

### 1

### Préparation de l'espace de travail

Pour cette activité, nous n'utiliserons pas le lutin fourni par défaut. Nous pouvons donc le supprimer (voir ci-contre).

En revanche, nous aurons besoin d'un lutin possédant quatre costumes distincts (le nom du costume est inscrit sous les dessins du lutin) :



face



dos



gauche

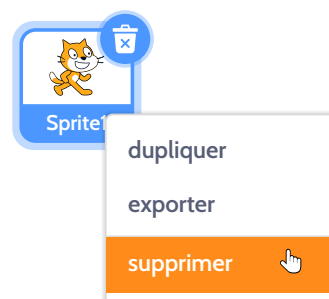


droite

Importer ce lutin en suivant la manipulation décrite ci-dessous.

#### Pour supprimer un lutin :

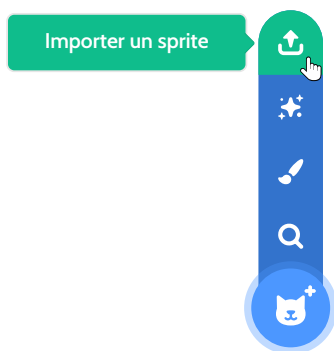
Image 1



Dans la zone des propriétés des lutins, effectuer un clic avec le bouton droit de la souris sur le lutin à supprimer et sélectionner le menu supprimer .

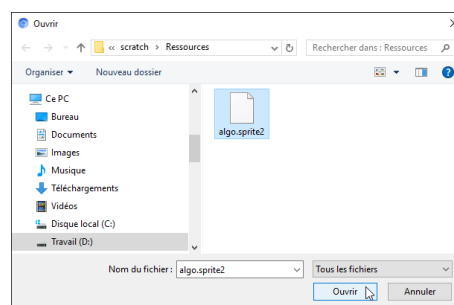
#### Pour importer un lutin :

Image 2



Cliquer sur le bouton  (importer un sprite ).

Image 3



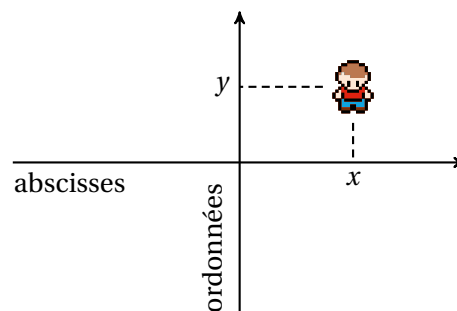
Dans le dossier Ressources, sélectionner le fichier algo.sprite2.

### 2

### Déplacement d'un lutin

Il s'agit de faire en sorte que le lutin se déplace dans la scène lorsque l'une des touches fléchées du clavier est pressée (évidemment, le lutin doit se déplacer dans la direction qui correspond à la touche choisie).

Le principe consiste à modifier les coordonnées du lutin en ajoutant ou en retirant une certaine valeur à l'abscisse ou à l'ordonnée. Pour ce faire, on utilisera les blocs **ajouter 10 à x** et **ajouter 10 à y**.



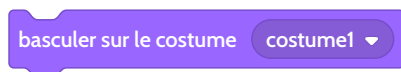
Plutôt que d'ajouter ou de retrancher une valeur définitivement fixée aux coordonnées du lutin, il semble plus judicieux d'utiliser une variable, nommée, par exemple, **pas** à laquelle on pourra attribuer la valeur 5 en début de programme.



En mathématiques, soustraire consiste à ajouter l'opposé. Dans Scratch, comme il n'existe pas d'instruction **enlever** à  $x$ , on utilise cette propriété pour retrancher **pas** à  $x$  ou à  $y$ .



Réaliser un programme qui permet de déplacer le lutin à travers la scène. Le costume du lutin doit changer en fonction de la direction adoptée. On pourra utiliser, en particulier, les blocs suivants :



Initialiser la variable **pas** à 5

Répéter indéfiniment :

Si la touche ← est pressée alors :

Basculer sur le costume *Gauche*

Retirer **pas** à  $x$

Fin du test

⋮

Fin de la boucle

Algorithme

### 3

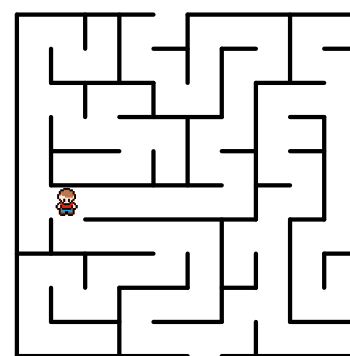
## Déplacement dans un labyrinthe

Nous désirons que le lutin se déplace à l'intérieur d'un labyrinthe. Pour ce faire, importer le lutin *labyrinthe.sprite2* et positionner le labyrinthe au centre de la scène.

