

# MITIC 4e

INSTITUT  
*florimont*

01000001

01110101011101000110010000000111010001000000100001001  
111010101110010011100110010000000111010001000000100001001  
100101011011100110111101101001011101000010000001001110011  
000010110010001100001011101010110010000100000011001010111  
01000010000001010011011010010110101101110110111000100  
0000101100110010101110010011001000110000101101110001011  
1000100000010010010110111001101110100011010010111010  
0011101011101000010000001000110011011000110111101110010  
011010010110110101101111011011100111010000101110000000000

Informatique 4<sup>e</sup> – Fiches MITIC

Institut Florimont

Petit-Lancy (Suisse)

© Tout droit réservé. Crédit photographie couverture : Institut Florimont. Illustration des premières pages de chapitre issue de *Codex Leicester* de Leonardo da Vinci (domaine public).

2<sup>ème</sup> édition v2.0

juin 2021



# Informatique 4<sup>e</sup>

## Fiches MITIC

Institut Florimont

Ce livret appartient à .....

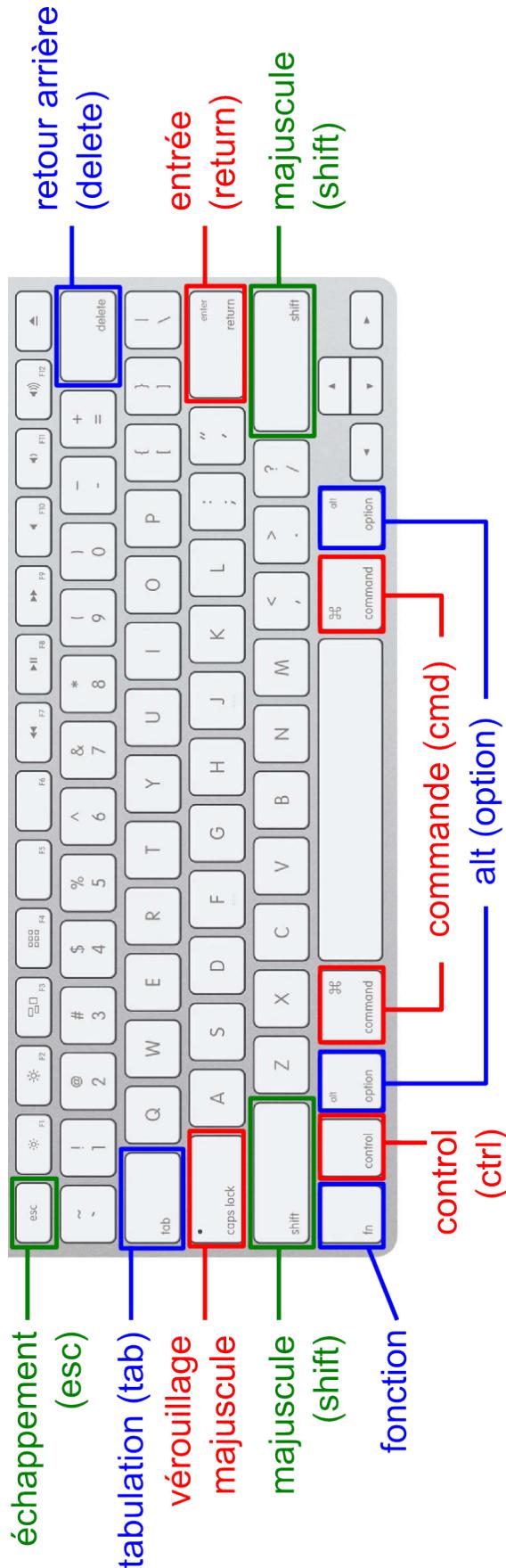


# **Table des matières**





## Les touches spéciales du clavier



Pour sauvegarder son travail : cmd + S

Pour annuler la dernière opération : cmd + Z

# Philosophie du document

Vous avez entre les mains le troisième tome d'une série de trois fascicules qui accompagneront les élèves des classes de 6<sup>e</sup>, 5<sup>e</sup> et 4<sup>e</sup> jusqu'au moment où ils recevront un ordinateur qu'ils seront en mesure d'exploiter au mieux pour leur travail. Le but est de les rendre aptes à réaliser au mieux les nombreux travaux qu'ils auront à faire sur ordinateur au cours de leur scolarité secondaire.

Ce document se présente sous la forme d'un livret qui rassemble des compétences MITIC<sup>1</sup> permettant aux élèves d'apprendre à utiliser les logiciels et espaces numériques mis à leur disposition. Pour l'année de 4<sup>e</sup>, sont traités les logiciels *Microsoft Word* (traitement de texte), *Microsoft Excel* (tableur grapheur), *Gimp* (retouche d'image) et *Scratch* (programmation). Au début de chaque chapitre un lien permettant de télécharger le logiciel est fourni.

Les activités proposées sont conçues pour permettre d'exploiter à plusieurs occasions au cours de l'année les différentes compétences découvertes pour chaque logiciel. Le logiciel *Gimp*, par exemple, est utilisé lors d'une séance de français pour créer un calligramme (*séance 1*), exploité à nouveau en SVT pour légendier une image (*séance 2*) puis en arts visuels pour réaliser une composition graphique (*séance 3*) selon un calendrier proposé en début de chapitre. Lorsque cela était possible, nous avons essayé de faire coïncider les notions abordées dans les différentes séances avec le programme de la matière concernée.

Professeurs, c'est à vous que revient la tâche délicate d'inclure le contenu de ces fiches dans votre progression. À vous de le faire vivre : arriver en salle informatique et demander aux élèves de remettre en forme un texte de Molière ne présente que peu d'intérêt pédagogique. Donnez du sens à ces fiches et profitez-en pour diversifier votre enseignement. N'hésitez pas à exploiter dans vos cours les techniques présentées dans ce fascicule afin que les élèves utilisent plusieurs fois leurs nouvelles compétences et, par là-même, les pérennisent.

Merci d'avance à tous pour votre implication.

L'équipe de rédaction.

---

1. MITIC : Médias, Images et Technologies de l'Information et de la Communication.



Les ordinateurs sont des machines qui exécutent des programmes. On peut écrire des programmes dans différents *langages de programmation*, par exemple *Python*, *C++*, *Java*... ou encore *Scratch*.

## Synoptique

- Logiciel<sup>1</sup> : *Scratch 3.0*
- Matière concernée : mathématiques.
- Compétences :
  - créer des nouveaux blocs avec ou sans paramètre d'entrée ;
  - changer les arrière-plans de la scène ;
  - changer les costumes du sprite ;
  - créer puis détruire des clones d'un sprite.

## Les années précédentes, vous avez appris...

Les compétences listées ci-dessous ont été vues en classes de 6<sup>e</sup> et de 5<sup>e</sup>. Vous en aurez à nouveau besoin pour les activités de cette année. Si nécessaire, reportez-vous aux *Fiches MITIC* des années précédentes pour revoir comment :

- choisir et paramétriser l'objet sprite et l'objet scène (6<sup>e</sup>) ;
- créer/insérer un nouvel objet (6<sup>e</sup>) ;
- associer un code à un objet (6<sup>e</sup>) ;
- utiliser la structure conditionnelle if (bloc *si ..*) (6<sup>e</sup>) ;
- écrire un programme simple qui réponde à une problématique donnée (6<sup>e</sup>) ;
- créer une variable et modifier sa valeur (5<sup>e</sup>) ;
- utiliser la boucle for (bloc *répéter n fois*) (5<sup>e</sup>) ;
- utiliser la structure if .. then .. else (bloc *si .. alors .. sinon*) (5<sup>e</sup>) ;
- utiliser la boucle infinie (bloc *répéter indéfiniment*) (5<sup>e</sup>) ;
- lire un algorithme écrit sous la forme d'un *flowchart* (5<sup>e</sup>) ;

---

1. Le logiciel *Scratch* est librement téléchargeable : <https://scratch.mit.edu/scratch2download/>

- écrire un programme à partir d'un *flowchart* (5<sup>e</sup>).

