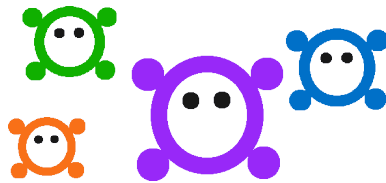


# Aquaakbot et les bestioles





# **SqueakBot et les bestioles**

Une aventure pour découvrir pas à pas SqueakBot

Planète Sciences

19 juin 2012



# Table des matières

<b>1</b>	<b>Bienvenue !</b>	<b>5</b>
<b>2</b>	<b>Prendre en main SqueakBot</b>	<b>7</b>
2.1	Le monde des objets . . . . .	7
2.2	La palette des artistes . . . . .	9
2.3	Notre premier programme . . . . .	12
2.4	Voir les trajectoires . . . . .	16
2.5	Un grain de folie ! . . . . .	17
<b>3</b>	<b>La danse des scripts</b>	<b>21</b>
3.1	Jouons avec les briques . . . . .	21
3.2	Actions, scripts, paramètres... . . . . .	24
<b>4</b>	<b>Les bestioles se multiplient</b>	<b>31</b>
4.1	Sauvegarder son projet . . . . .	31
4.2	Les bestioles font des bébés . . . . .	32
4.3	Quand je serais grand... introduisons les variables . . . . .	36
4.4	Ça grandit, ça grandit . . . . .	40
4.5	Bon appétit ! . . . . .	41
4.5.1	Des petits paquets de nourriture qui se font grignoter . . . . .	41
4.5.2	Les fruits disparaissent petit à petit . . . . .	44
<b>5</b>	<b>Regardons vivre notre monde merveilleux !</b>	<b>47</b>
5.1	Préparons la simulation . . . . .	47
5.2	Le centre de contrôle . . . . .	47
5.3	L'évolution de la population . . . . .	47
<b>6</b>	<b>Pour finir...</b>	<b>49</b>
6.1	Conclusion . . . . .	49
6.2	Pour aller plus loin . . . . .	49
<b>7</b>	<b>Annexes</b>	<b>51</b>
7.1	Installer SqueakBot . . . . .	51
7.2	Astuces et problèmes courants . . . . .	51
7.3	Contacts . . . . .	51
	<b>Glossaire</b>	<b>53</b>

## Aux adultes

L'une des forces et, en réalité, la principale raison d'être de SqueakBot, c'est qu'il permet aux enfants, jeunes et moins jeunes, d'expérimenter autant qu'ils le souhaitent, sans jamais se demander si ce qu'il font est juste ou faux, bon ou pas bon.

Les pages qui suivent proposent une découverte approfondie de SqueakBot dans le cadre de la réalisation d'un projet relativement complexe, qui devrait permettre à chacun de devenir vraiment à l'aise à l'intérieur de l'interface parfois un peu déroutante de SqueakBot, tout en mesurant l'étendue des possibilités qu'il ouvre.

Cependant ce tutoriel ne devrait sans doute pas être utilisé tel quel pour mener un projet avec les enfants. Il ne s'agit en particulier pas de remettre ce livret dans les mains des enfants et de leur dire « Amusez-vous bien ! » : à suivre pas à pas le projet, ils passeraient à côté de cet esprit d'expérimentation, ce vent de créativité que nous voulons développer.

Nous ne pouvons donc que vous encourager à vous servir de ce livre comme un outil pour votre propre pratique, comme une référence, comme une source d'idée, et ponctuellement, comme un guide pour les enfants.

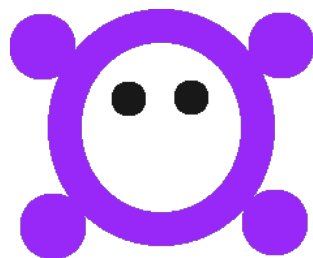
Bonne découverte !

# Chapitre 1

## Bienvenue !

Bienvenue ! Dans les pages qui suivent, nous allons ensemble, étape par étape, découvrir SqueakBot. C'est un univers un peu déroutant au début, mais c'est un vrai univers : dans SqueakBot, toutes les histoires peuvent être racontées ! L'objectif, c'est qu'il soit facile de donner vie à toutes les idées qui nous passent par la tête, que ce soit des manchots qui sautent sur le dos d'une baleine, des fleurs qui dansent, ou des bestioles bizarroïdes qui se multiplient.

Des bestioles bizarroïdes qui se multiplient ?



Des bestioles comme ça, peut-être ?

FiXme Fatal!]FiXme Fatal!<sup>1</sup>

Qui se multiplie comme ça ?

...et pourquoi pas, tiens ! Allez, lançons nous dans ce projet : Faire, en partant de zéro, une petite simulation d'écosystème, avec des bestioles qui naissent, qui se reproduisent, qui se nourrissent et qui meurent. Voyons ce que ça peut donner.

---

1. FiXme Fatal: Insérer une capture avec plein de bestioles

**Bienvenue !**

---



# Chapitre 2

## Prendre en main SqueakBot

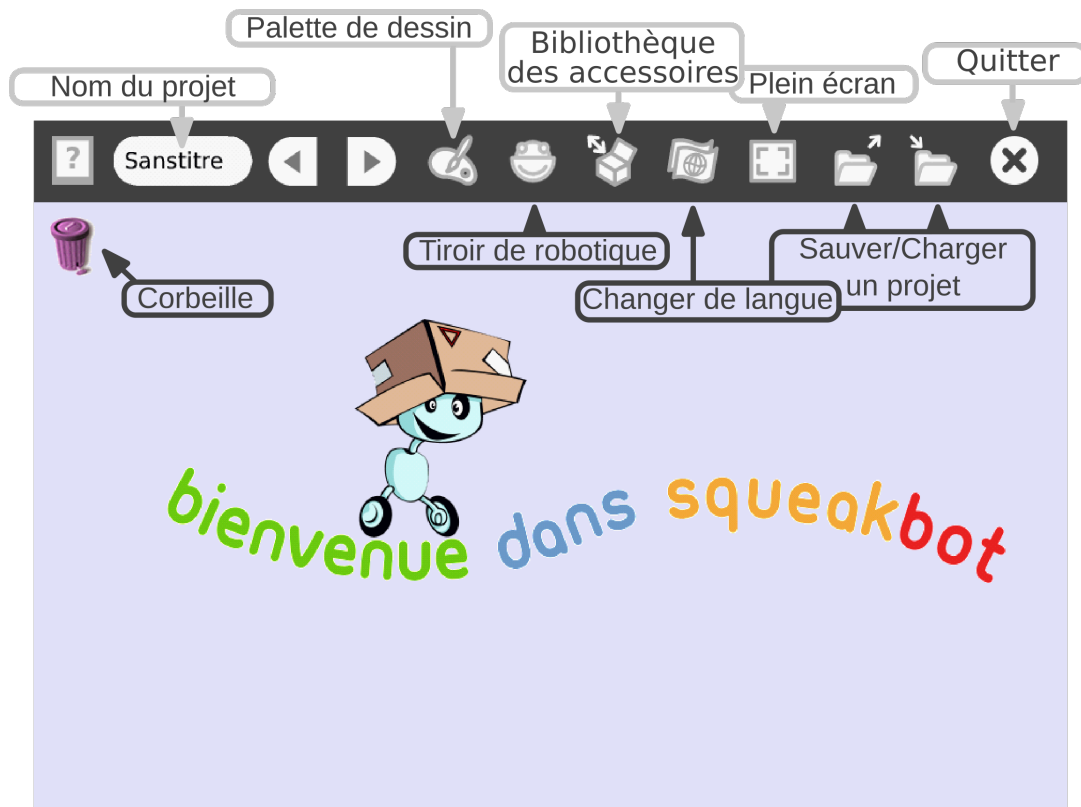
### 2.1 Le monde des objets

Au démarrage de SqueakBot, vous devriez tomber sur quelque chose qui ressemble de près à l'image ci-dessous :



Voici les principales fonctions des différents boutons que l'on voit dans l'interface :

## Prendre en main SqueakBot

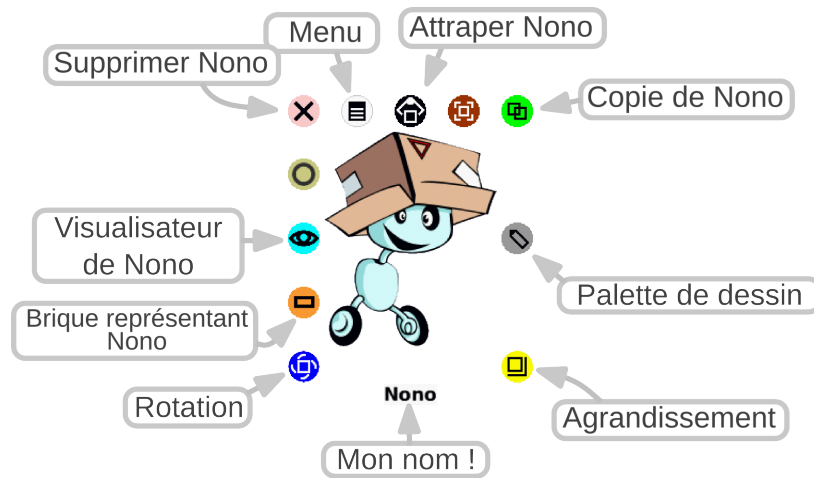



Faisons un clic droit sur le robot Minibot :



Toute une série de petites icônes apparaissent. C'est ce qu'on appelle le **halo**. C'est elles qui vont nous permettre de manipuler nos objets.


Voici le rôle des principales d'entre elles :



En cliquant (et en restant appuyé) sur l'icône  **Pivoter**, on peut faire tourner l'objet :



Nous verrons plus tard à quoi servent les autres icônes et comment modifier le cap et le centre de rotation d'un objet.

En attendant, vous pouvez essayer de redimensionner Minibot avec l'icône  **Redimensionner**.

Avant de continuer, faisons un peu de place : cliquons sur les objets qui sont sur le bureau pour les saisir, et glissons le texte et Minibot sur la corbeille. Si nécessaire, on peut les y récupérer plus tard en double-cliquant sur la corbeille.

Voilà, avec ces premiers pas, nous avons vu ce qu'était un objet dans SqueakBot, et nous avons découvert son halo, et les manipulations de base qui lui sont associées. Nous sommes en bonne route vers la programmation, mais avant cela, voyons comment créer nos propres objets en les dessinant.

## 2.2 La palette des artistes

Dans SqueakBot, la plupart des objets sont créés à partir de dessin. Pas de limite à notre imagination !

Pour démarrer un dessin, il suffit de cliquer sur l'icône de la palette, en haut de l'écran :


## Prendre en main SqueakBot

---

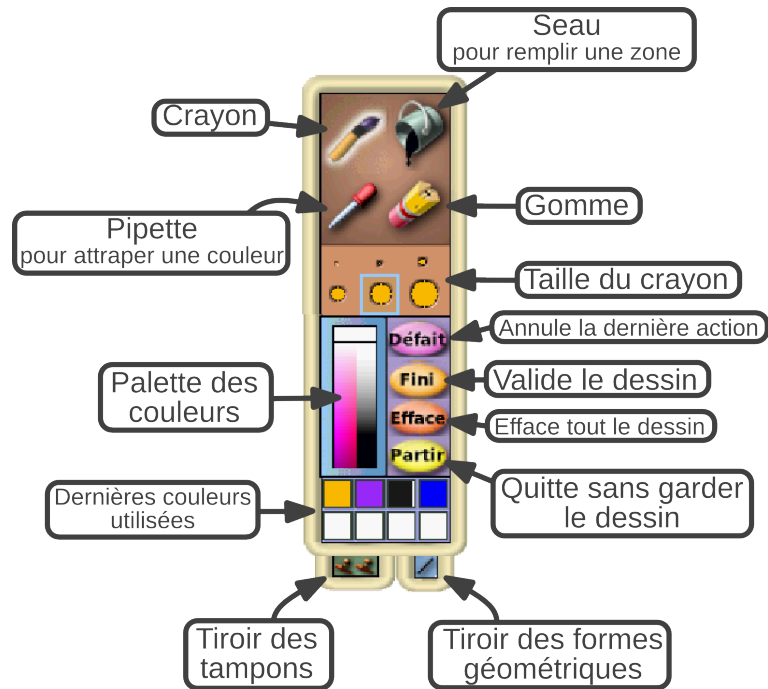


Ceci ouvre le mode dessin :

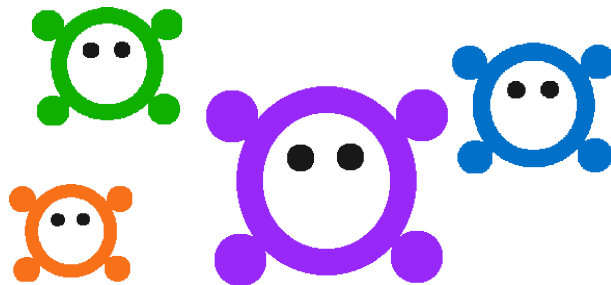


Au centre de l'écran apparaît une zone blanche : c'est le **calque de dessin**. Tout le dessin doit tenir dedans, et s'il est trop petit, on peut le redimensionner comme tous les objets, en faisant un clic droit dessus, puis en utilisant l'icône .