Malheureusement, comme vous pouvez le constater, le lutin *algo* se déplace « par-dessus » le labyrinthe.

Pour que le lutin *algo* ne franchisse pas les murs du labyrinthe, le principe consiste à détecter une éventuelle collision entre le lutin *algo* et le lutin *labyrinthe* quand une touche fléchée a été pressée. En cas de collision, on rétablit immédiatement le lutin *algo* à sa position précédente.

Image 4



Répéter indéfiniment :

Mettre **ancien x** à **abscisse x**

Mettre **ancien y** à **ordonnée y**

⋮

Si *labyrinthe* est touché alors :

    Aller à ( **ancien x** ; **ancien y** )

Fin du test

Fin de la boucle

Algorithme

- 1) Créer deux variables, **ancien x** et **ancien y**, destinées à stocker les coordonnées du lutin *algo* avant déplacement.
- 2) Dans le script du lutin *algo*, insérer une condition permettant de tester une collision avec le lutin *labyrinthe* et rétablir la position précédente du lutin *algo* dans le cas où le test s'avère positif.

Aide

touche le **pointeur de souris** ?

Ce bloc (que l'on utilise généralement à l'intérieur d'une instruction **Si**) permet de détecter lorsqu'un lutin atteint le pointeur de la souris, touche le bord de la scène ou touche un autre lutin (utiliser la liste déroulante pour sélectionner le type de détection souhaité).

Aide

aller à x: **0** et y: **0**

Ce bloc permet de déplacer un lutin aux coordonnées indiquées en paramètres (*x* pour l'abscisse et *y* pour l'ordonnée).

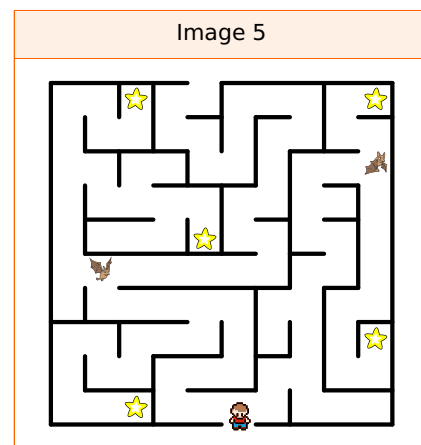


En fonction de la position initiale du lutin *algo* et selon la valeur attribuée à la variable **pas**, il se peut que le lutin se trouve « coincé » contre une paroi du labyrinthe. Dans ce cas, il convient de modifier la position initiale du lutin (utiliser un bloc **aller à x: 0 et y: 0** en début de script) et/ou de diminuer légèrement la valeur de la variable **pas**.

## 4

## Création d'un jeu

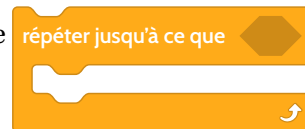
Il est désormais possible d'imaginer différents jeux qui prennent appui sur la traversée d'un labyrinthe par un lutin. Par exemple, le lutin peut avoir pour objectif de récupérer, le plus rapidement possible, des étoiles disséminées dans le labyrinthe et en évitant d'éventuels ennemis.



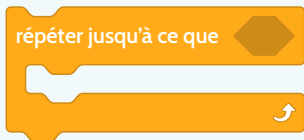
### 1 La sortie du labyrinthe

En premier lieu, il s'avère nécessaire de détecter le moment où le lutin *algo* sort du labyrinthe afin d'arrêter le script.

Il s'agit donc de remplacer la boucle **répéter indéfiniment** par une boucle du type **répéter jusqu'à ce que**.



répéter jusqu'à ce que



Ce bloc prend en paramètre une condition : tant que la condition n'est pas vérifiée, les instructions situées à l'intérieur du bloc sont répétées.