## 1 Séance 1 : Créer des nouveaux blocs

(Auteur de cette séance : Thomas Morel, Institut Florimont)

### 1.1 Travail de préparation...

Afin de bien préparer la séance, vous pouvez regarder une courte vidéo explicative des principales fonctionnalités de *XXX* en suivant ce lien ou QR-code :

lien

QR-code

### 1.2 Pour bien démarrer...

#### Passer Scratch en langue française

Avant de commencer, vous pouvez si nécessaire choisir la langue de l'interface en cliquant sur l'icone en haut à gauche . Nous choisirons par exemple **Français**



#### Penser à enregistrer régulièrement

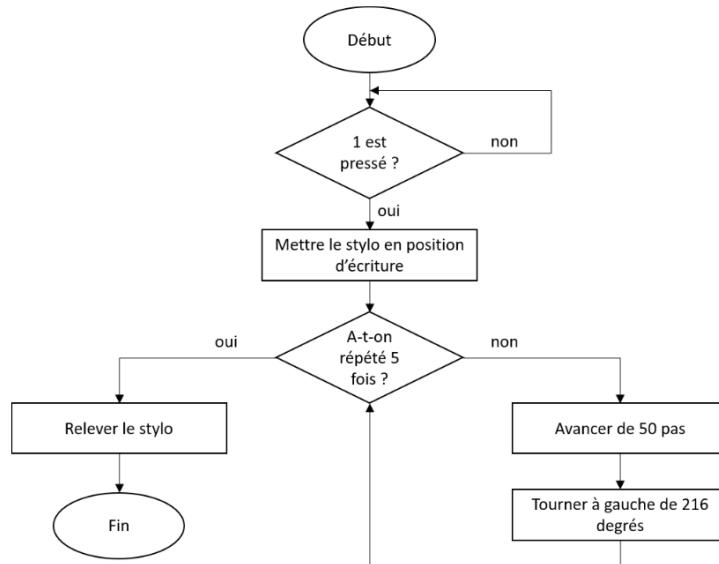
Dès que vous avez ouvert un nouveau programme dans Scratch, sauvegardez-le au format Nom-seance1.sb3 : dans le menu **Fichier**, choisir **Sauvegarder sur votre ordinateur**. Pendant que vous travaillez, pensez à sauvegarder régulièrement votre travail.

### 1.3 Sujet de l'activité...

Le but de cette séance est de découvrir une notion importante en programmation : la notion de *fonction*. Une fonction est une portion de code isolée que l'on peut appeler pour l'exécuter à chaque fois qu'on en a besoin. En *Scratch*, une fonction est créée quand on crée un nouveau bloc. C'est ce que nous allons découvrir lors de cette séance. Dans cette activité, nous allons exceptionnellement suivre les différentes étapes pas à pas afin de découvrir l'intérêt d'utiliser des fonctions. Il vous faudra donc lire tout le contenu de l'énoncé pour bien comprendre

### Créer une étoile à 5 branches

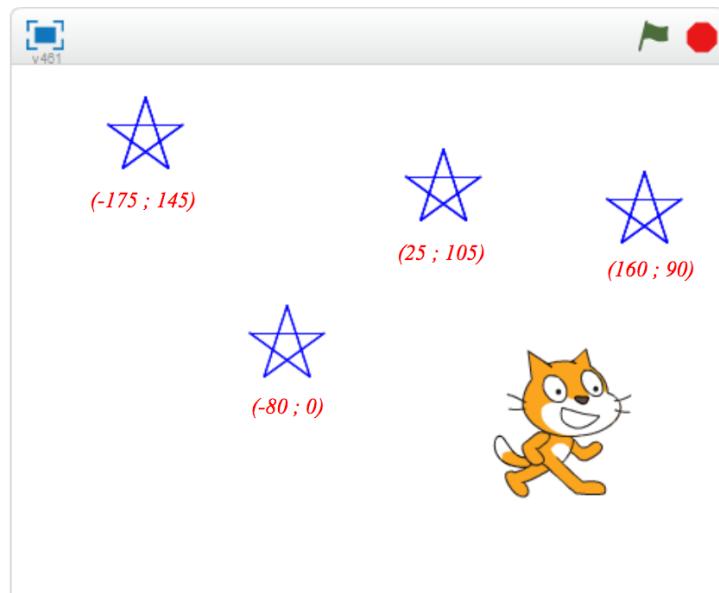
À partir de l'algorithme ci-dessous, créer un code qui dessine une étoile à 5 branches.



Le résultat doit ressembler à ceci : .

### Créer plusieurs étoiles

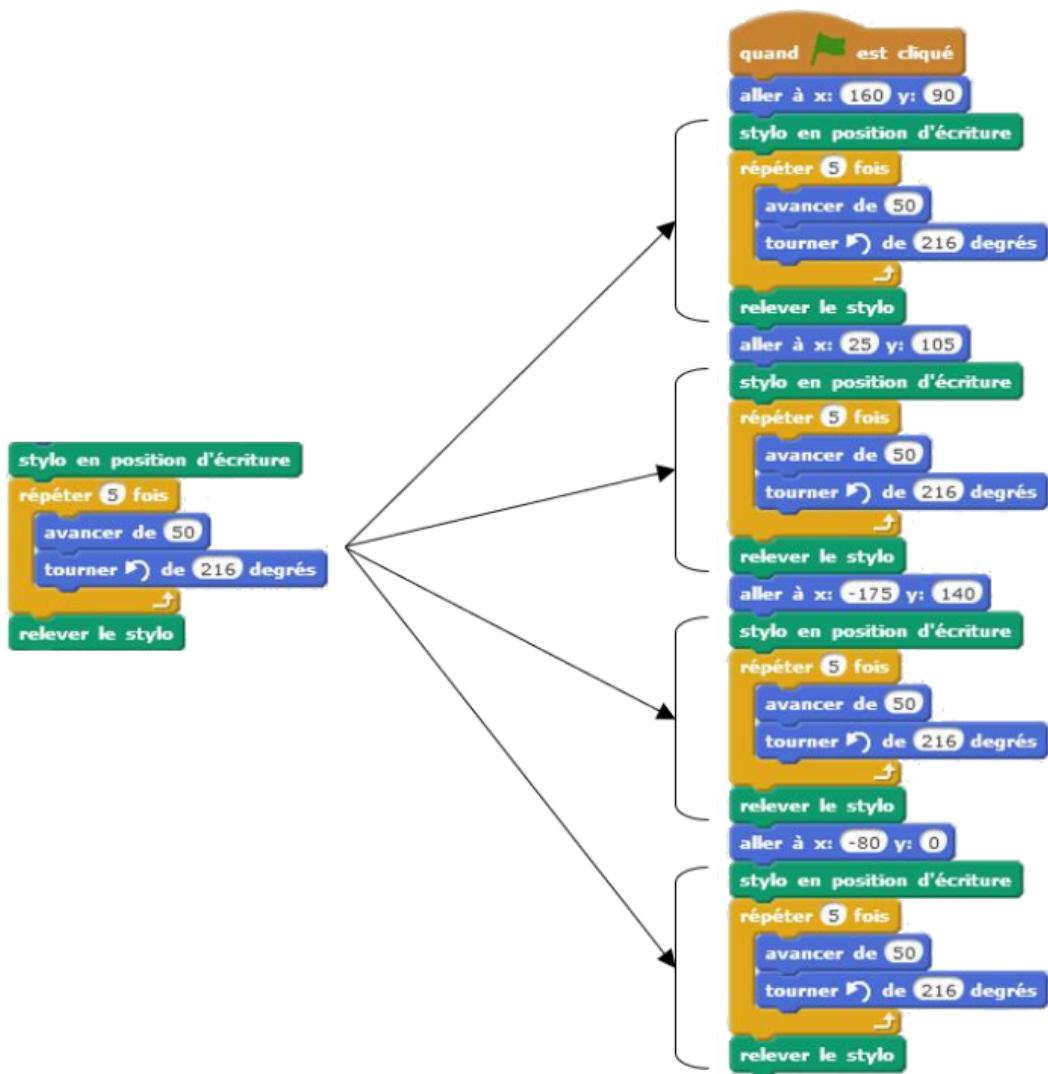
Modifier maintenant le code précédent afin d'obtenir la figure suivante (seules les étoiles sont à reproduire, les coordonnées sont mentionnées en rouge à titre indicatif).