L'illustration ci-dessous présente les différentes fonctionnalités de la palette de dessin :



Avec ça, ça ne devrait pas poser de problèmes pour dessiner une bestiole ! (pense à utiliser les formes géométriques qui sont dans le tiroir en bas à droite).



Une fois notre dessin terminé, nous pouvons cliquer sur **Fin** pour quitter le mode dessin : ça y est, nous venons de créer un nouvel objet.

Donnons lui un nom en faisant un clic droit dessus, puis un clic gauche sur le mot **Dessin**. Nous pouvons maintenant taper le nom que nous voulons, suivi de *Entrée*.



Nous sommes maintenant prêt pour la programmation.

## Prendre en main SqueakBot



En cliquant sur le bouton



SqueakBot passe en plein écran, ce qui est pratique pour gagner un peu de place !


## 2.3 Notre premier programme

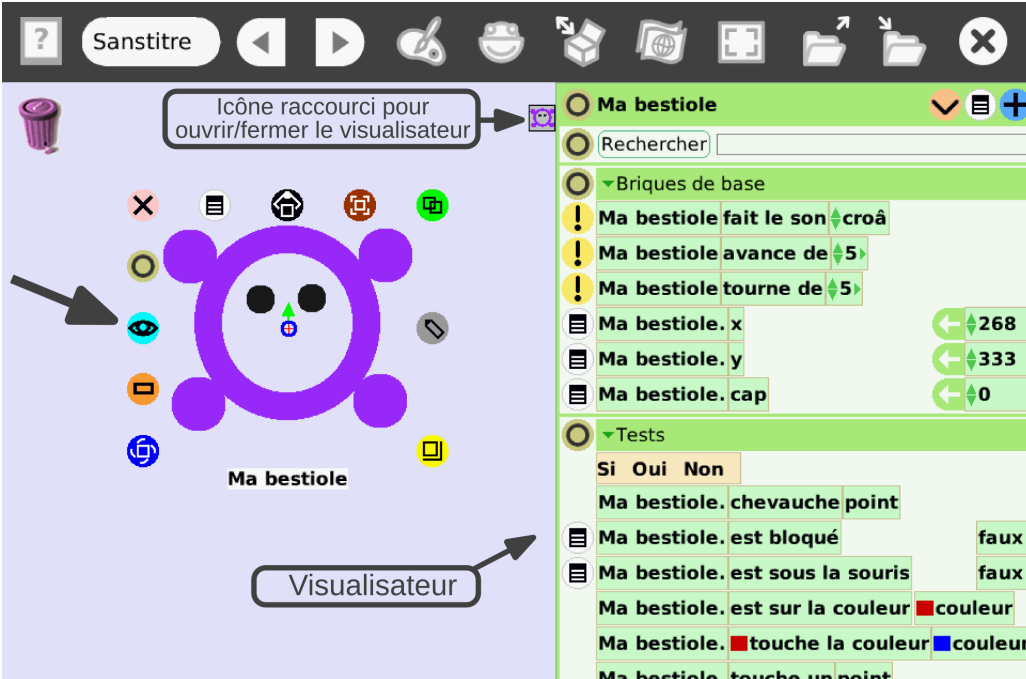
Nous avons fait connaissance avec les objets. Ils sont encore bien statiques... Essayons maintenant de leur donner vie. Leur donner vie, ça va être, pour nous, associer à nos objets des **programmes** qui vont par exemple les faire bouger ou les modifier.



Dans SqueakBot, les programmes sont intimement liés aux objets. Chaque objet a ses propres programmes. C'est pour cela que **Smalltalk**, le langage de programmation qui est derrière SqueakBot est appelé **langage orienté objet**.

Le point de départ pour tous les programmes (ou **script** comme nous allons les appeler dans SqueakBot), c'est le **visualisateur** chaque objet.

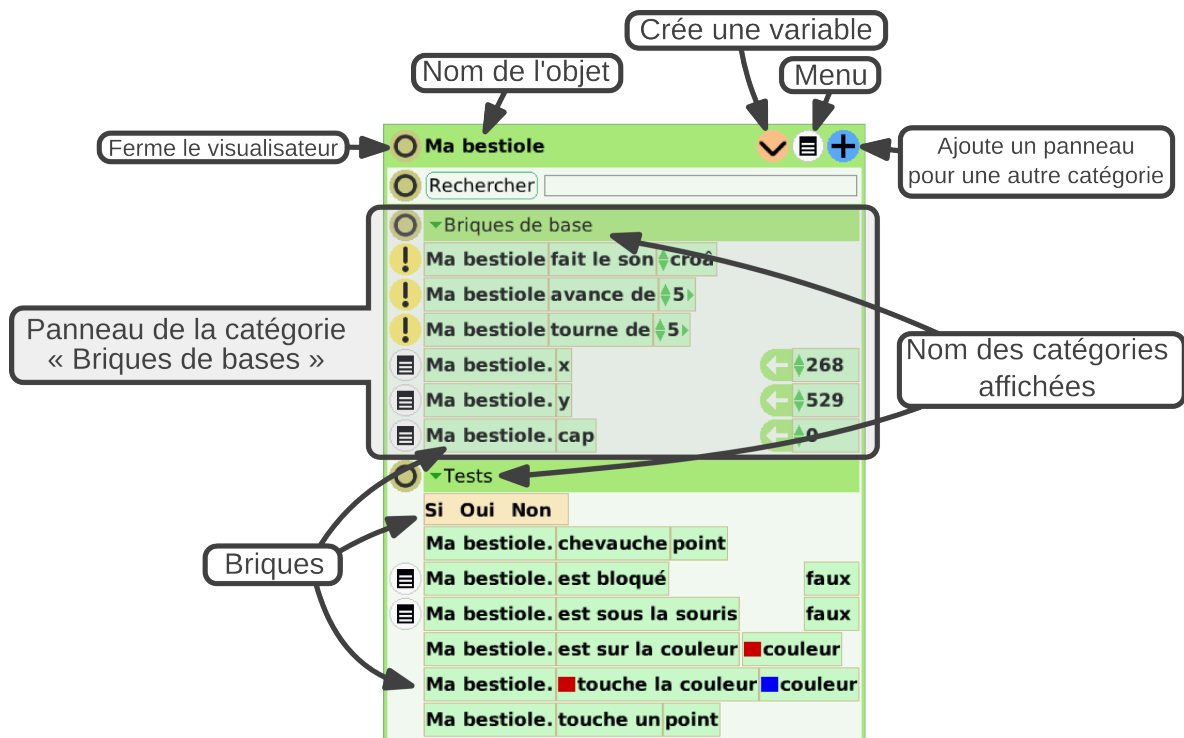
Pour ouvrir le visualisateur, il suffit de cliquer sur l'icône  « œil » du halo de notre objet :



The screenshot shows the SqueakBot interface. At the top, there is a toolbar with various icons. Below it, a purple trash can icon is visible. The main workspace contains a purple bear-like object labeled 'Ma bestiole'. A callout box points to a small eye icon in the object's halo, labeled 'icône raccourci pour ouvrir/fermer le visualisateur'. Another callout box points to the 'Ma bestiole' label, labeled 'Visualisateur'. On the right side, a panel displays the object's properties and scripts. The 'Ma bestiole' panel includes a search bar, a 'Briques de base' section with three scripts: 'Ma bestiole fait le son croâ', 'Ma bestiole avance de 5', and 'Ma bestiole tourne de 5'. Below that, there are three sliders for 'Ma bestiole. x' (set to 268), 'Ma bestiole. y' (set to 333), and 'Ma bestiole. cap' (set to 0). The 'Tests' section shows a table with columns 'Si', 'Oui', and 'Non', and several test scripts like 'Ma bestiole. chevauche point', 'Ma bestiole. est bloqué', 'Ma bestiole. est sous la souris', 'Ma bestiole. est sur la couleur', and 'Ma bestiole. touche la couleur'.

Le visualisateur est l'élément d'interface qui à l'air le plus compliqué dans SqueakBot. En fait, son organisation est plutôt simple, mais comme il permet d'accéder à presque toutes les fonctions de programmation des objets, beaucoup de choses sont affichées.

Comme le montre l'illustration ci-dessous, le visualisateur se compose de grandes catégories (dans l'exemple ci-dessous, on voit les catégories **Briques de base** et **Tests**) qui contiennent un certain nombre de briques.

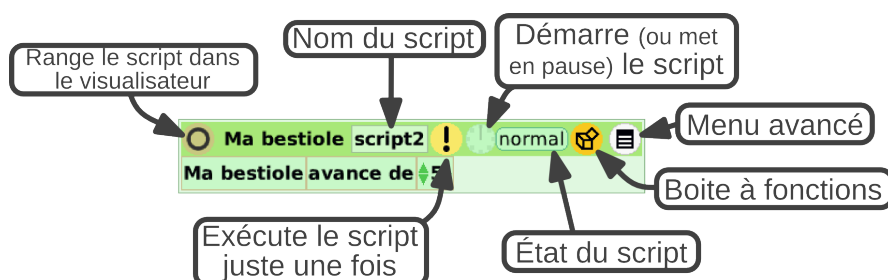



Nous verrons un peu plus tard les différents types de brique. Pour l'instant, essayons de faire notre premier programme, pour faire bouger notre bestiole. Pour cela, faisons un clic gauche sur la brique `avance de 5`, et glissons cette brique sur le bureau, à côté du dessin de l'objet :


## Prendre en main SqueakBot



Étudions un peu plus en détail la nouvelle fenêtre de script qui vient d'apparaître :



Les scripts peuvent soit être exécuté juste une fois en cliquant sur le  point d'exclamation jaune (dans notre cas, notre bestiole va donc avancer de 5 pixels), soit ils peuvent être démarrés pour de bon.

En cliquant sur la petite horloge  à côté de *normal*, on démarre le script. Cela veut dire que, tant qu'on ne le mettra pas en pause en re-cliquant sur l'horloge, SqueakBot va exécuter en boucle le contenu du script. Essayons !

Normalement, la bestiole devrait se mettre à bouger vers le haut : en réalité, elle avance dans la direction de son **cap**. Nous verrons un peu plus tard ce que signifie le **cap**, et comment le modifier.

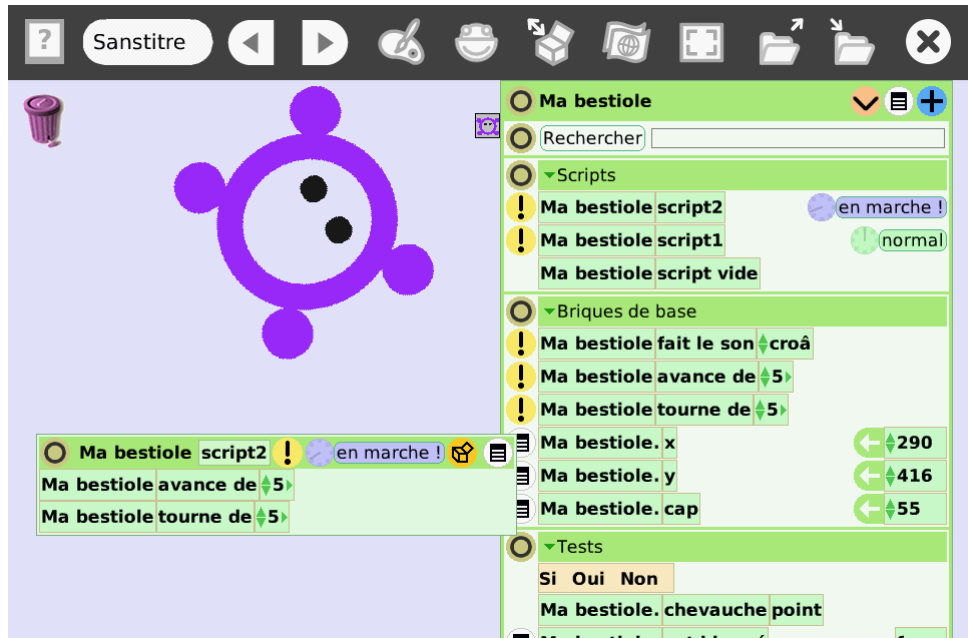


Seules les zones non-transparentes d'un objet peuvent être « saisies » : avec les enfants, il vaut mieux remplir (avec le seau) les zones transparentes à l'intérieur d'un objet avec une couleur pour qu'ils soient plus facile à attraper.

À n'importe quel moment, même quand le programme est en marche, il est possible de cliquer sur le dessin pour le prendre et le déplacer ailleurs sur le bureau. C'est pratique quand un objet est coincé contre un bord, par exemple.



Essayons maintenant de compléter le script pour que la bestiole tourne en même temps qu'elle avance. Pour cela, prenons la brique **tourne de 5** et déposons-la en-dessous ou au-dessus de la brique **avance de 5**, à l'intérieur du script que nous avons créé.