Pour détecter la sortie du labyrinthe par le lutin *algo*, il suffit de vérifier le moment où son ordonnée devient plus grande qu'une certaine valeur (à déterminer en fonction de la position du lutin *labyrinthe* sur la scène et en supposant que l'entrée du labyrinthe soit « en bas » et la sortie « en haut »).

De surcroît, le lutin *algo* ne doit être autorisé à sortir du labyrinthe que lorsque toutes les étoiles ont été collectées. Il est donc nécessaire de créer une variable **étoiles** permettant de compter le nombre d'étoiles qu'il reste à ramasser.

ordonnée y

Ce bloc renvoie l'ordonnée du lutin dans lequel il est utilisé.



Ce bloc permet de juxtaposer deux conditions qui doivent être vérifiées toutes les deux.

Pour indiquer que le jeu est terminé, on peut, par exemple, modifier la couleur de la scène, ce qui revient à ajouter un arrière-plan coloré à la scène.

### Pour créer un nouvel arrière-plan :

Image 6

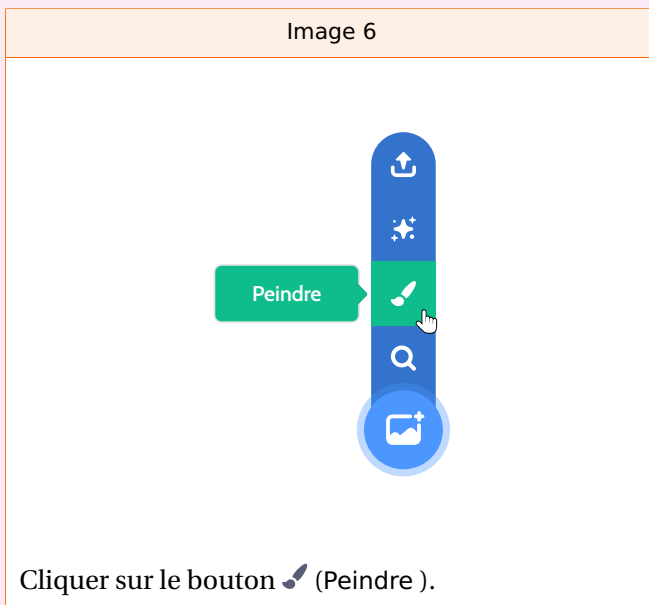
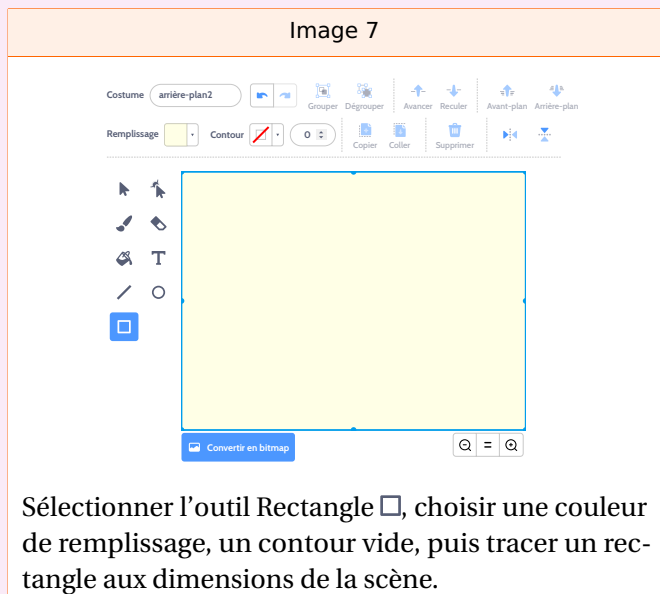


Image 7



basculer sur l'arrière-plan

arrière-plan1 ▼

Ce bloc permet de sélectionner l'arrière-plan de la scène.

## 2 Les étoiles

Commençons par ajouter un nouveau lutin en forme d'étoile en le choisissant dans la bibliothèque.

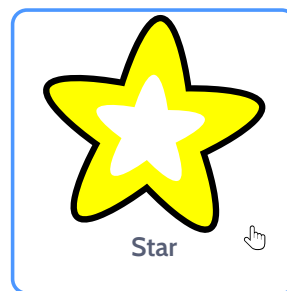
## Pour choisir un lutin dans la bibliothèque :

Image 8




Cliquer sur le bouton  (choisir un sprite).