Utiliser le bloc **aller à x: 0 y: 0** pour positionner les étoiles aux bons endroits. Que pensez-vous de votre dernier code? Combien de blocs avez-vous utilisés? Le code finalement obtenu est-il facile à comprendre?

### Créer un nouveau bloc

Le code précédent comporte beaucoup de blocs et paraît donc, à première vue, compliqué à lire. On remarque également qu'une partie du code est reproduite quatre fois.



Afin de ne pas répéter le même code plusieurs fois, nous allons créer un nouveau bloc qui contiendra le code dessinant l'étoile, que nous nommerons : *étoile* (figure ci-contre).

- ① Cliquer sur Ajouter blocs ;
- ② Cliquer sur le bouton Crée un bloc ;
- ③ Entrer le nom de votre bloc (*étoile*) ;
- ④ Valider en cliquant sur le bouton Ok.



Définir le bloc étoile de la manière suivante :



Il ne reste plus qu'à remplacer dans le code principal les parties de code correspondantes par le bloc **étoile**.

Vérifier que le nouveau code donne le même résultat que précédemment.

### À retenir...

Créer ses propres blocs évite de recopier du code qui apparaît plusieurs fois, ainsi le code devient plus court. Le programme ne sera pas plus rapide et le résultat sera le même, mais le code sera plus facile à écrire et à lire !

### Paramétriser un nouveau bloc à l'aide d'une variable

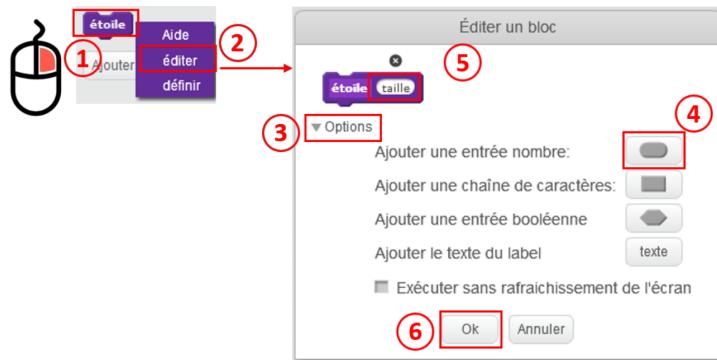
Il est possible de spécifier des paramètres en entrée d'un bloc créé par l'utilisateur afin de le rendre plus flexible. Plusieurs types de paramètres peuvent être utilisés en entrée du bloc :

- un nombre ;
  - exemple : un paramètre « taille » pouvant prendre les valeurs 30, 40, 50... qui définirait la taille de l'étoile dessinée.
- une chaîne de caractères ;
  - exemple : un paramètre « couleur » pouvant prendre les valeurs « rouge », « bleue »... qui définirait la couleur de l'étoile dessinée.
- une variable booléenne ;
  - exemple : un paramètre « valeur\_defaut » pouvant prendre uniquement les valeurs 0 ou 1. Par exemple, quand valeur\_defaut vaut 1, les valeurs de taille et couleur pourraient être respectivement forcées à 30 et en bleu.

Améliorons le bloc étoile afin qu'il puisse dessiner des étoiles de différentes tailles. Pour cela, il faut ajouter un paramètre d'entrée au bloc, que nous nommerons **taille**. Nous pourrons ainsi indiquer, au moment de l'appel du bloc, la taille de l'étoile.

### Ajouter un paramètre d'entrée de type nombre au bloc étoile :

- Effectuer un clic droit sur le bloc étoile (① sur la figure ci-dessous).
- Dans le menu qui s'affiche, cliquer sur **éditer** (②).
- Dans la fenêtre **Éditer un bloc**, dérouler le menu **Options** (③).
- Choisir l'option **Ajouter une entrée nombre** (④).
- Modifier le nom de votre paramètre d'entrée, choisir **taille** (⑤).
- Cliquer sur **Ok** pour terminer (⑥).



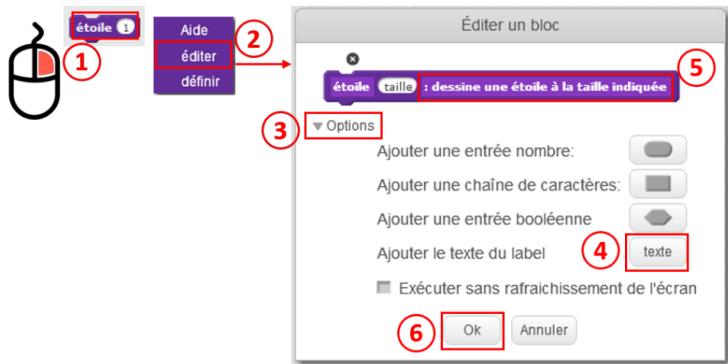
Redessiner les 4 étoiles, chacune ayant une taille différente : 30, 40, 50, 60.

### Ajouter un commentaire à un nouveau bloc

Pour faciliter la lecture du programme et comprendre rapidement à quoi sert le bloc créé, un commentaire peut être ajouté au moment de la définition du bloc.

### Ajouter un commentaire à la fonction étoile

- Effectuer un clic droit sur votre bloc étoile (① sur la figure ci-dessous).
- Dans le menu qui s'affiche, cliquer sur **éditer** (②).
- Dans la fenêtre **Éditer un bloc**, dérouler le menu **Options** (③).
- Choisir l'option **Ajouter le texte du label** (④).
- Entrer le commentaire : « dessine une étoile à la taille indiquée » (⑤).
- Cliquer sur **Ok** pour terminer (⑥).



Vérifier que le commentaire apparaît bien dans le code principal.

### À retenir...

Il est possible d'associer au bloc un ou plusieurs paramètres pour optimiser et adapter son comportement. Dans *Scratch*, ces paramètres peuvent être de 3 types :

- type nombre ;
- type chaîne de caractères ;
- type booléen.

Les noms donnés aux paramètres sont très importants, puisqu'ils permettent au lecteur de tout de suite comprendre l'utilité du paramètre. Dans le cas d'un bloc compliqué, il est important d'ajouter un commentaire décrivant succinctement son utilité.

### Pensez à rendre votre travail

Une fois votre travail terminé, vous exporterez votre fichier au format .sb3 (le fichier doit être nommé à partir de votre nom : Nom-seance1.sb3), puis vous le rendrez sur *Teams* à l'endroit indiqué par votre enseignant (si nécessaire, se reporter à la fiche méthode *Remettre son devoir*, page ??).

## 1.4 Pour aller plus loin...

### Plus de paramètres

Le code principal peut encore être réduit en intégrant les blocs  dans un nouveau bloc créé.