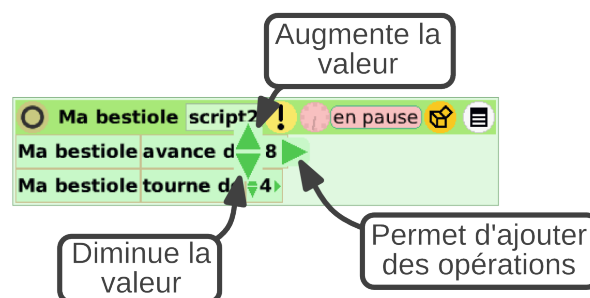


Si notre script était déjà en marche, nous devrions voir notre bestiole commencer à tourner, sinon, mettons-le en route.



Le chemin que décrit notre objet s'appelle sa **trajectoire**. En faisant varier la vitesse et la vitesse de rotation, on modifie cette dernière. On remarque rapidement que cette trajectoire est un... cercle !

Comme nous pouvons le voir, les briques **avance de 5** et **tourne de 5** prennent une vitesse comme option (en programmation, on appelle ça un **paramètre**). Essayons de les faire varier pendant l'exécution du script !

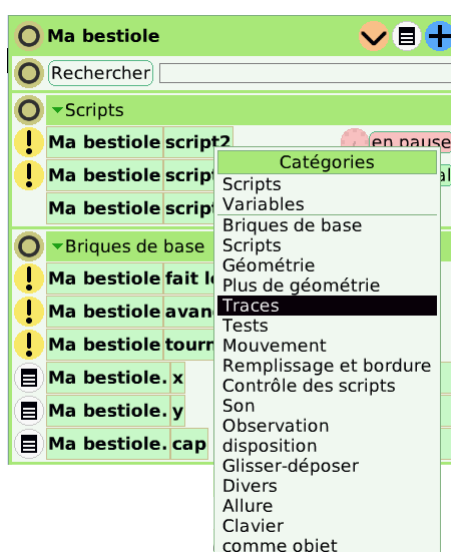


Que penser de cette trajectoire ? Comment pourrait-on faire tourner la bestiole dans l'autre sens ?

### 2.4 Voir les trajectoires

Pour mieux voir la trajectoire de notre bestiole, nous allons lui demander de laisser une trace de son passage.

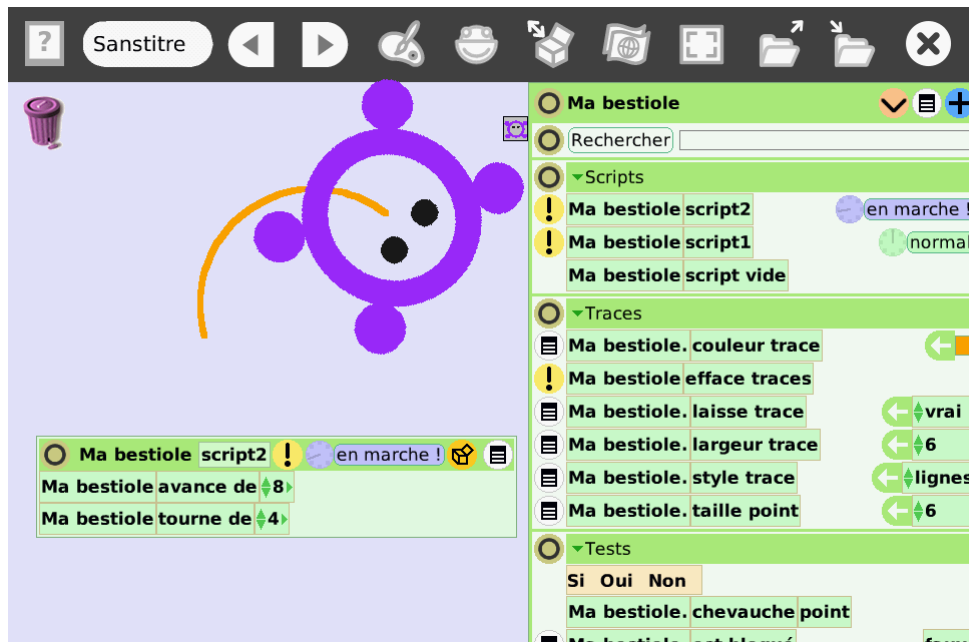
Pour cela, ouvrons la catégorie **Traces**, en cliquant sur le titre d'une des catégories déjà affichées du visualisateur (**Scripts**, **Briques de bases** et **Tests** devraient déjà être visible), puis en sélectionnant *Traces*.




Dans le panneau de la catégorie **Traces**, choisissons une couleur de trace, augmentons un peu la largeur de la trace, puis changeons la valeur de la brique **laisse trace** pour la mettre à *vrai* (en cliquant sur la flèche du bas ou du haut, à côté de *faux*).



Si besoin, on redémarre le script.





Cette fois-ci plus de doute, notre bestiole parcourt un cercle !

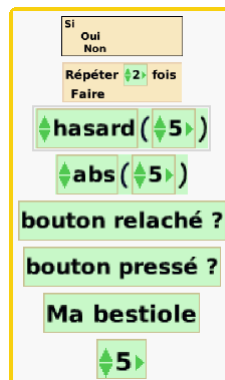
Il est possible d'effacer les traces en cliquant sur le  point d'exclamation devant la brique `efface traces`, et recommencer, en changeant la vitesse de la bestiole. Pour arrêter de laisser des traces, il suffit de remettre à *faux* la brique `laisse trace`.

## 2.5 Un grain de folie !

...les bestioles, en général, n'ont pas une belle trajectoire circulaire comme cela : essayons de mettre un grain de folie là-dedans.

Pour cela, nous allons utiliser un peu de **...hasard**.

Si ce n'est pas fait, mettons notre script en pause en cliquant sur  l'horloge, puis cliquons sur l'icône  **Boîte à fonctions** dans la fenêtre du script.



Sélectionnons la brique `hasard`, et glissons-là **à la place** de la vitesse que nous avions auparavant. Faisons de même pour la vitesse de rotation de la brique `tourne de` :

## Prendre en main SqueakBot



Démarrons le script! Quelque chose à l'air de clocher : notre bestiole tourne sur elle-même, sans vraiment avancer !

Une idée ?

Logique ! la brique **hasard** nous renvoie une valeur au hasard entre 0 et la valeur qu'on lui donne en paramètre (dans l'exemple au-dessus, une valeur entre 0 et 10 pour la vitesse, et entre 0 et 15 pour la vitesse de rotation).

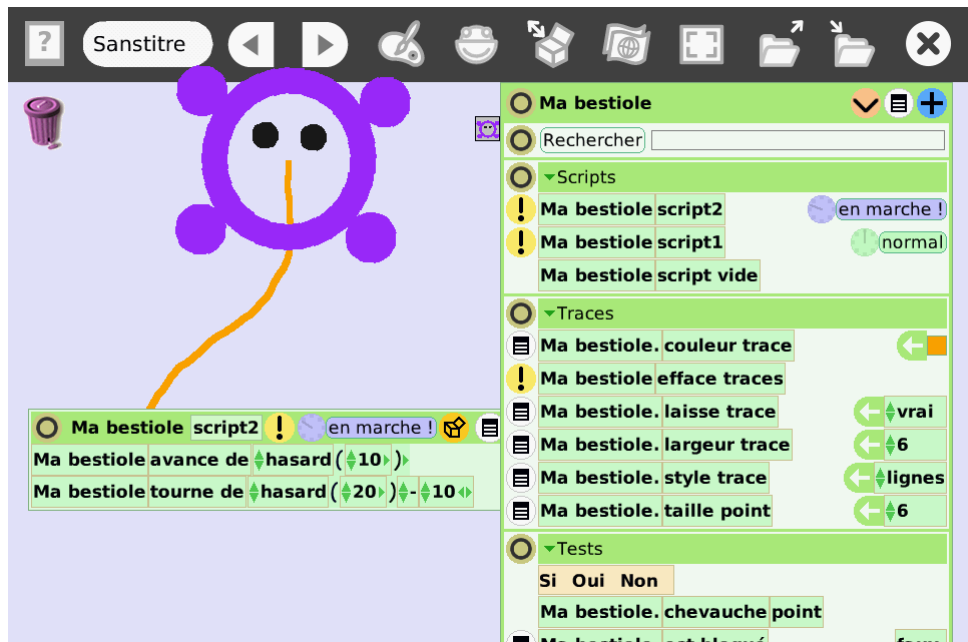
Ça veut dire que la bestiole tourne toujours dans le même sens, vers la droite.

Comment faire pour qu'elle tourne parfois à droite, parfois à gauche, pour avoir ainsi une trajectoire qui ait l'air un peu plus hésitante ?


Si par exemple, on veut que la bestiole tourne de -10 à +10, la solution la plus simple consiste à tirer au sort un nombre entre 0 et 20, et à soustraire 10. Pour faire cela, on peut utiliser la flèche verte au bout de la ligne de **tourne de** : elle permet d'ajouter des opérations.

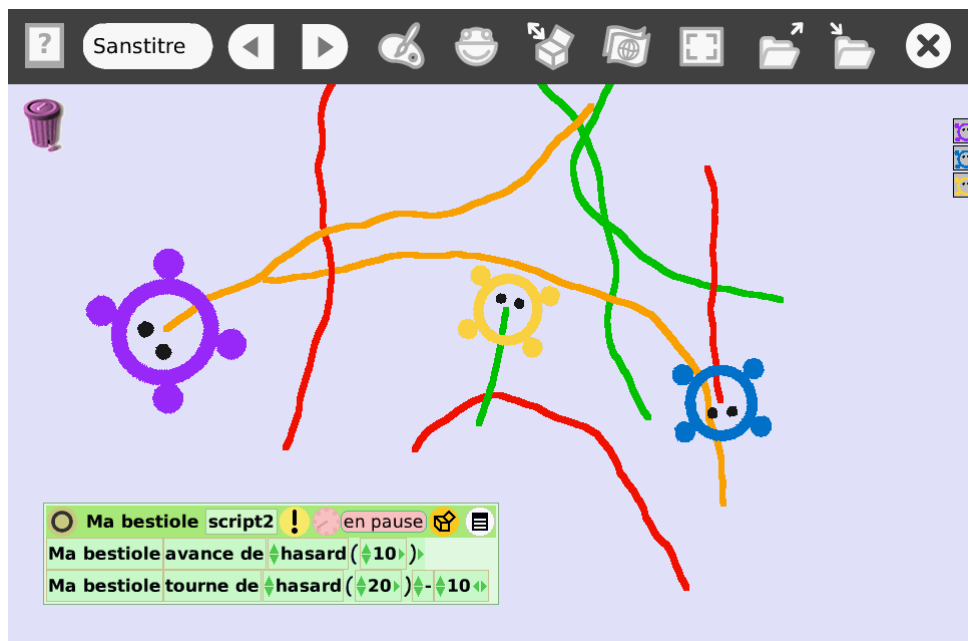


Une astuce pour ne pas que la bestiole se retrouve coincée dans un coin : si on ajoute la brique **rebondit** (qui est disponible dans la catégorie **Mouvements**) en haut ou en bas de notre script, la bestiole fera demi-tour lorsqu'elle heurte un bord de fenêtre.



Cette fois-ci, nous avons une vraie trajectoire de bestiole qui ne sait pas trop où elle va !

Voilà, nous savons maintenant comment créer un petit programme ! Avant de passer à la suite, une petite astuce : après que le script est démarré, et pendant que la bestiole bouge, affichons son halo en faisant un clic droit dessus. Cliquons maintenant sur l'icône  **Dupliquer** : hop ! nous venons de faire une copie de notre bestiole.





# Chapitre 3


## La danse des scripts

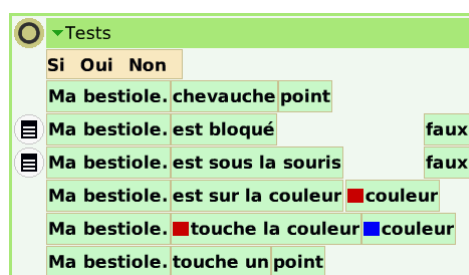
### 3.1 Jouons avec les briques

Pour continuer, allons explorer quelques unes des catégories de briques disponibles dans le visualisateur. Au chapitre 2.3, nous avons vu comment était organisé le visualisateur. Regardons maintenant de plus près certaines des catégories existantes.



Les tests sont au cœur de tous les programmes informatiques. C'est ce qu'on appelle une **Structure de contrôle**.

