin *Wizard* (le nommer *mage*) et créer un lutin *magie* de ce style :

– Créer une variable x mage et une variable y mage.
– Attribuer à x mage une valeur aléatoire entre –210 et 60 et à y mage une valeur aléatoire entre –120 et 120.
– Créer une variable angle qui pourra prendre une valeur aléatoire entre –30 et +30. Faire tourner le *mage* de cette valeur.



Étape 3 ■ Introduction du hasard dans le jeu

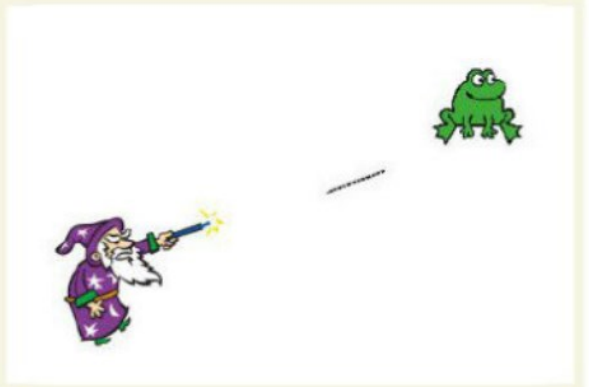
– Créer une variable dé, lui attribuer une valeur aléatoire entre 1 et 10.

Si la valeur de dé est inférieure à 8, le *mage* envoie de la magie par sa baguette : à cet effet, un message est envoyé au lutin *magie*.

Prévoir ensuite une temporisation d'une seconde environ.

Sinon, on refait bouger le mage et on relance le dé.

À la réception de ce message, *magie* se positionne par rapport à la baguette du *mage* en prenant la même orientation. *Magie* avance alors jusqu'à toucher le bord : il est désormais caché.



Projet 4 Le mage et la grenouille

Étape 4 ■ Traitement de l'événement victorieux

– Dès que la grenouille est touchée par *magie*, elle tremble en tirant et rentrant la langue.

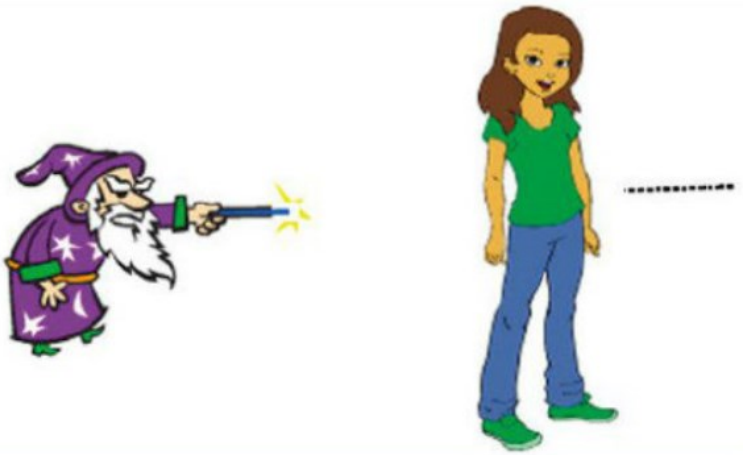
L'objectif est d'être touché le plus souvent possible par *magie*.

Un **compteur** doit donc mémoriser le nombre de fois où la grenouille est touchée.

Étape 5 ■ Traitement de la fin de partie

On peut désormais cacher le compteur.

À partir de la 11e fois où la grenouille a été touchée :
– si le dé donne 1, la grenouille se transforme en prince (autre lutin), et le jeu s'arrête ;
– si le dé donne 2, la grenouille se transforme en princesse (autre lutin) et le jeu s'arrête ;
– dans les autre cas, le jeu continue.



Étape 6 ■ Comptabilisation des points

– Choisir un arrière-plan.

– Terminer le jeu en créant une variable « POINTS ».

- Au démarrage, on a 1 000 points.