1. Créez un nouveau bloc intitulé **étoile&position** qui dessine une étoile en position (x,y) dont la définition est la suivante :

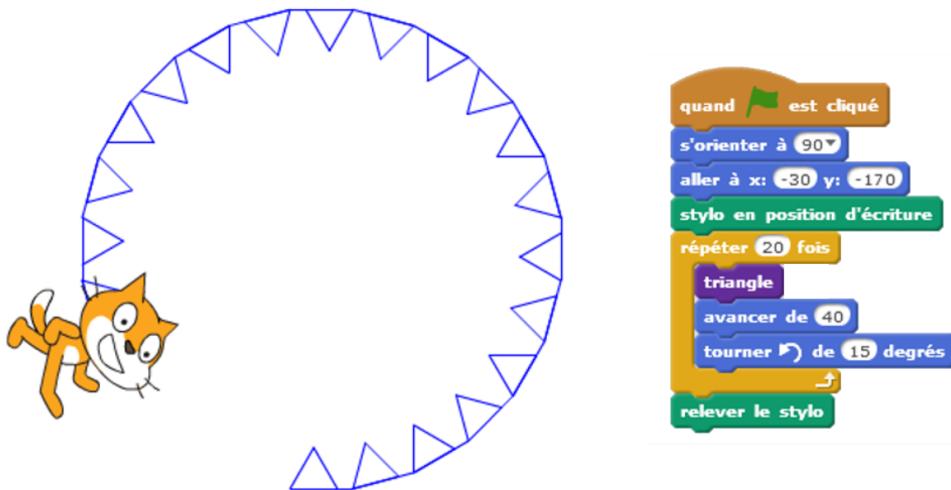


*Aide* : vous pouvez réutiliser le bloc **étoile** à l'intérieur du bloc **étoile&position** si vous le désirez.

2. Modifiez le code principal en utilisant le bloc **étoile&position**.

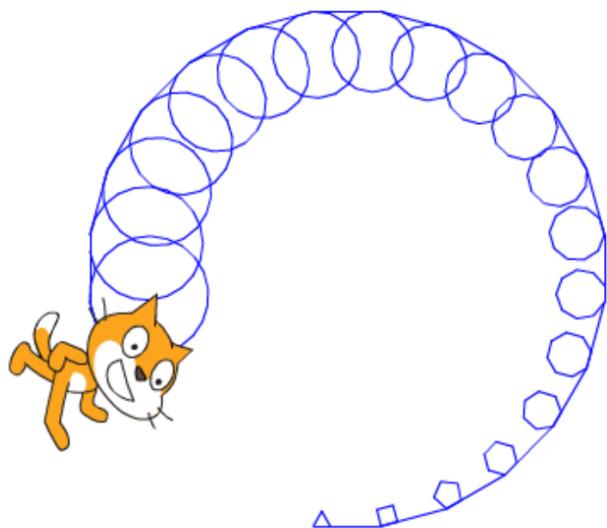
### Création d'un nouveau bloc « triangle »

Définir la fonction `triangle` contenu dans le code ci-dessous qui permet d'obtenir la figure suivante :



### De plus en plus de côtés...

Écrire le programme qui génère le résultat suivant :



*Aide* : il est possible de créer une variable `n` qui contient le nombre de côtés de chaque forme pour ensuite l'utiliser dans la boucle.

## 2 Séance 2 : Un clone de *Flappy Bird*

### 2.1 Travail de préparation...

Afin de bien préparer la séance, vous pouvez regarder une courte vidéo explicative des principales fonctionnalités de *Microsoft Excel* en suivant ce lien ou QR-code :

lien  
QR-code

### 2.2 Pour bien démarrer...

Dès que vous avez ouvert un nouveau programme dans Scratch, sauvegardez-le au format Nom-seance2.sb3 : dans le menu **Fichier**, choisir **Sauvegarder sur votre ordinateur**. Pendant que vous travaillez, pensez à sauvegarder régulièrement votre travail.

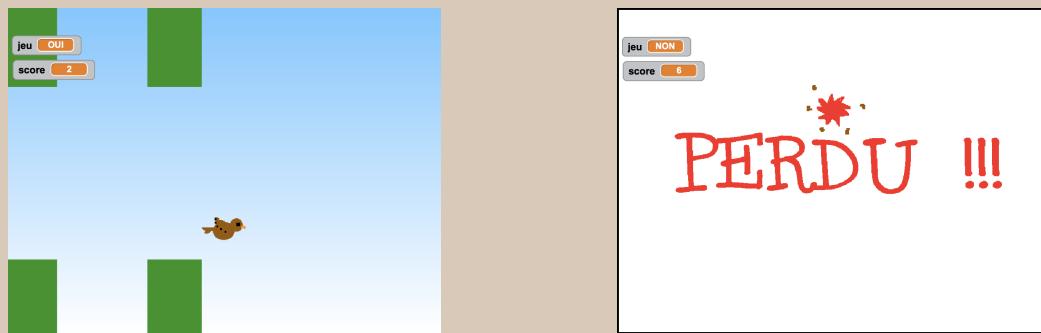
## 2.3 Sujet de l'activité...

Le but de cette séance est de créer un jeu de type *Flappy Bird*. C'est un jeu vidéo d'obstacles développé par Nguyen Hà Dong et sorti en mai 2013 (figure ci-dessous). Selon Wikipedia : « *Le gameplay repose sur l'agilité du joueur, qui doit faire avancer un oiseau dans un environnement à défilement horizontal en tapotant sur l'écran tactile, tout en évitant des tuyaux présents en haut et en bas de l'écran. Les règles de jeu sont très simples : lorsque l'oiseau touche un tuyau ou heurte le sol, la partie est terminée.* »



Le jeu créé ici sera très simple : à vous de l'améliorer pour qu'il ressemble davantage à l'original ou à vos désirs !

Que devons-nous faire pour créer un tel jeu ? Première étape : se renseigner pour voir à quoi ressemble le jeu *Flappy Bird*. Il est possible de trouver des vidéos sur *Youtube*. Voilà par exemple à quoi pourra ressembler notre clone de *Flappy Bird* : sur la figure ci-dessous à gauche, le jeu en cours d'exécution et à droite lorsque le joueur a perdu.



On va utiliser une nouvelle fonctionnalité dans *Scratch* : la création de différents *costumes* pour un *sprite* et la création de différents *arrière-plans* pour une *scène*.

Pour ce jeu il nous faut donc :

- des tuyaux (*sprite Tuyau* possédant différents *costumes*) ;
- un oiseau (*sprite Oiseau* possédant deux *costumes*) ;
- un décor (*scène* possédant deux *arrière-plans* différents).

Voilà les différents éléments (voir plus bas des indications pour leur création) :

- une *scène* qui contiendra comme premier *arrière-plan* le fond d'écran (figure à gauche ci-dessous) et comme second *arrière-plan* l'écran qui indique que le joueur a perdu (à droite ci-dessous) ;



- un *sprite* pour l'oiseau, qui contiendra comme premier *costume* l'oiseau du jeu (figure à gauche ci-dessous) et comme second *costume* l'oiseau après un crash contre un tuyau (à droite ci-dessous) ;



- un *sprite* pour les tuyaux, qui contiendra plusieurs *costumes* pour chaque tuyau du jeu (figures ci-dessous).



Une fois votre travail terminé, vous exporterez votre fichier au format .sb3 (le fichier doit être nommé à partir de votre nom : **Nom-seance2.sb3**), puis vous le rendrez sur *Teams* à l'endroit indiqué par votre enseignant (si nécessaire, se reporter à la fiche méthode *Remettre son devoir*, page ??).

**Pour obtenir de l'aide, rendez-vous à la page ??**

## 2.4 Pour aller plus loin...

Pour améliorer ce jeu, il est possible de :

- faire en sorte que si l'oiseau touche le sol la partie soit perdue ;
- modifier la vitesse de chute de l'oiseau et sa vitesse de remontée ;
- ajouter des vies à l'aide d'une variable (le joueur commence avec trois vies puis en perd une à chaque fois qu'un tuyau ou que le sol est touché) ;
- ajouter des sons (début du jeu, arrivée d'un tuyau, mort de l'oiseau, etc.) ;
- augmenter le rythme d'apparition des tuyaux ;
- améliorer la qualité des graphiques ;
- augmenter le nombre de costumes différents disponibles pour les tuyaux.