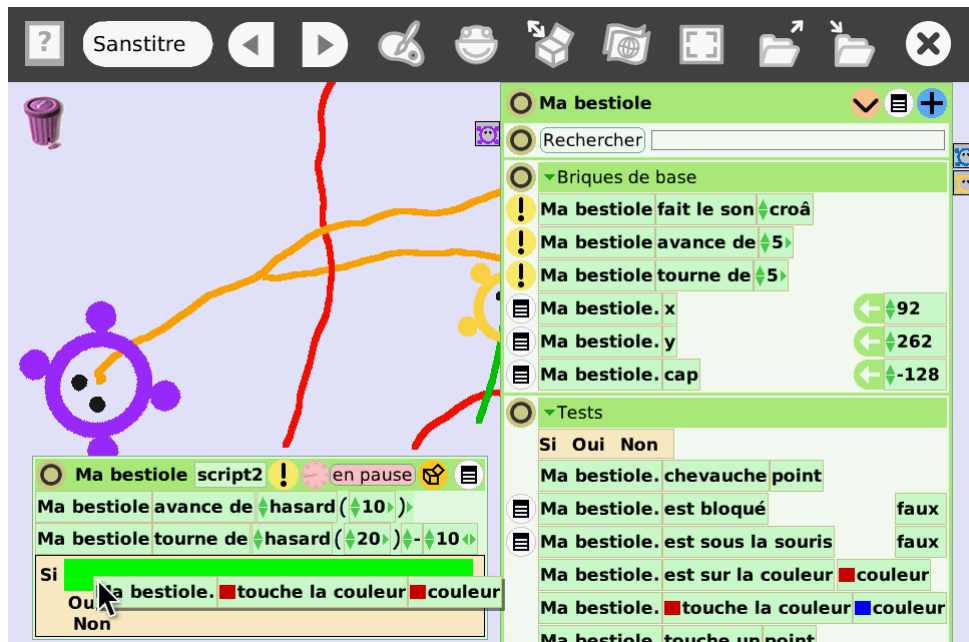
Commençons par la catégorie **Tests** qui doit déjà être affichée à l'écran (sinon, on peut masquer les autres panneaux avec l'icône  **Minimiser**) :



La première brique, **Si Oui Non**, est une pièce essentielle pour construire les programmes. C'est elle qui permet d'ajouter un test. Les autres briques de cette catégorie sont les différents tests que nous avons à disposition.

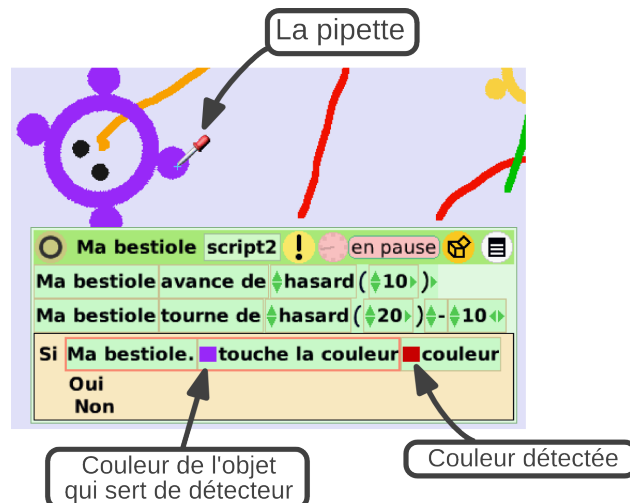
Tentons tout de suite d'utiliser cette brique. Glissons-la dans le script de notre bestiole, en dessous des autres briques, puis ajoutons le test **touche la couleur** en le faisant glisser devant le **Si** :

## La danse des scripts



Le test **touche la couleur** est Vrai lorsque une certaine couleur de notre objet (pour l'instant, le rouge, comme l'indique le premier petit carré dans la brique de test) touche une autre couleur (pour l'instant, rouge aussi : c'est la couleur du deuxième carré).

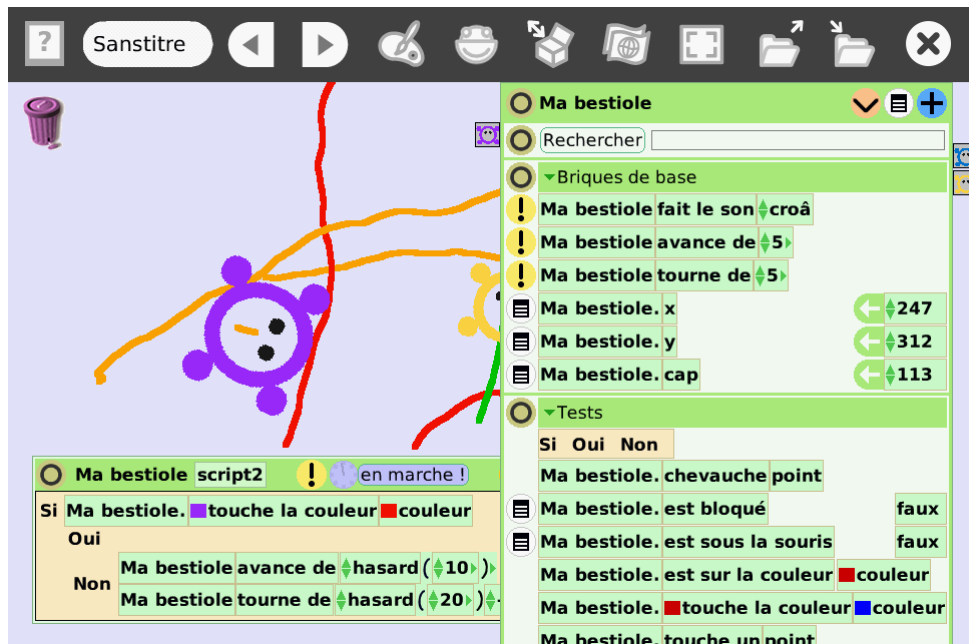
Changons la couleur de notre objet qui doit déclencher le test en cliquant sur le premier carré rouge, puis, à l'aide de la pipette qui apparaît, en cliquant sur la couleur que nous voulons :



Choisissons de la même manière comme couleur à détecter la couleur d'une des traces déjà à l'écran.

Maintenant, nous voudrions que notre bestiole s'arrête dès qu'elle touche cette trace. Comment faire ? Il suffit de **déplacer** les briques **avance de** et **tourne de** qui sont au-dessus de notre script à l'intérieur du test, et plus précisément, en face du **Non**. Ainsi, notre bestiole n'avancera et ne tournera **que si elle ne touche pas** la couleur de la trace :





Sur l'exemple ci-dessus, le script est en marche, mais la bestiole n'avance pas car elle touche la trace rouge.

La catégorie **Tests** contient cinq autres tests. Nous en utiliserons certains un peu plus tard. Si vous laissez la souris quelques instants sur chacun d'eux, une bulle d'aide apparaît, qui donne un bref descriptif de ce que fait le test.

Passons à la catégorie **Divers** qui contient deux briques très utiles, les briques **Montrer** et **Cacher**.



Modifions notre programme précédent pour faire disparaître notre bestiole quand elle touche une trace :



## La danse des scripts



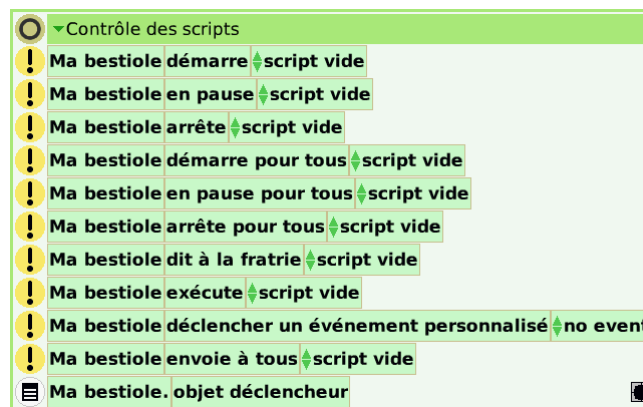
N'oublie pas ! on peut toujours exécuter une action (ou un script) une fois en cliquant sur le **!** point d'exclamation placé devant.

En exécutant le programme, notre bestiole va disparaître dès qu'elle touche la couleur détectée ...mais si tu affiches toujours la trace de ta bestiole, tu devrais voir la trace repartir. en effet, vu que notre objet est invisible, les couleurs ne se touchent plus, notre test devient `Faux`, et la bestiole repart ! Nous allons voir tout de suite comment améliorer cela.

En attendant, nous pouvons faire réapparaître la bestiole en cliquant une fois sur le **!** point d'exclamation placé devant la brique `Montrer` dans le visualisateur.

### 3.2 Actions, scripts, paramètres...

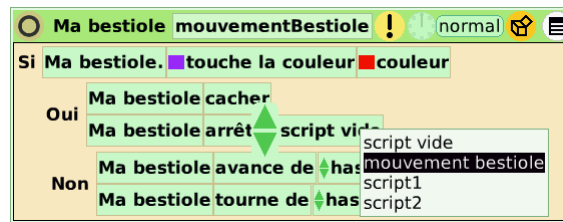
Comment faire pour que la bestiole s'arrête, même quand elle est invisible ? Une solution est tout simplement ...d'arrêter le script ! Normalement un script est démarré et arrêté en cliquant sur la **🕒** petite horloge. Mais on peut aussi programmer une mise en route ou un arrêt. Les briques pour cela sont dans la catégorie **Contrôle des scripts** :



Renommons d'abord notre script afin de lui donner un nom clair, par exemple `MouvementBestiole`. Pour cela, il faut cliquer sur le nom du script, et entrer le nouveau nom :

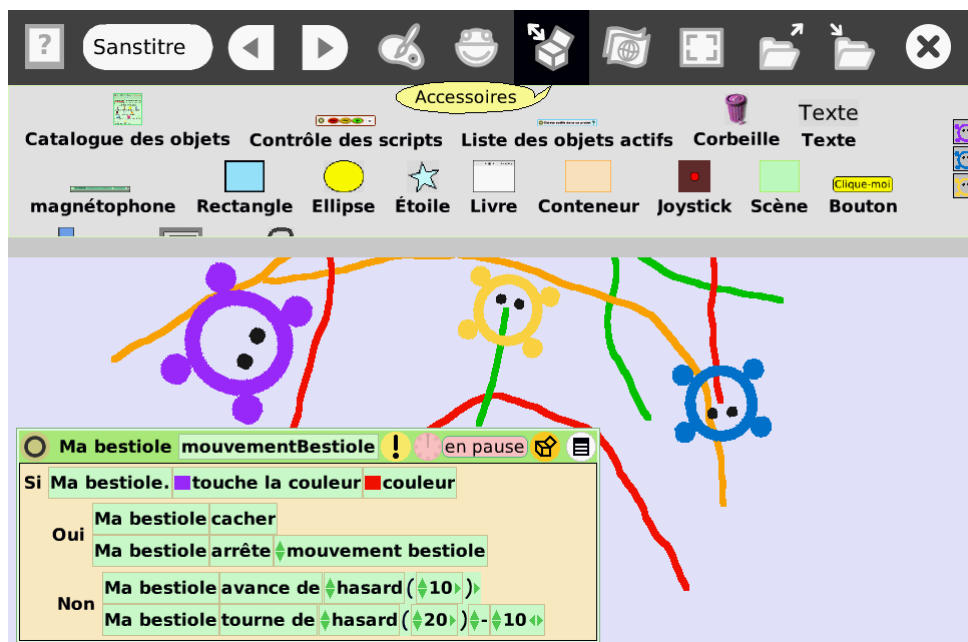


Ensuite, si nous ajoutons au script de la bestiole la brique `arrête` en face du `Oui` , et que nous sélectionnons le script `mouvement bestiole`, nous devrions voir notre bestiole disparaître et, cette fois, s'arrêter sur la couleur.



Une utilisation bien pratique du contrôle des scripts, c'est de pouvoir faire un bouton d'arrêt d'urgence et de remise à zéro : il s'agit d'ajouter un bouton qui, quand on clique dessus, arrête tous les scripts et remet les objets dans leur état initial.

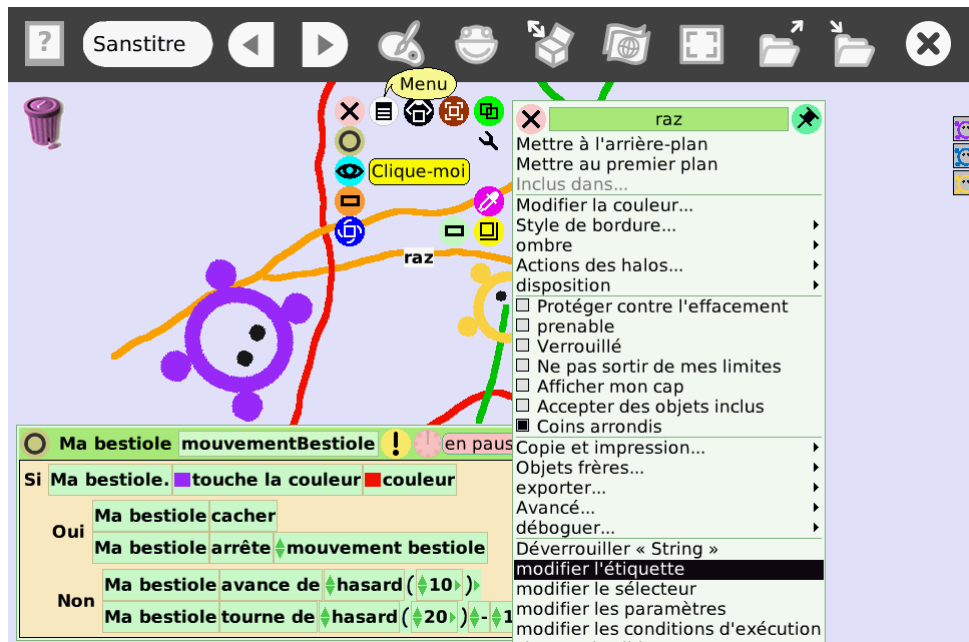
Pour cela, nous allons commencer par créer un bouton. Pour ce faire, il faut ouvrir le menu **Accessoires**, puis cliquer sur le bouton :



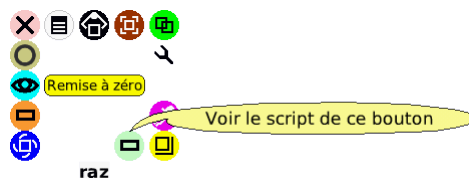
On ne peut pas déplacer le bouton simplement en cliquant dessus comme les autres objets. Pour le déplacer, utilisez l'icône **Attraper** dans le halo du bouton.