- À chaque fois que la grenouille tremble, on ajoute 200 points.
- À chaque fois que la grenouille n'est pas touchée, on perd 100 points.
- Lorsque le compteur dépasse les 10, on a 20 points à chaque tir jusqu'à la transformation.

Projet 5 Le nombre mystère

Es-tu capable de trouver un nombre mystère choisi par le chat ?
Tu devras réussir avec le moins d'essais possibles...
Une bonne stratégie te permettra de gagner systématiquement en 10 ou 11 coups maximum.

Étape 1 ■ Programmer la base du jeu

– Écrire un algorithme dans lequel une variable cachée prend une valeur aléatoire entre 1 et 1000 qu'il faudra retrouver.
– Demander à l'utilisateur de proposer un nombre.

Pour chaque proposition, le chat indique si la valeur cherchée est plus grande ou plus petite que la valeur proposée.

Étape 2 ■ Comptabiliser le nombre d'essais

– Compléter l'algorithme en indiquant le nombre d'essais effectués pour identifier le nombre mystère.

– Lorsque le nombre cherché a été trouvé, faire dire au chat en combien de coups cela a été réussi.

Étape 3 ■ Ajouter un chronomètre

– Ajouter un chronomètre sur lequel le temps de jeu défile en permanence jusqu'à ce que la valeur soit trouvée.

– Lorsque la valeur exacte est trouvée, le chronomètre n'apparait plus à l'affichage et le chat doit indiquer le nombre de coups et le temps qu'il a fallu pour trouver le nombre.

Projet 5 Le nombre mystère

Étape 4 ■ Établir un système de calcul de points

– Calculer un score et le faire dire au chat à la fin de la partie. Par exemple, le score peut être de 1 000 points auquel on enlève 50 points à chaque essai et 10 points par seconde passée.

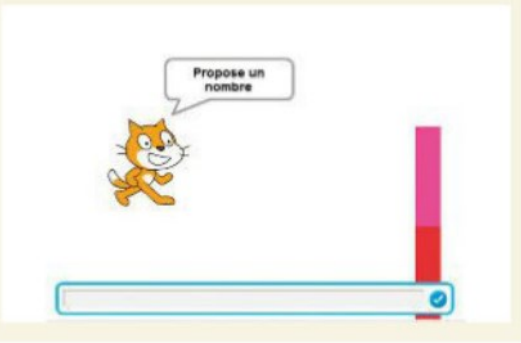
Le score peut n'être calculé qu'à la fin ou bien être affiché et diminuer au fur et à mesure. Pense à utiliser arrondi de pour que le score soit un nombre entier.

Étape 5 ■ Faire apparaitre une jauge

– Masquer le chronomètre et à la place, créer une jauge qui diminue au fur et à mesure pendant une durée d'environ 60 secondes.

Aide

Pour la jauge : créer un lutin à partir de rectangles de couleur.
Un autre lutin blanc de même forme permettra de le cacher peu à peu.



– Si le nombre mystère n'est pas trouvé dans ce délai, un message disant : « Pas assez rapide ! » apparait.
– Un deuxième message : « Il fallait trouver ... » apparait ensuite.
– Le jeu s'arrête alors.

Étape 6 ■ Encadrer le nombre cherché

– Faire apparaitre à l'écran un encadrement du nombre cherché en fonction des propositions faites.
– Pour faire cet encadrement, il faut stocker dans une variable la plus grande des propositions inférieures au nombre cherché, et stocker dans une autre la plus petite des propositions supérieures au nombre cherché.

ÉVOLUTIONS POSSIBLES

– Créer un arrière-plan.
– Faire demander au début du jeu entre quels nombres doit se trouver le nombre mystère (donc pas forcément entre 1 et 1000).
– Créer un autre lutin qui fera des propositions pour retrouver le nombre et le programmer de manière à ce qu'il soit le plus performant possible.

نفعنا
الله و اياكم
زملائي الكرام

mahjoubi.bilel@gmail.com

Cahier d'activite
2019-2020