## 3 Séance 3 : Tracer une fonction affine

### 3.1 Travail de préparation...

Afin de bien préparer la séance, vous pouvez regarder une courte vidéo explicative des principales fonctionnalités de *Microsoft Excel* en suivant ce lien ou QR-code :

lien

QR-code

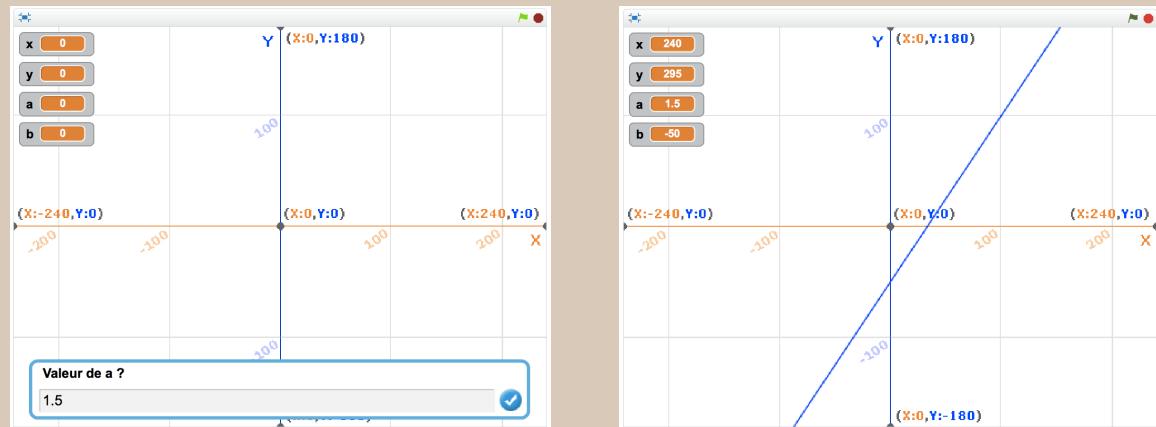
### 3.2 Pour bien démarrer...

Dès que vous avez ouvert un nouveau programme dans *Scratch*, sauvegardez-le au format Nom-seance3.sb3 : dans le menu **Fichier**, choisir **Sauvegarder sur votre ordinateur**. Pendant que vous travaillez, pensez à sauvegarder régulièrement votre travail.

### 3.3 Sujet de l'activité...

Le but de cette séance est d'utiliser *Scratch* pour réaliser le tracé d'une fonction affine (fonctions mathématiques de la forme  $y = a \times x + b$ ). Cette séance sera moins guidée que les deux précédentes : à vous de mettre en œuvre vos connaissances pour atteindre le but !

Les deux captures d'écran ci-dessous montre l'aspect du projet au moment où une donnée est entrée (figure de gauche) et après le tracer de la fonction (figure de droite).



Pour réaliser cette activité, la première étape consiste à sélectionner la scène et à choisir dans la bibliothèque des arrière-plans celui qui porte le nom **xy-grid** : il permet d'avoir à l'écran le repère dans lequel la fonction sera tracée.

Le programme utilise quatre variables qu'il faut créer : *a* (contient la valeur du coefficient directeur de la droite), *b* (contient l'ordonnée à l'origine), *x* (valeur de l'abscisse) et *y* (valeur de l'image de *x*).

Une fois votre travail terminé, vous exporterez votre fichier au format .sb3 (le fichier doit être nommé à partir de votre nom : **Nom-seance3.sb3**), puis vous le rendrez sur *Teams* à l'endroit indiqué par votre enseignant (si nécessaire, se reporter à la fiche méthode *Remettre son devoir*, page ??).

**Pour obtenir de l'aide, rendez-vous à la page ??**

### 3.4 Pour aller plus loin...

Vous avez réussi ? Félicitations !

Pour améliorer votre programme, vous pouvez :

- cacher les variables *x* et *y* dont l'affichage à l'écran n'est pas utile ;
- modifier la couleur et l'épaisseur du tracé ;
- modifier le programme pour qu'il trace une parabole (fonction de la forme  $y = a x^2 + b x + c$ ) ;
- modifier le programme pour qu'il trace une droite et une parabole.

## 4 Aide pour réaliser les activités

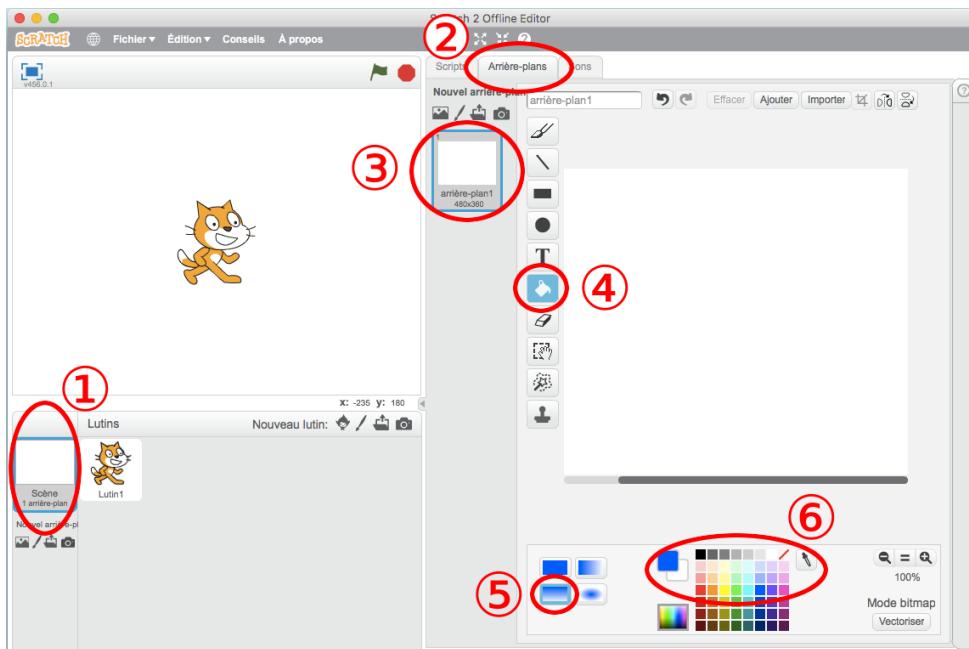
### 4.1 Aide pour la séance 1

Pour la séance 1, toutes les explications sont fournies dans l'énoncé.

### 4.2 Aide pour la séance 2

#### Construction des arrière-plans de la scène

On va tout d'abord créer l'arrière-plan du jeu. Une fois *Scratch* ouvert (figure ci-dessous), cliquer sur la scène ①, vérifier que l'onglet arrière-plans est sélectionné ②, choisir l'arrière-plan n° 1 ③, sélectionner le seau de peinture ④, puis l'option dégradé ⑤, choisir enfin une couleur ⑥. Terminer en cliquant dans la zone blanche pour peindre l'arrière-plan.



On va ensuite créer l'arrière-plan qui doit être affiché lorsque la partie est perdue. C'est un nouvel arrière-plan appartenant à la même scène.

Pour ajouter un nouvel arrière-plan à la scène en cours, cliquer sur le pinceau ① (figure ci-dessous), choisir ensuite l'outil texte ② puis la police à utiliser ③ et enfin la couleur ④. Terminer en cliquant dans la zone blanche pour écrire le texte souhaité. Il faut tirer sur les poignées qui apparaissent pour agrandir le texte jusqu'à la taille souhaitée, puis si nécessaire le déplacer à l'aide de la souris.