Renommons-le raz (remise à zéro), et, dans le halo, cliquons sur l'icône **Menu** pour afficher le menu du bouton. L'option **Modifier l'étiquette** va nous permettre de changer l'étiquette du bouton pour quelque chose comme « Remise à zéro ».

## La danse des scripts




En cliquant sur l'icône verte, on peut afficher un script qui se déclenchera lorsqu'on appuie sur le bouton :



Ce script de remise à zéro devra :

1. arrêter le script `MouvementBestiole`,
2. effacer toutes les traces,
3. replacer la bestiole au centre de l'écran.

Nous savons déjà comment le remplir pour arrêter le script de la bestiole et effacer toutes les traces.

Il faudrait aussi mettre le paramètre de la brique `laisse trace` de la bestiole à `Faux`, afin qu'elle ne laisse pas de trace en permanence. Comment faire pour changer la valeur d'un paramètre dans un programme ? Jusqu'à présent, nous n'avons ajouté à nos programmes que des **actions** (on reconnaît ces briques à l'icône ), et si nous essayons de glisser une brique **paramètres**, nous n'y arrivons pas.

Les **actions** peuvent être exécutées en cliquant sur le point d'exclamation

Les **paramètres** ne sont pas directement exécutables. Mais on peut utiliser leur valeur et parfois, la modifier

En cliquant ici, on obtient l'action "donner une valeur à ce paramètre"

Les paramètres ont un **type** : numérique comme ici...


...une couleur comme ici...


...vrai ou faux (valeur booléenne, utilisable dans un test) comme ici...

et beaucoup d'autres sont possibles !

En réalité, pour **modifier** la valeur d'un paramètre (et donc, en quelque sorte, transformer une brique de paramètre en brique d'action), il faut cliquer sur l'icône **←** **Assignation**.



On peut toujours masquer un script, sans le supprimer, en cliquant sur l'icône .

Pour remettre la bestiole au centre de l'écran, le plus simple est de l'y déplacer, puis d'afficher la catégorie **Briques de base** de la bestiole. En faisant glisser les briques **x**, **y** et **cap** via leur  flèche verte dans le script de remise à zéro, on va pouvoir automatiquement replacer notre bestiole au milieu de l'écran à chaque fois que le script **raz** sera exécuté : en effet, celui-ci va **assigner** (ou *affecter*) les valeurs en pixels et en degré que nous venons de définir aux paramètres **x**, **y** et **cap**.


## La danse des scripts



Vous avez peut-être remarqué que ce script n'était pas un script de l'objet *Ma bestiole*, mais d'un objet appelé *world*. L'objet **Monde** (*world* en anglais) est le conteneur le plus général : tout les autres objets en font partie.



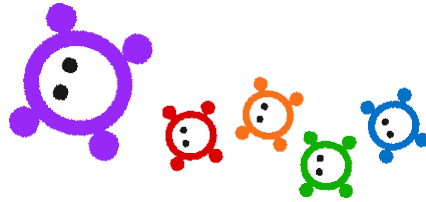
Le script *raz* pourrait en réalité être créé dans n'importe quel objet : ça marcherait tout aussi bien. Mais pour faciliter la compréhension des programmes, il est important de garder une bonne organisation, et de veiller à attacher les programmes aux objets concernés.

Si vous voulez retrouver plus tard le script *raz*, affichez le visualisateur du monde en faisant un clic droit sur le bureau pour faire apparaître le halo du monde, et comme d'habitude, cliquez sur .



Le script `raz` devrait être présent dans la catégorie **Scripts**.

Utilisons notre script `raz` pour faire un peu de ménage avant de passer à la suite : les choses sérieuses vont commencer !





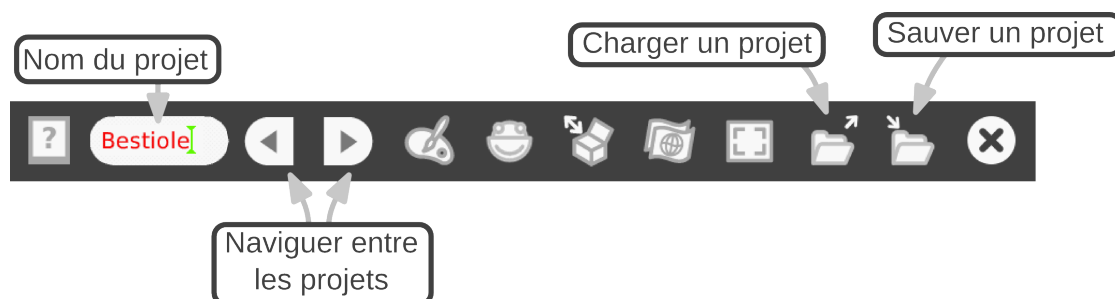


# Chapitre 4

## Les bestioles se multiplient

### 4.1 Sauvegarder son projet

Dans les prochains chapitres, nous allons être amenés à faire des tests durant lesquels pleins de « copies » de nos bestioles vont être créées, ce qui pourra parfois faire « planter » SqueakBot... Avant d'aller plus loin, il est donc utile de voir comment sauvegarder notre projet. Ça sera aussi bien utile si on veut l'échanger avec d'autres ou le présenter sur Internet.



Il faut d'abord nommer notre projet (par exemple, *Bestioles*), puis cliquer sur le bouton du menu **Publier** (ce bouton permet de sauvegarder le projet sur le disque dur, mais aussi de le publier en ligne si vous avez un compte *Squeakland*).




Il est possible de créer d'autres dossiers en faisant un clic du milieu dans l'arborescence de l'écran de sauvegarde. Cependant un bogue rend la sélection de l'entrée dans le menu qui s'ouvre alors un peu difficile. Il peut être nécessaire de déplacer un peu la fenêtre et de garder appuyer le bouton du milieu.

Les deux écrans suivant permettent d'ajouter des méta-données à votre projet, puis de le sauvegarder dans le dossier par défaut *My Squeak*.

## Les bestioles se multiplient

Description de ce projet	
Nom du projet	Bestioles
Description	
Auteur	
Tags	
Sujet	(none)
Âge cible	(none)
Région	(none)
<input type="button" value="OK"/> <input type="button" value="Annuler"/>	

Publier ce projet	
	<ul style="list-style-type: none"><li>My Squeakland<ul style="list-style-type: none"><li>My Squeak</li></ul></li></ul>
<input type="button" value="Connexion"/> (non connecté)	<input type="button" value="Enregistrer"/> <input type="button" value="Annuler"/>

Dans les étapes suivantes, n'oubliez pas de sauvegarder régulièrement votre projet !

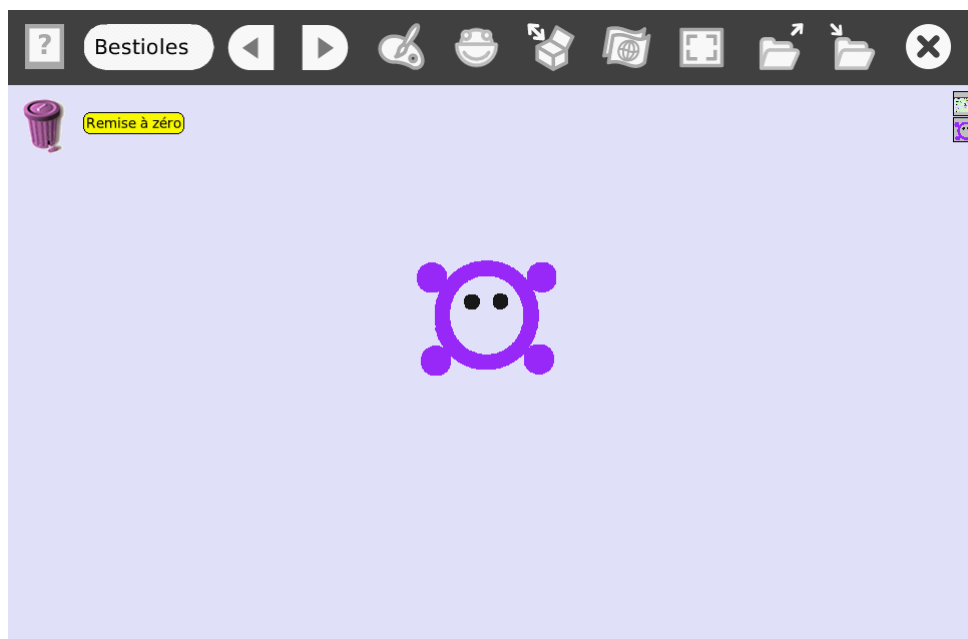


Il existe un greffon pour les navigateurs courant permettant d'afficher directement, sur Internet, un projet SqueakBot : bien pratique pour valoriser un travail ! Il peut être téléchargé depuis cette page : <http://wiki.squeak.org/squeak/1865> (en anglais).



## 4.2 Les bestioles font des bébés

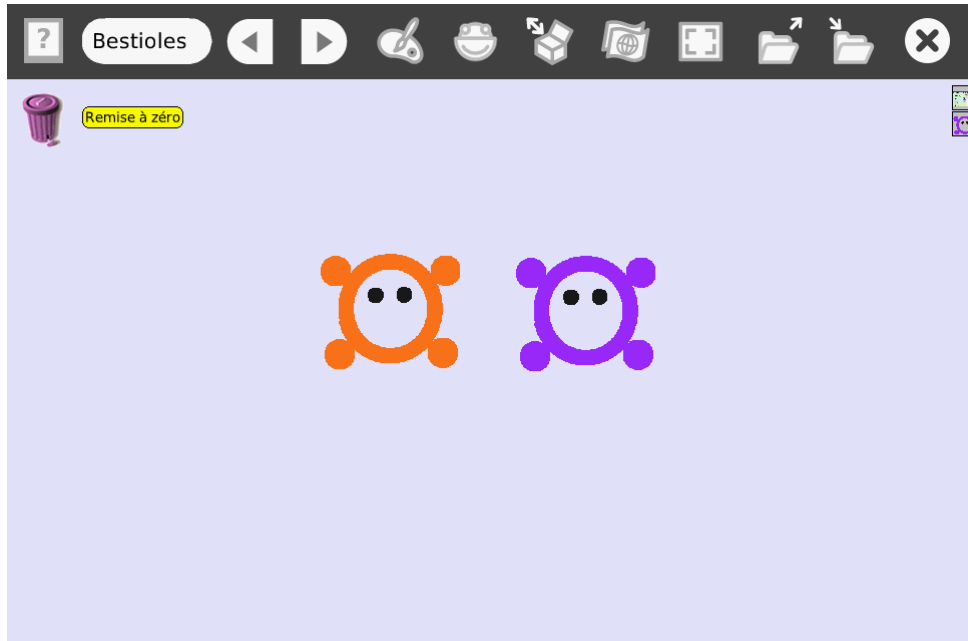
L'objectif de notre projet est de simuler l'évolution d'une population de bestiole qui se reproduit, se nourrit et meure. La première étape est donc de les faire se « reproduire ». Nous allons simplifier un peu ce que nous propose la nature, et des bébés bestioles apparaîtront dès que deux bestioles de couleur différente se toucheront.

Pour cela, repartons où nous en étions resté au chapitre précédent :




## Les bestioles se multiplient

et copions notre bestiole en changeant sa couleur (avec l'icône  **Duplicer** puis l'icône  **Redessiner** pour remplir notre bestiole avec une autre couleur).



Pour créer une copie de notre bestiole, c'est assez simple : créons un nouveau script (en glissant un **script vide** depuis la catégorie **Scripts** de la bestiole) que nous pouvons nommer `seMultiplie`.

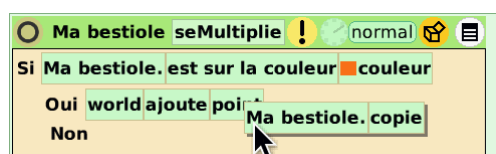
**Attention : le script décrit ci-dessous risque de bloquer SqueakBot car il crée très rapidement des centaines de copies de bestioles. Avant de le tester, pensez à sauvegarder votre image !**

Dans ce script, nous allons ajouter un test de couleur (brique **est sur la couleur**), et quand le test est vrai, nous allons utiliser la brique **ajoute** de la catégorie **Conteneur** de l'objet `Monde` (accessible via son halo et , en faisant un clic droit sur le bureau).