Image 9



Sélectionner le lutin *Star*.



Il sera nécessaire d'adapter les dimensions du lutin en forme d'étoile aux dimensions du labyrinthe (une taille de  $20 \times 20$  doit convenir). Pour cela, utiliser les poignées de redimensionnement qui apparaissent lorsqu'on clique sur le costume du lutin dans l'onglet  **Costumes**.

Le script associé au lutin en forme d'étoile consiste à garder l'étoile apparente tant que le lutin *algo* n'a pas touché l'étoile. Pour cela, on pourra utiliser une instruction

**attendre jusqu'à ce que**

Une fois l'étoile touchée par le lutin *algo*, il faut penser à la cacher et à retirer 1 à la variable **étoiles**.

Algorithme

- Aller en (... ; ...)
- Montrer le lutin
- Attendre jusqu'à *algo* touché
- Retirer 1 à **étoiles**
- Cacher

Lorsque le script associé au lutin en forme d'étoile est fonctionnel, dupliquer plusieurs fois ce lutin et positionner les différentes étoiles dans le labyrinthe.

### 3 Le chronomètre

Lorsque le programme est lancé, le chronomètre doit se déclencher.

Aide

**réinitialiser le chronomètre**

Ce bloc permet de remettre le chronomètre à zéro.

Lorsque le lutin *algo* sort du labyrinthe après avoir collecté toutes les étoiles, le temps écoulé depuis le début du jeu doit être affiché. Il faut donc créer un nouveau lutin (son costume peut être vide) qui laissera affichée la durée écoulée depuis le début de la partie (on enverra un message au lutin pour cela).

Aide

**dire** Bonjour

Ce bloc permet d'afficher un message dans une bulle.

Aide

**chronomètre**

Ce bloc se comporte comme une variable qui contient la valeur du chronomètre, exprimée en secondes (au millième près).

Pour rendre l'affichage de la durée écoulée plus lisible, il est possible d'arrondir la valeur du chronomètre à la seconde près.

Aide

**arrondi de**

Ce bloc renvoie l'arrondi à l'unité de la valeur qui lui est fournie en paramètre.

## 4 Les chauves-souris

Quelques chauves-souris longent les couloirs du labyrinthe. Si le lutin a lgo a le malheur d'en toucher une, il est automatiquement renvoyé à l'entrée du labyrinthe.



Créer un lutin chauve-souris soit en le choisissant dans la bibliothèque soit en important le sprite bat2.sprite3 puis, adapter sa taille à celle du labyrinthe.

Pour que la chauve-souris longe un couloir, il suffit de la faire glisser indéfiniment d'une extrémité à l'autre du couloir.

**Aide**

glisser en  secondes à x:  y:

Ce bloc permet de faire glisser un lutin (en effectuant une translation) jusqu'à la position définie par les paramètres  $x$  et  $y$  (les coordonnées du point d'arrivée).

Puisque le lutin chauve-souris de la bibliothèque dispose de plusieurs costumes, il est possible de donner l'illusion que la chauve-souris bat des ailes en alternant les costumes.



Si la chauve-souris longe un couloir horizontal, il est nécessaire de prévoir des costumes obtenus par symétrie selon un axe vertical lorsque celle-ci se déplace de la droite vers la gauche.

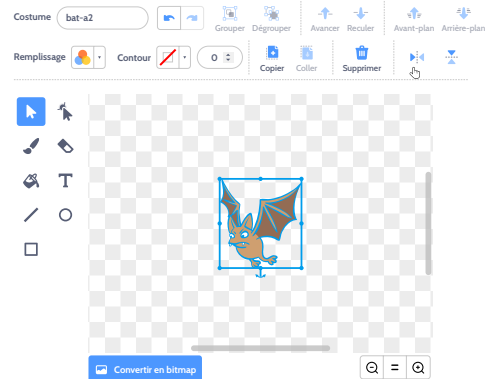
### Pour dupliquer un costume et le modifier :


Image 10



Dans l'onglet  **Costumes**, effectuer un clic avec le bouton droit de la souris sur le costume à dupliquer et sélectionner le menu dupliquer.

Image 11



Sélectionner le costume dupliqué et effectuer les modifications (cliquer sur le bouton Retournement horizontal  pour obtenir le symétrique du costume selon un axe vertical).