À partir de maintenant il est possible de changer l'arrière-plan de la scène grâce au bloc suivant, disponible dans les blocs **Apparence** :



### Construction des costumes du sprite Oiseau

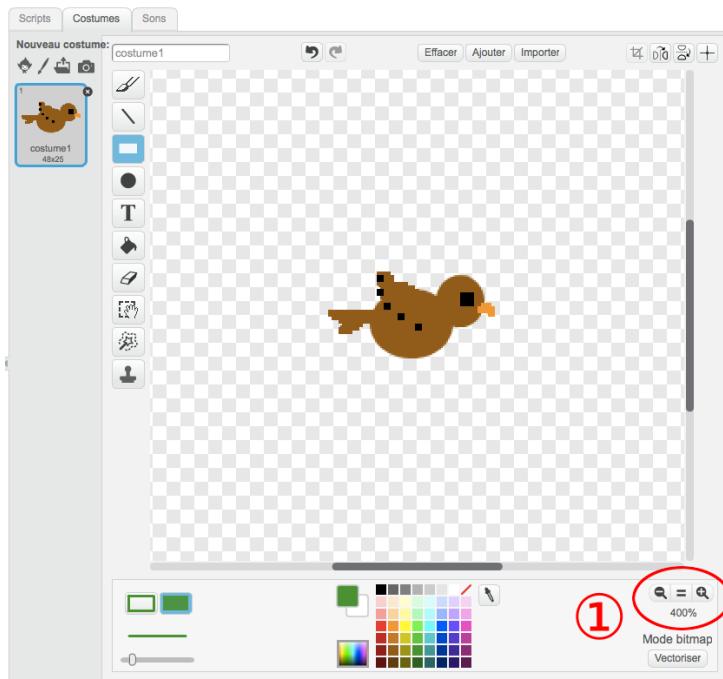
*Remarque : pour gagner du temps, il est possible d'utiliser directement des sprites de la bibliothèque plutôt que de dessiner deux nouveaux costumes pour notre sprite. Dans ce cas, il faudra réduire la taille d'affichage du sprite afin qu'il soit plus petit à l'écran. Pour cela, utiliser le bloc Mettre à .. % de la taille initiale disponible dans les blocs d'Apparence.*

*Si des sprites de la bibliothèque sont utilisés pour définir les deux costumes du sprite Oiseau, alors rendez-vous directement au paragraphe Construction des costumes du sprite Tuyau, § ??.*

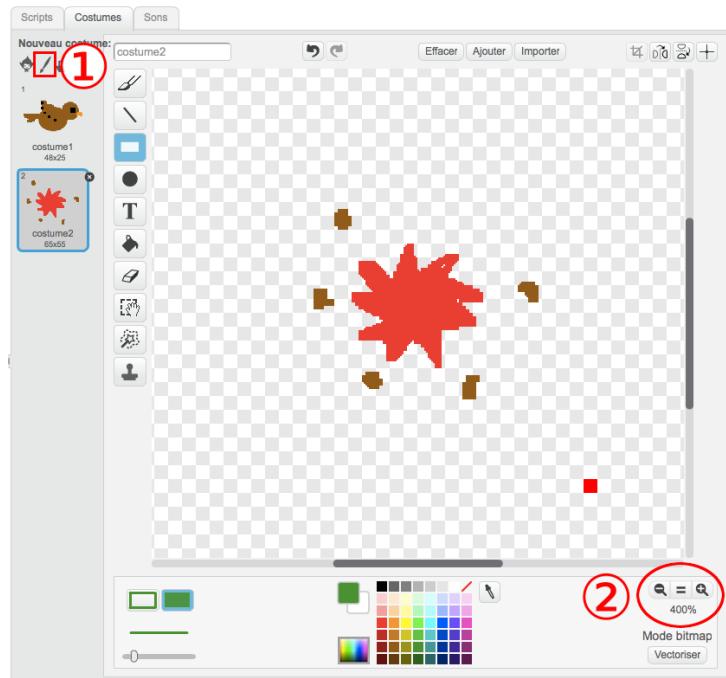
Commencer par supprimer le sprite par défaut en effectuant un clic droit dessus puis en choisissant **Supprimer** (figure à gauche ci-dessous). Créer ensuite un nouveau sprite à dessiner en cliquant sur le pinceau (figure à droite ci-dessous).



La première étape est de créer un premier costume pour notre sprite. Pour cela, dessiner un oiseau (ou toute autre forme !) à l'aide des outils de dessin (ici l'outil ellipse et l'outil pinceau). **Attention !** L'oiseau doit être petit : observer que la zone de dessin a été agrandie à 400 % à l'aide de l'outil zoom (① sur la figure ci-dessous).



La seconde étape consiste à créer un second costume pour notre sprite. Ce costume sera utilisé lorsque l'oiseau heurte un tube. Pour créer un nouveau costume, cliquer sur le pinceau (① sur la figure ci-dessous), positionner le zoom sur 400 % puis dessiner le costume souhaité.



À partir de maintenant il est possible de changer le costume du sprite grâce au bloc suivant, disponible dans les blocs **Apparence** :



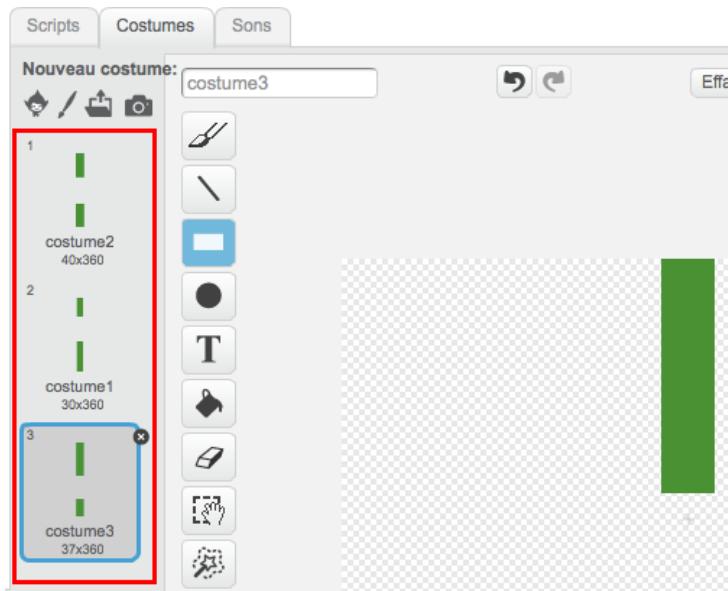
*Remarque* : il est possible de modifier le nom des costumes ou des arrière-plans en utilisant la zone de saisie en haut de la zone de dessin. Il est également possible de changer le nom des sprites en sélectionnant le sprite puis en cliquant sur le ⓘ (figure ci-dessous à gauche). Le nouveau nom peut alors être entré dans la zone de saisie (figure à droite ci-dessous).



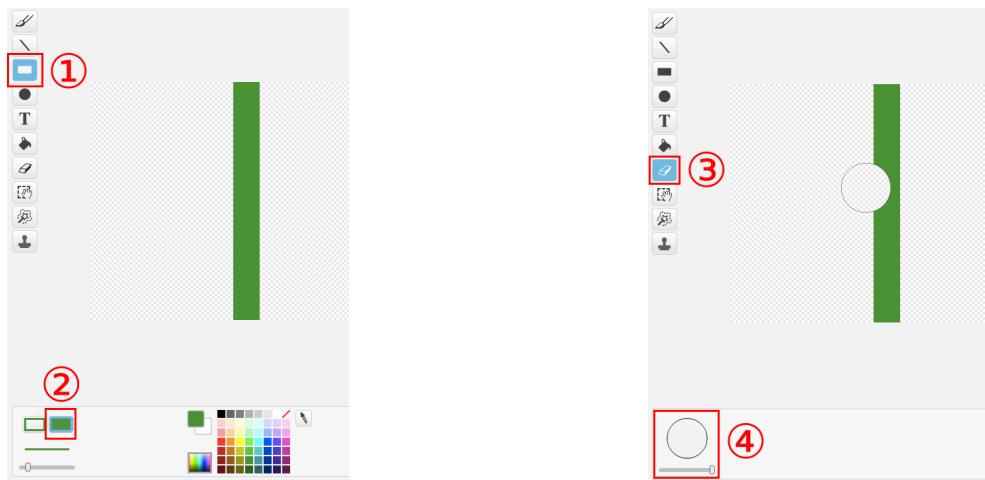
### Construction des costumes du sprite Tuyau

*Remarque* : si vous avez changé la valeur du zoom de la zone de dessin lors de la création du costume du sprite à l'étape précédente, ne pas oublier de la ramener à 100 % avant de poursuivre ! Dans le cas contraire, vos tuyaux apparaîtront trop petits à l'écran pendant le jeu.