Cette brique à par défaut comme paramètre un objet **point** :

**Monde ajoute point**

Or, nous ne voulons pas ajouter un point, mais une copie de la bestiole. Nous pouvons en obtenir une dans la catégorie **Divers** de la bestiole. Prenons cette brique **Ma bestiole.copie** et remplaçons **point** :

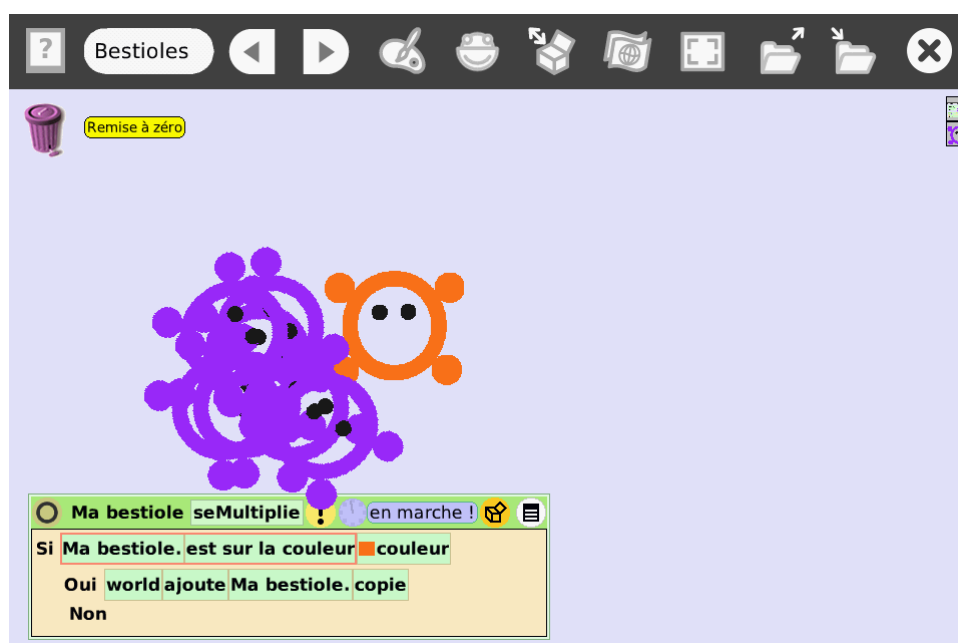


## Les bestioles se multiplient



Tous les scripts peuvent être démarrés et arrêtés depuis la catégorie **Scripts** du visualisateur de l'objet auquel ils appartiennent. C'est parfois bien pratique pour ne pas envahir le bureau avec des multitudes de fenêtres.

Sauvegardons l'image en cours, et testons notre nouveau script en démarrant préalablement le script `mouvementBestiole`. Une catastrophe s'annonce : en effet, dès que la bestiole violette touche la bestiole orange, SqueakBot va créer très rapidement des copies de la bestiole violette et en quelques secondes, le logiciel va se bloquer... On a gagné le droit d'interrompre et de redémarrer SqueakBot...



Comment éviter cela ?

Il y a plusieurs choses à faire :

1. Créer un script qui permette de détruire facilement toutes les copies,
2. Rajouter ce script au script de remise à zéro,
3. Éloigner notre bestiole de la bestiole violette dès qu'elle a fait un bébé,
4. Essayer d'éviter que le bébé ne touche lui aussi directement la bestiole orange pour, là non plus, ne pas faire de « bébés en série ».

Prenons ces différents points dans l'ordre :

Pour détruire facilement toutes les copies de notre bestiole, il faut d'abord savoir que les copies réalisées avec la brique `copie` sont ce que SqueakBot appelle des *frères* de notre bestiole. Ils gardent un lien particulier avec leur modèle initial, ce qui permet par exemple d'exécuter un script sur toutes les copies à la fois avec la brique `dit à la fratrie` de la catégorie **Contrôle des scripts**.

Si on ajoute un script `meure` à notre bestiole, qui la fait disparaître, on peut « tuer » tous les frères en utilisant la brique `dit à la fratrie meure`. Créons un nouveau script `tuerMesFreres` qui fait ça :

## Les bestioles se multiplient



Par contre, attention à ne pas supprimer notre première bestiole en exécutant son script `meure`, sinon, il faudra tout recommencer !

Pour s'assurer de toujours garder un modèle (ou *prototype*) de notre bestiole, une bonne idée est de la glisser dans le menu **Accessoires** :




Il est maintenant facile de refaire des copies de notre bestiole si par hasard on les supprime toute, en faisant un simple glisser-déposer depuis le tiroir des accessoires vers le bureau.

FiXme Fatal!]FiXme Fatal!<sup>1</sup>

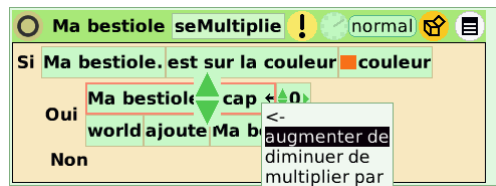
Ajoutons le script `tuerMesFreres` dans le script de remise à zéro (accessible en cliquant sur l'icône  du halo du bouton *Remise à zéro*).


Ensuite, nous voulons éloigner la bestiole violette de la bestiole orange quand elles se touchent. Pour cela, nous pouvons ajouter des actions dans le test de couleur qui crée les copies. Il faut faire faire demi-tour à la bestiole (c'est-à-dire ajouter 180 degrés à son `cap`), et la faire avancer un peu pour être sûr qu'elle ne touche plus la bestiole orange.

Pour ajouter 180 degrés au `cap`, il faut glisser la brique `cap` de la bestiole dans le test, en l'attrapant par la  flèche verte. En cliquant ensuite sur `cap ←`, on accède à un menu dans lequel on peut choisir `augmenter de`.

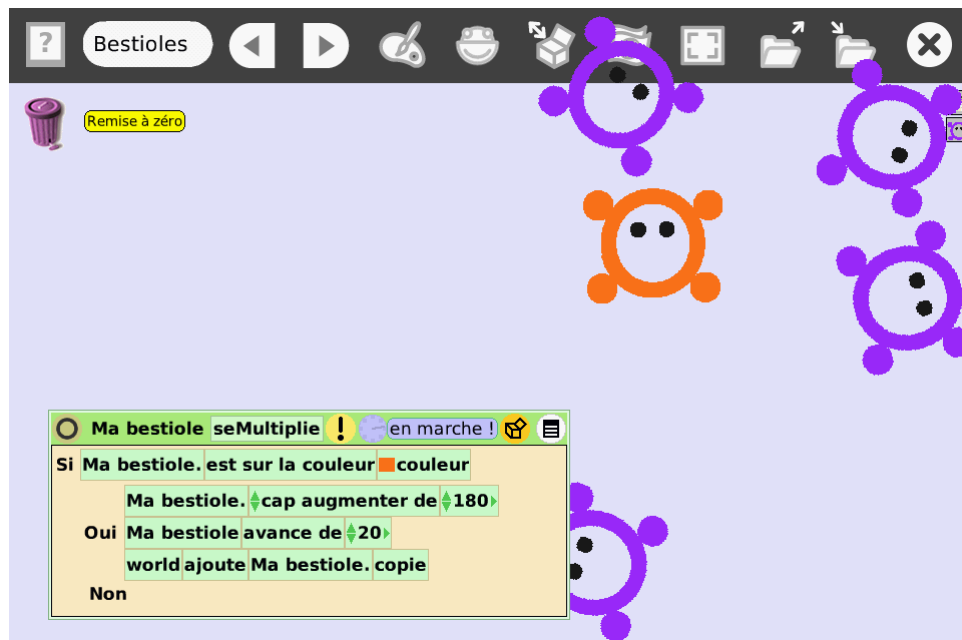
1. FiXme Fatal: Probleme avec cette approche, le script RAZ ne marche plus !

## Les bestioles se multiplient



Attention : nos bestioles se reproduisent quand une bestiole violette touche du orange ...or la brique de test contient du orange ! Les bestioles vont donc aussi se multiplier lorsqu'elles touchent la brique de test. On peut éviter ça en masquant le test avec l'icône . Voyez-vous une autre manière de faire, en changeant le type de test ?

Après quelques contacts, de nouvelles bestioles apparaissent :

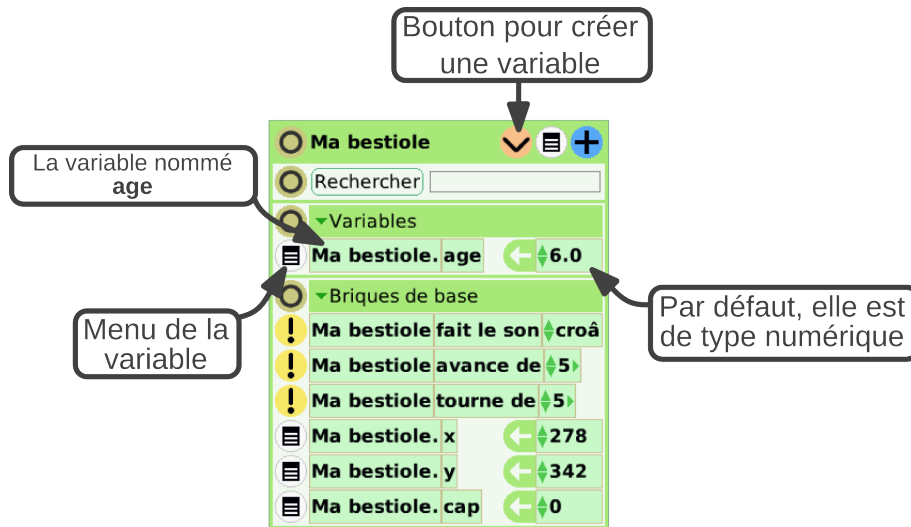


### 4.3 Quand je serais grand... introduisons les variables

Nous voulons simuler que nos bestioles vieillissent. Pour cela, nous allons avoir besoin d'une nouvelle brique paramètre pour nos bestioles qui contiendra l'âge.


C'est très simple à faire grâce au bouton  **Variable** du visualisateur d'un objet :

## Les bestioles se multiplient

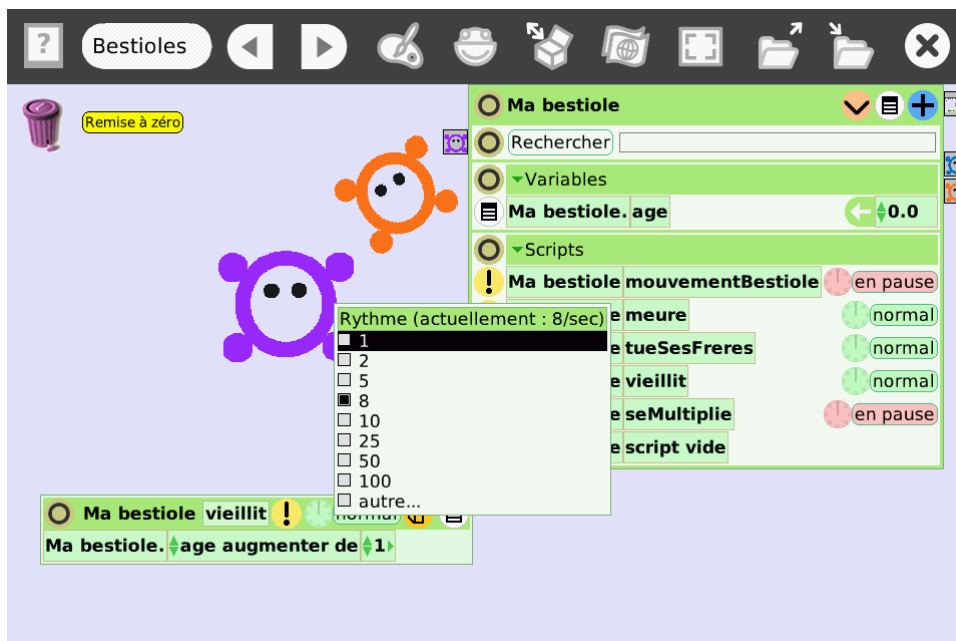


Créons donc une variable `age`, et changeons sa valeur pour la mettre à zéro.  
Comment faire pour faire « vieillir » la bestiole ? Il faudrait incrémenter, par exemple toute les secondes, l'âge de la bestiole.



Pour changer le rythme d'exécution d'un programme, il faut cliquer et rester appuyé sur l'icône  du script jusqu'à ce que le menu *Rythme* s'affiche. On peut alors choisir combien de fois par seconde le script doit s'exécuter.

Créons un nouveau script, `vieillit`, chargé de cette tâche, et changeons sa vitesse d'exécution pour qu'il soit appelé une fois par seconde :




Si on teste ce script en le mettant en marche, l'âge de la bestiole devrait augmenter petit à petit.