Le but est de créer un nouveau sprite avec trois costumes différents correspondant à trois tuyaux différents comme montré sur la figure ci-dessous.



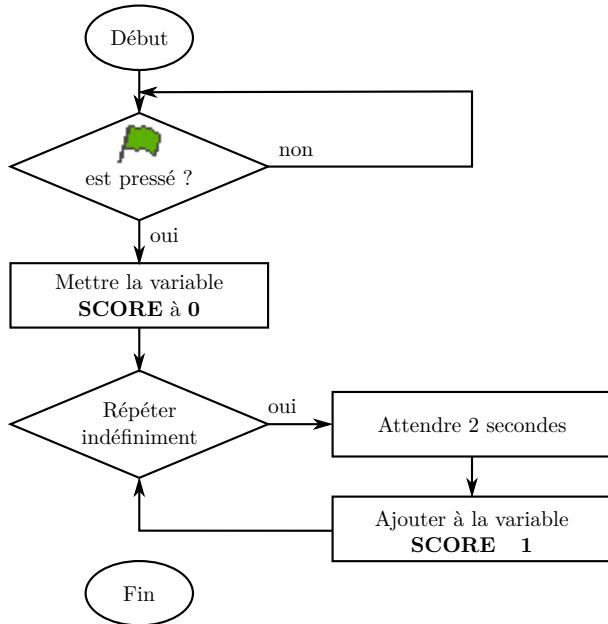
Pour cela, créer un nouveau sprite à dessiner puis dessiner trois costumes différents. Pour dessiner un tuyau, le plus simple est de dessiner un grand rectangle rempli de couleur à l'aide de l'outil rectangle ① en choisissant le remplissage des formes créées ② (voir figure ci-dessous à gauche). À l'aide de l'outil gomme ③ dont on règle la taille ④, couper en deux le tuyau (figure à droite ci-dessous).



### Code associé à la scène

Le code associé à la scène est très simple. C'est lui qui permet de compter les points gagné par le joueur. Le joueur gagne 1 point toutes les 2 secondes de jeu : pour gagner beaucoup de points, il faut donc rester en vie le plus longtemps possible !

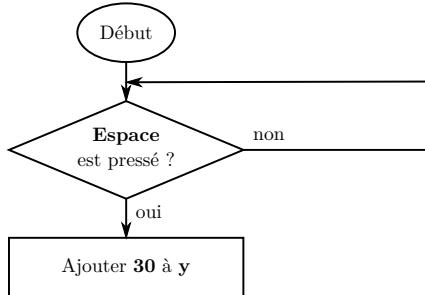
Il faut tout d'abord commencer par créer une variable **SCORE**. Pour cela, cliquer sur les blocs *Données* puis sur le bouton *Créer une variable*. Lui donner le nom **SCORE**. Le diagramme *flowchart* suivant permet de construire le code associé à la scène :



On remarquera que la fin n'est jamais atteinte puisqu'une boucle *Répéter indéfiniment* est présente (boucle infinie). Le comptage des points prendra fin lorsque le bloc **Stop tout** sera appelé (voir plus bas).

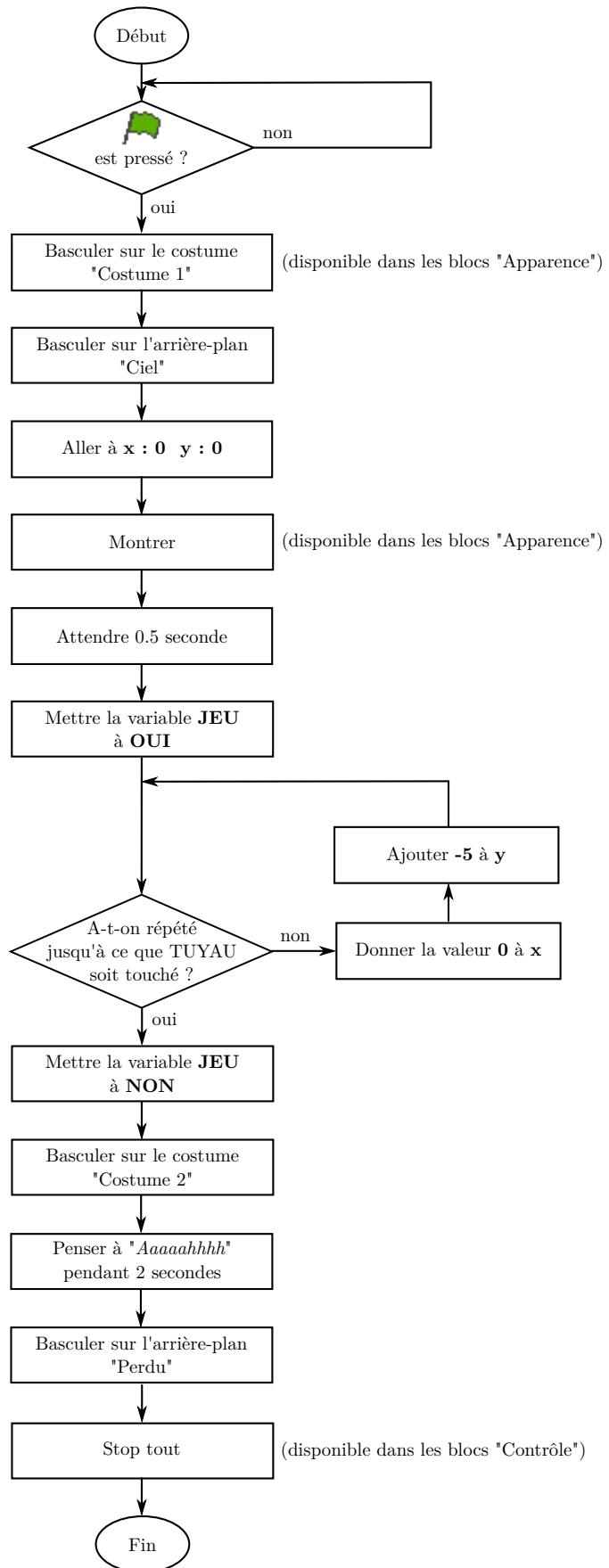
### Codes associés au sprite Oiseau

Deux codes sont associés au sprite **Oiseau**. Le premier est très simple : il permet de faire « remonter » l'oiseau lorsque la touche **Espace** est pressée. Son diagramme *flowchart* est donné ci-dessous.



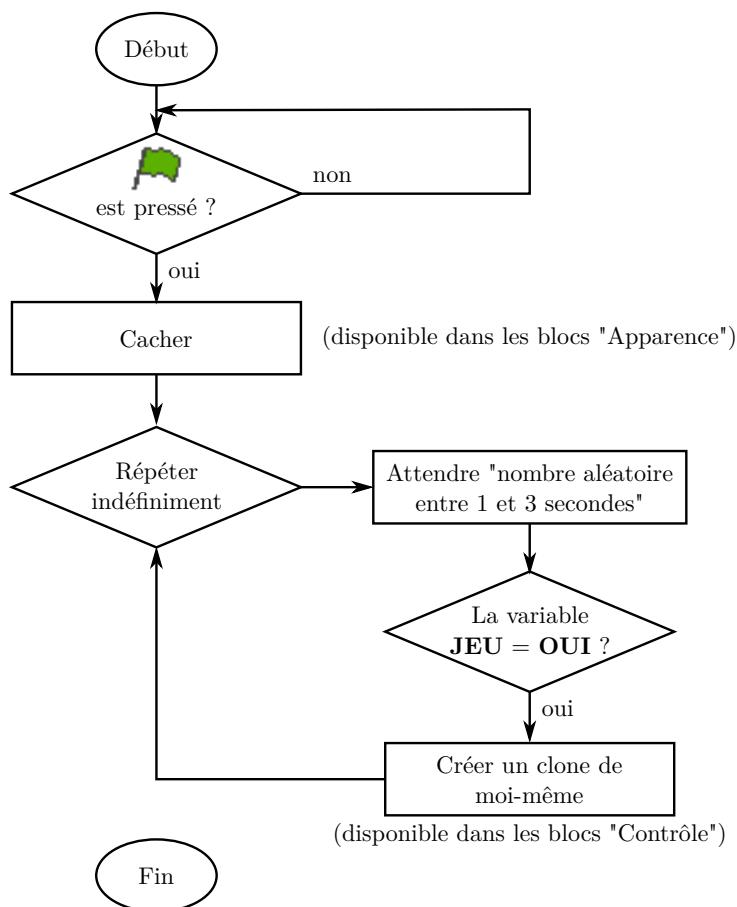
Le second code (voir page suivante) correspond au cœur du jeu : il permet de faire « tomber » l'oiseau en permanence et il laisse tourner le jeu jusqu'à ce que l'oiseau touche un tuyau. Une fois un tuyau touché, le code arrête le jeu, change le costume associé au sprite **Oiseau** afin de montrer qu'un obstacle est heurté et change l'arrière-plan associé

à la scène pour afficher le message «*Perdu !*». Le diagramme *flowchart* de ce code est donné ci-dessous. Attention, il faut créer une variable JEU qui prend la valeur OUI quand la partie est en cours et la valeur NON quand la partie est perdue.



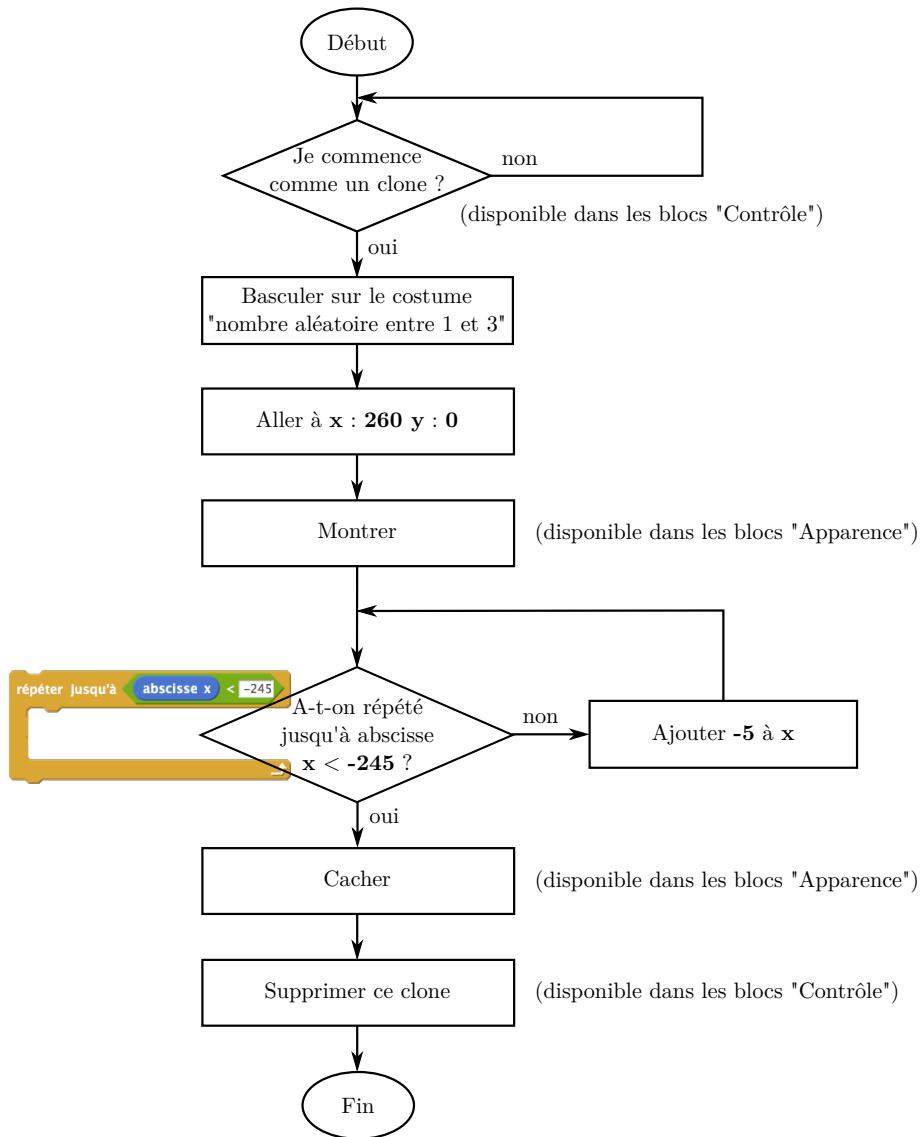
### Codes associés au sprite Tuyau

Le premier code à construire permet de créer un clone de tuyau, choisi au hasard parmi les trois costumes du sprite Tuyau. Comme il faut beaucoup de tuyaux en même temps à l'écran, on va créer des *clones* du sprite qui existeront le temps de traverser l'écran de jeu. Les clones sont créés tant que la variable JEU a pour valeur OUI. Le diagramme *flowchart* correspondant est donné ci-dessous :



On remarquera que là encore la fin n'est jamais atteinte (utilisation d'une boucle infinie).

Le second code permet de contrôler un clone du sprite Tuyau. Ce code est appelé lorsqu'un clone est créé (bloc **Quand je commence comme un clone**). Il permet de positionner le clone à droite de l'écran, puis de le faire avancer petit à petit jusqu'à l'autre bord de l'écran. Arrivé à ce point, le clone est caché puis détruit. Voilà le diagramme *flowchart* correspondant :



Il est possible de modifier la vitesse avec laquelle les tuyaux traversent l'écran. Au choix, vous pouvez :

- modifier la valeur  $-5$  par une autre ;
- utiliser une valeur aléatoire en lieu et place de la valeur  $-5$  ;
- créer une variable vitesse qui change de valeur en fonction de la variable score, ce qui permet de faire avancer les tuyaux de plus en plus vite et rend le jeu plus difficile au fur et à mesure que le temps passe.

### 4.3 Aide pour la séance 3

Tous les codes qui seront écrits concernent le sprite : il faut donc sélectionner celui-ci avant de commencer à programmer.

La code principal (code qui va appeler tous les autres) sera celui de la figure ci-dessous. Il contient des nouveaux blocs que vous allez créer (inutile donc de les chercher tant que vous ne les avez pas créé !).



*Remarque :* le bloc **cacher** permet de faire disparaître le sprite.

La figure ci-dessous présente les trois nouveaux blocs à créer.



#### 4.4 Création du bloc **entrée des données**

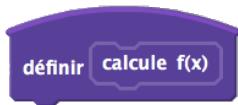


Ce nouveau bloc doit permettre de demander à l'utilisateur d'entrer une valeur pour la variable  $a$  et une valeur pour la variable  $b$ . À chaque fois la réponse de l'utilisateur est stockée dans la variable correspondante.

*Aide :* Lorsqu'on utilise le bloc **demander ... et attendre**, la réponse de l'utilisateur est stockée dans la variable **réponse** (figure ci-dessous).



#### 4.5 Création du bloc **calcule f(x)**



Ce nouveau bloc est très simple : il range la valeur  $a \times x + b$  dans la variable  $y$ .

## 4.6 Creation du bloc **trace f(x)**



Ce nouveau bloc permet de tracer tous les points de la fonction, c'est--dire tous les points de coordonnee  $(x ; y)$  o u  $y = a \times x + b$ . Il faut commencer le trace avec la valeur  $x = -240$  (bord gauche de l'cran), puis augmenter la valeur de  $x$  de 10 en 10 (inutile de tracer tous les points).