## Les bestioles se multiplient

Pour faire mourir notre bestiole quand elle devient trop vieille, il suffit d'ajouter un test au script `vieillit` qui appelle le script `meure` de la bestiole quand son âge est supérieur à l'âge maximal (par exemple, 100) :




Pour faire une comparaison d'une valeur numérique, il suffit de glisser une brique numérique sur le test : les opérateurs de comparaison vont être automatiquement ajoutés, comme sur la capture d'écran ci-contre.

**Attention !** Si vous attendez que la bestiole meure, elle va disparaître, avec tous ses scripts ! C'est le moment de faire une copie de votre bestiole (avec l'icône  **Dupliquer**) à garder dans un coin (ou dans le tiroir **Accessoires**, comme nous avons vu plus tôt). Pensez aussi à faire des sauvegardes régulières.

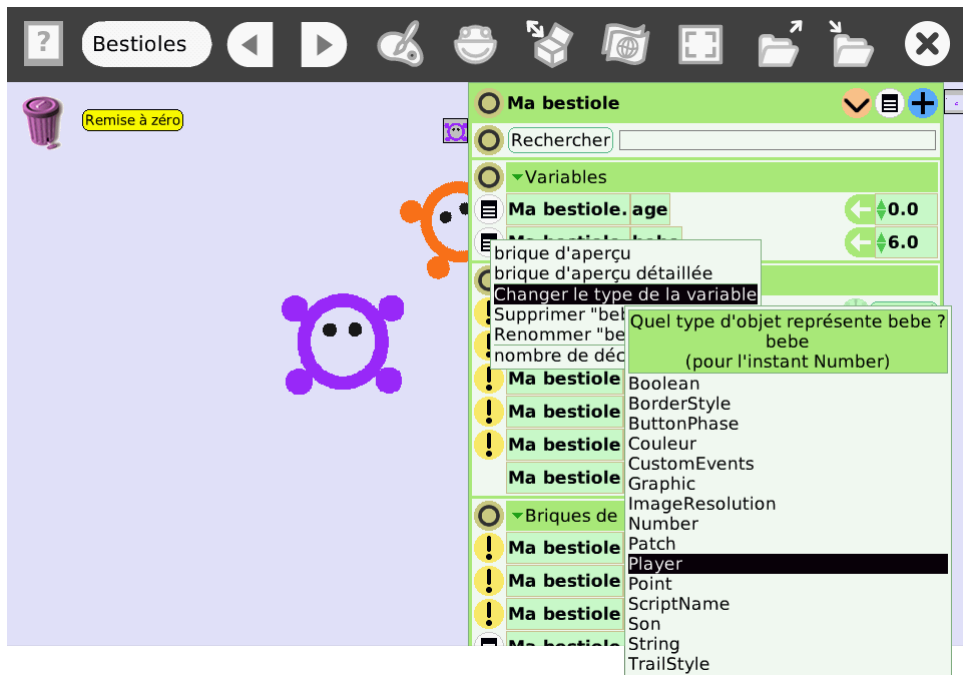
Si nous relançons la simulation, avec les bestioles qui se déplacent, se reproduisent, et meurent, nous allons cependant constater qu'il y a un problème : tous les enfants d'une bestiole meurent en même temps qu'elle, même s'ils ont été créés bien plus tard (et sont donc plus jeunes). Comment cela se fait-il ?

Lorsqu'on crée un enfant, on crée une copie de la bestiole initiale, avec tous ses paramètres... y compris son âge actuel ! Nous voulons copier certains paramètres (la position de la bestiole, par exemple, afin que les petits apparaissent au même endroit que leurs parents, ou encore, l'état des scripts – démarrés ou non – pour que les enfants se mettent en marche immédiatement), mais par contre, il faut remettre l'âge à zéro.

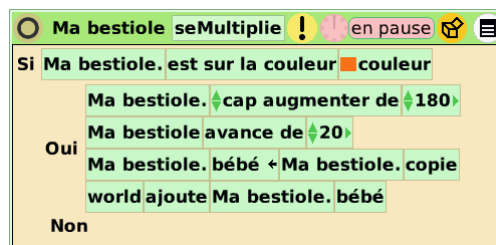
Pour pouvoir faire ça, il faut que l'on garde une **référence** vers les nouveaux objets créés quand on les copie, pour pouvoir ensuite changer leur âge. Et pour garder une référence, on utilise une variable !

Ajoutons à notre bestiole une nouvelle variable (avec l'icône ) que nous allons appeler *bébé* et changeons son type actuel (*nombre*) pour le remplacer par le type *objet* (*Player* en anglais) :

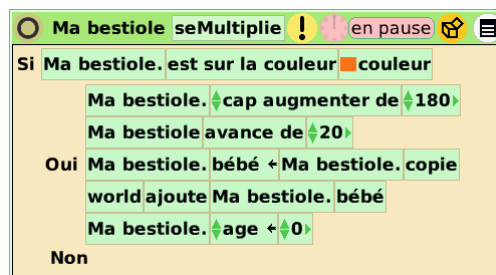




Modifions maintenant le script `seMultiplie` : nous allons donner à la variable `bébé` la valeur `Ma bestiole.copie`, et ajouter dans le Monde cette copie, grâce à la variable `bébé` qui fait maintenant *référence* à la copie :

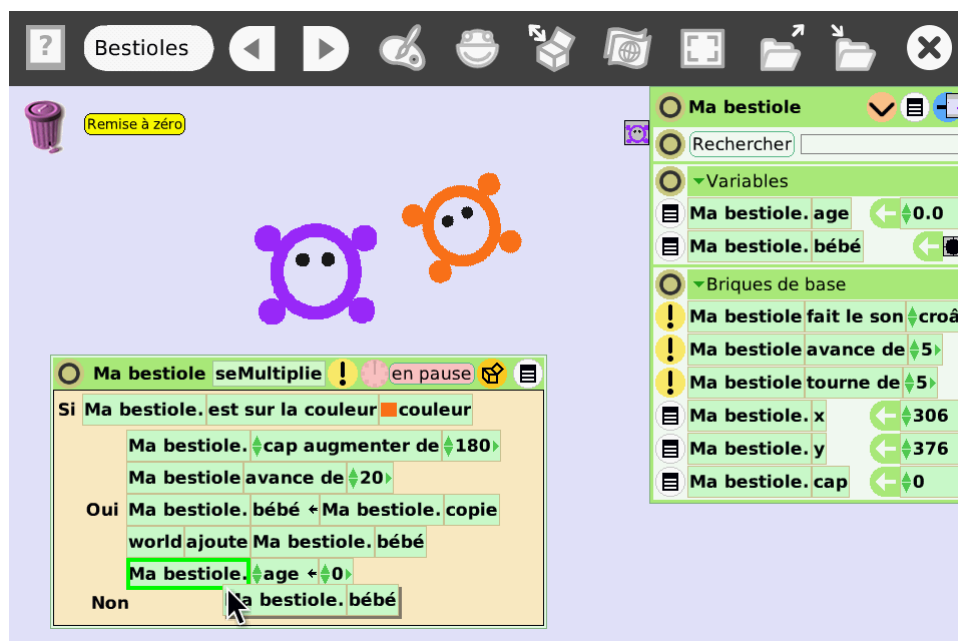


Jusque là, ça ne change pas grand chose. Mais comme on a maintenant sous la main une référence à notre nouvelle copie de bestiole, on va pouvoir modifier son âge : pour cela, deux étapes. D'abord, il faut ajouter une brique `âge ←` normalement et la mettre à zéro :



Puis il faut **remplacer** la partie `Ma bestiole.` (c'est à dire, l'objet duquel on veut changer l'âge) par la variable `bébé` :


## Les bestioles se multiplient



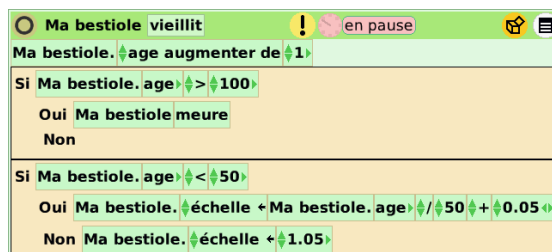
Maintenant, quand de nouvelles bestioles voient le jour, elles démarrent bien avec un âge de 0.

### 4.4 Ça grandit, ça grandit

En vieillissant, forcément, on grandit ! En réutilisant la variable `age`, il est facile de faire changer la taille de la bestiole en fonction de son nombre d'années.

Dans la catégorie **Géométrie**, la propriété `échelle` permet de jouer avec la taille de nos objets (essayez de redimensionner l'objet dont le visualisateur est ouvert avec , et voyez comment évolue l'échelle).

Modifions le script `vieillit` pour faire grandir notre bestiole (sans la faire partir de 0 : on ne verrait plus du tout le bébé ! :



À toi de jouer : Tu peux d'abord améliorer le calcul de l'échelle pour arriver pile à 1.0 à 50 ans, et non à 1.05. Ensuite, comment faire pour que la bestiole se « tasse » lorsqu'elle devient trop vieille, vers 80 ans ? Penses à imbriquer les blocs **Si Oui Non** !

### 4.5 Bon appétit !

Notre simulation commence à prendre forme. Dernière chose importante : l'alimentation de nos charmantes bestioles. En effet, tout animal a besoin d'énergie pour se déplacer !


Nous allons essayer, dans ce chapitre, de programmer les choses suivantes :

- Des petits paquets de nourriture qui puissent être mangés,
- Le niveau d'énergie des bestioles, qui baisse en permanence, et remonte quand la bestiole trouve de la nourriture,
- De la nourriture, qui disparaît au fur et à mesure qu'elle est mangée,
- La vitesse de la bestiole, qui dépend de l'énergie dont elle dispose.

#### 4.5.1 Des petits paquets de nourriture qui se font grignoter

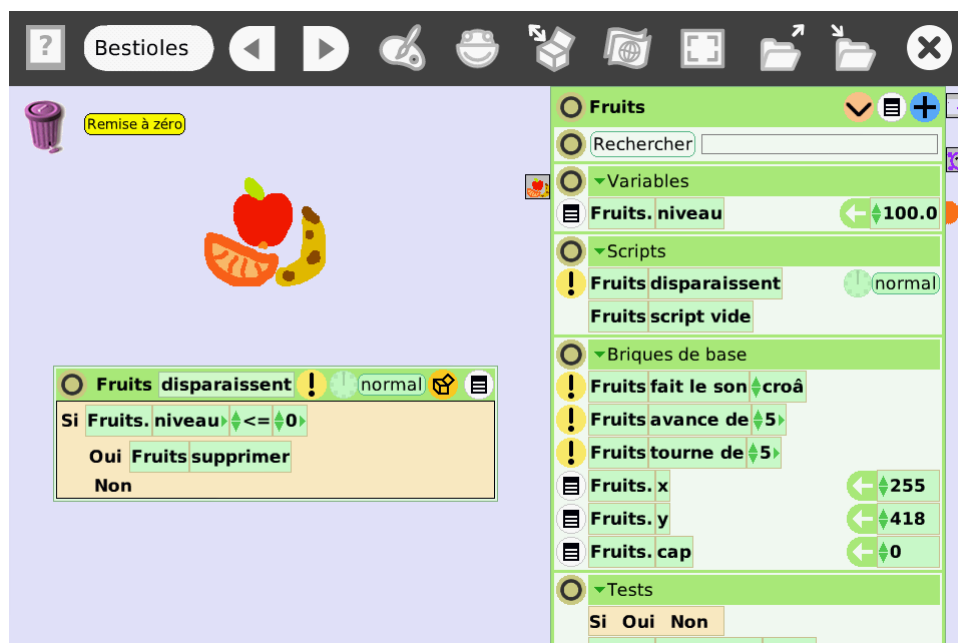
Dessignons pour commencer quelques fruits :



Appelons ce nouvel objet `Fruits`, et ajoutons-lui une nouvelle variable (avec l'icône  dans son visualisateur) appelée `niveau` : cette variable représente la quantité de nourriture encore disponible sur ce tas là.

Ajoutons encore un script aux fruits pour les faire disparaître lorsque le niveau est à zéro (c'est un script très proche de celui que nous avons créé lorsque les bestioles meurent) :

## Les bestioles se multiplient



Pour ne pas que les fruits disparaissent trop rapidement, pense bien à changer le rythme du script (en restant appuyé sur l'icône **Horloge**). Tu peux le mettre par exemple à 1 Hz (1 fois par seconde).

Glissons les fruits dans le tiroir des accessoires pour en garder une copie toujours disponible quand on voudra ajouter de la nourriture dans le monde.

Il faut maintenant aller modifier les bestioles pour qu'elles mangent nos fruits.

Commençons par leur ajouter une variable *énergie*, et créons un script appelé *fatigue* qui fait diminuer l'énergie des bestioles :

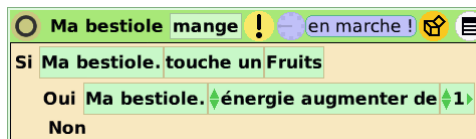


## Les bestioles se multiplient

Dès que nous mettrons en route le script, l'énergie de notre bestiole diminuera petit à petit. Pour ce script aussi, change le rythme pour que les bestioles ne s'épuisent pas trop vite.


Il faut maintenant faire en sorte que les bestioles puissent se nourrir, et regagner ainsi de l'énergie.

Pour cela, créons un nouveau script dans notre bestiole, `mange`. Ce script utilise une brique de test ainsi que le test `touche un` que tu trouveras, comme tous les tests, dans la catégorie **Tests** de la bestiole :

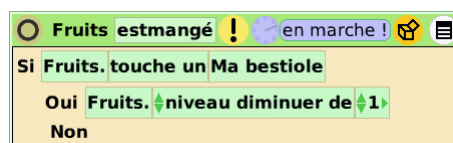


À toi de jouer : peux-tu modifier un peu le script pour s'assurer que l'énergie de la bestiole ne dépasse jamais 100 ?

Le test `touche un` est un peu particulier : il permet de détecter quand notre objet touche une **copie** d'un autre objet. Il prend en paramètre le **modèle** (l'objet dont on va faire des copies). Dans notre script, le modèle est donc `Fruits`.

Si on teste le script maintenant, il ne se passera rien : le contact entre la bestiole et les fruits ne sera pas détecté. Le test ne fonctionne que sur des *copies* du modèle. Il faut donc faire une copie des fruits (icône ) , et cette fois, le test va marcher. L'intérêt de ce test est qu'on va pouvoir faire autant de copies de fruits qu'on veut, le test marchera pour toutes ces copies.

De même, créons un script `est mangé` dans les fruits, inversé par rapport au script `mange` :



On peut maintenant tester notre système de nourriture. Démarrons les scripts `mange` et `est mangé` de nos modèles de bestiole et de fruit, puis créons des copies des modèles.

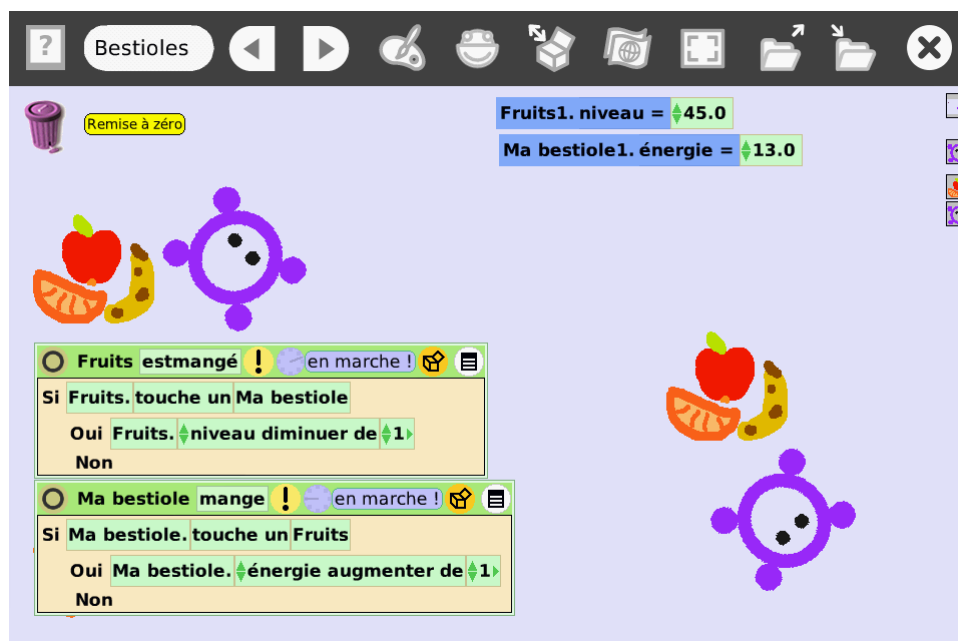



Pour afficher la valeur d'une variable sur le bureau, il suffit de l'attraper dans le visualisateur, et de la lâcher sur l'aire de jeu. Pratique pour voir comment les programmes se comportent.

Tu peux aussi démarrer les scripts `mouvementBestiole` et `fatigue` de la copie de bestiole.

Lorsque notre copie de bestiole passe sur la copie de fruit, le niveau des fruits baisse, et l'énergie de la bestiole augmente.

## Les bestioles se multiplient





À toi de jouer : essaye de modifier les fruits pour qu'ils affichent en permanence leur niveau d'énergie restante. Petite piste : on peut créer un objet texte qu'on *ajoute* au fruit. Pour permettre à un fruit d'accepter des *objets inclus*, il faut activer l'option dans son menu (icône  de son halo).

Ça y est, notre simulation prend en compte l'alimentation de nos bestioles. Si elles ne mangent pas, elles perdent de l'énergie et finissent par mourir.

### 4.5.2 Les fruits disparaissent petit à petit

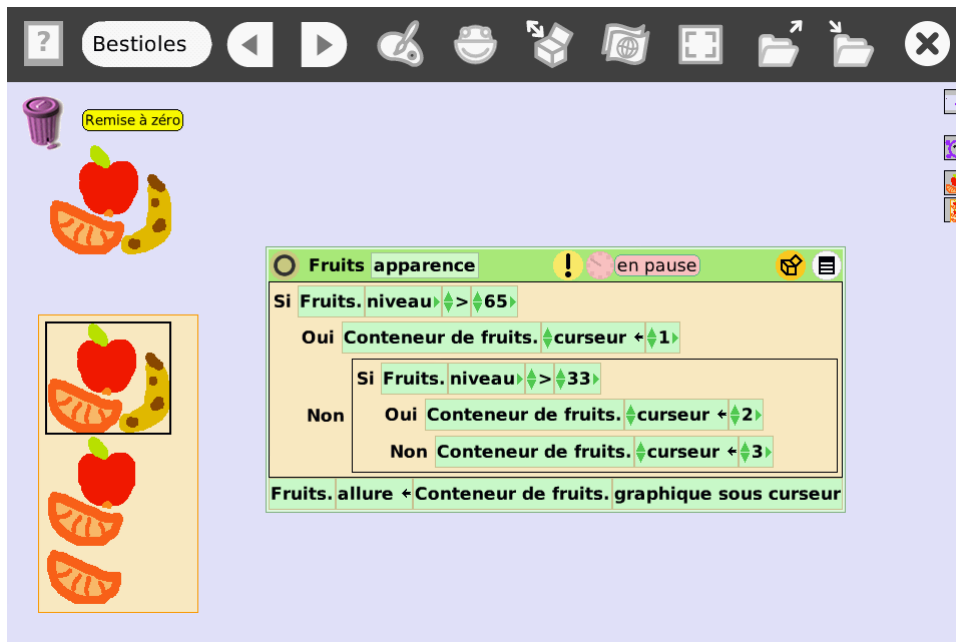
Au fur et à mesure que les bestioles mangent la nourriture, il serait logique que celle-ci disparaissent.

Une méthode pour faire disparaître petit à petit nos fruits consiste à réaliser plusieurs dessins (en  dupliquant et  modifiant le dessin original), et à les regrouper dans un **conteneur** (disponible dans le tiroir **Accessoires**) :



Ce conteneur va nous servir pour modifier l'apparence de nos fruits en fonction de leur niveau d'énergie restante.

Créons un nouveau script, `apparence`, pour les fruits :



Ce script utilise deux tests imbriqués pour déplacer le **curseur** du conteneur (le rectangle noir) sur le bon dessin, en fonction du niveau d'énergie.

Ensuite, il change l'**allure** des fruits (brique `allure` dans la catégorie **Allure** des fruits) en utilisant la brique `graphique sous curseur` de la catégorie **Conteneur** du conteneur.

On peut ensuite démarrer le script (et le laisser tourner en permanence), puis changer à la main la valeur de l'énergie des fruits dans leur visualisateur. Dès qu'on passe en-dessous de 65 puis en-dessous de 33, les fruits devraient changer d'apparence.



Cette technique avec le **Conteneur** permet de réaliser toutes sortes d'animations cycliques. Par exemple, un bonhomme qui marche.



À toi de jouer : pour finir cette partie, modifie le script de déplacement des bestioles afin que leur niveau d'énergie influe sur leur vitesse : rapide quand elles sont pleines d'énergie, lentes quand elles sont épuisées.

## **Les bestioles se multiplient**

---



# Chapitre 5

## Regardons vivre notre monde merveilleux !

### 5.1 Préparons la simulation

Ça y est, tous les éléments de notre simulation de vie de bestiole sont en place. Il est temps de passer à la simulation proprement dite.

Un peu d'organisation s'impose : avant de commencer à simuler, il faut préparer des modèles propres de bestioles et de fruits, les réduire un peu en taille, et les mettre au chaud dans le tiroir **Accessoires**.

–

### 5.2 Le centre de contrôle

FiXme Fatal!]FiXme Fatal!<sup>1</sup>

[

### 5.3 L'évolution de la population

FiXme Fatal!]FiXme Fatal!<sup>2</sup>

[

---

1. FiXme Fatal: Mettre les différentes constantes de l'application sous forme de champs modifiables

2. FiXme Fatal: Tracer le nombre de bestiole en fonction du temps

**Regardons vivre notre monde merveilleux !**

---

# Chapitre 6

## Pour finir...

### 6.1 Conclusion

### 6.2 Pour aller plus loin

FiXme Fatal!]FiXme Fatal!<sup>1</sup>

[

---

1. FiXme Fatal: Parler des applications en robotique

**Pour finir...**

---


# Chapitre 7

## Annexes

### 7.1 Installer SqueakBot


### 7.2 Astuces et problèmes courants

- **Un objet a disparu (il a été caché et j'ai fermé son visualisateur). Comment le faire réapparaître ?**

Il est possible de faire réapparaître tous les objets cachés depuis le menu **Menu des objets**. Pour y accéder, cliquez sur l'icône  **Menu** du halo du monde (qu'on obtient par un clic droit sur une zone libre du bureau), puis sur **Menu des objets**. Dans le menu des objets, cliquez enfin sur **Montrer les objets cachés !**.

- **Comment passer du mode protégé au mode développeur ?**

Par défaut, SqueakBot fonctionne en mode *protégé* ou *enfant*. dans ce mode, de nombreux menus sont désactivés pour éviter certaines fausses manipulations.

Il peut être parfois nécessaire de passer en mode *développeur* pour accéder à l'environnement complet. Pour cela, il faut accéder aux **Préférences** depuis le menu du monde ( l'icône  **Menu** dans le halo du monde). Ensuite, dans la zone de recherche, cherchez la propriété `eToyFriendly`. En désactivant cette propriété, on passe en mode *développeur*. Effectuez la manœuvre inverse pour repasser en mode protégé.

### 7.3 Contacts

Ce guide a été rédigé au sein de l'association Planète Sciences (<http://planete-sciences.org>). Planète Sciences, association Loi 1901, propose et organise de très nombreuses activités et actions autour des sciences expérimentales, essentiellement pour les jeunes.

Pour prendre contact avec les enseignants, animateurs, formateurs qui travaillent directement avec SqueakBot, envoyez un mail à la liste de discussion [boiteabots@planete-sciences.org](mailto:boiteabots@planete-sciences.org).

Plus largement, la communauté francophone des utilisateurs de Squeak (le sys-


## Annexes

---

tème sur lequel est basé SqueakBot) peut être contacté via la liste [squeak-fr@lists.squeakfoundation.org](mailto:squeak-fr@lists.squeakfoundation.org).

Enfin, de très nombreuses ressources éducatives en français sont disponibles sur le site de l'Ofset, [community.offset.org/index.php/Squeak](http://community.offset.org/index.php/Squeak). N'hésitez pas à aller vous promener sur ces pages.

# Glossaire

**cap** Le cap d'un objet est son orientation absolue (en degrés) : un cap de 0 équivaut à une orientation vers le haut de l'écran. Pour modifier le cap, il faut faire tourner un peu l'objet avec , puis cliquer sur la flèche verte **en maintenant la touche Majuscule appuyée**. 14


**halo** C'est l'ensemble des icônes qui entourent un objet quand on fait un clic droit dessus. 8

**Monde** Le « monde » est l'ensemble du bureau de SqueakBot. C'est un objet similaire aux autres : on peut afficher son halo, son visualisateur, lui associer des scripts.... 28

**paramètre** Un paramètre d'une fonction est une option que l'on passe à la fonction et que l'on peut modifier.. 15

**script** Les scripts sont les programmes associés à chaque objet. 12

**Variable** Une variable est une étiquette que l'on peut créer dans un objet, et qui va faire *référence* à un autre objet, à un texte, à une valeur numérique... ça permet essentiellement de manipuler ou d'utiliser cet autre objet ou valeur dans un script.. 36

**visualisateur** Le visualisateur est la zone, spécifique à chaque objet, qui contient toutes les briques permettant la programmation. On l'affiche grâce à l'icône  « œil » du halo de l'objet. 12

Planète Sciences 2011.  
Cet ouvrage est diffusé sous licence Creative Commons Paternité-Partage à l'identique.



Les sources de ce document peuvent être téléchargées depuis le site GitHub.  
[http://github.com/skadge/squeakbot\\_pas\\_a\\_pas/](http://github.com/skadge/squeakbot_pas_a_pas/)