

# Programmation par blocs

Groupe Algorithmique au Collège - I.R.E.M. de Clermont\*

11 décembre 2019

## Table des matières

<b>1</b>	<b>Initiation</b>	<b>3</b>
1.1	Fenêtres . . . . .	3
1.2	Repère . . . . .	4
1.3	Stylo . . . . .	4
1.4	Angles . . . . .	5
1.5	État du système . . . . .	6
<b>2</b>	<b>Variables</b>	<b>7</b>
2.1	Qu'est ce qu'une variable informatique? . . . . .	7
2.2	Dans Scratch . . . . .	7
2.3	Propriétés de la variable . . . . .	7
<b>3</b>	<b>Structures conditionnelles</b>	<b>8</b>
3.1	Conditions . . . . .	8
3.2	Structures conditionnelles . . . . .	9
<b>4</b>	<b>Boucles</b>	<b>10</b>
4.1	Les boucles pour répéter des instructions un nombre de fois fixé . . . . .	10
4.2	Les boucles pour répéter une instruction jusqu'à ce qu'une condition d'arrêt soit vérifiée . . . . .	11
4.3	Les boucles en programmation événementielle . . . . .	11
<b>5</b>	<b>Listes</b>	<b>12</b>
<b>6</b>	<b>Évènements</b>	<b>13</b>
6.1	Qu'est-ce qu'un évènement en programmation? . . . . .	13
6.2	Les différents types d'évènements dans Scratch. . . . .	13
6.2.1	Évènements déclenchés par l'utilisateur. . . . .	13
6.2.2	Évènements déclenchés par un programme : messages. . . . .	13
<b>7</b>	<b>Blocs</b>	<b>14</b>
7.1	Qu'est-ce qu'un bloc? . . . . .	14
7.2	Bloc simple, bloc à paramètre . . . . .	15

---

\*Joffrey Cottin (Lycée Saint-Pierre, Cusset); Pascal Lafourcade (UCA); Florian Liautaud (Collège Jean-Jacques Soulier, Montluçon); Guy Mas (Collège Joliot-Curie, Aubière); Michaël Meyroneinc-Condy (Lycée Murat, Issoire); Malika More (UCA); Gaëtan Perrin (Collège Jean Vilar, Riom).

## Résumé

Le document ci-après est le fruit du travail du groupe Algorithmique au Collège de l'I.R.E.M. de Clermont.

Il est écrit pour la version 3.0 du logiciel Scratch.

Il est à destination des enseignants du second degré à la recherche de traces écrites pour leurs élèves.

Nous avons essayé de produire un discours le plus scientifique possible, tout en restant accessible pour des élèves. Il ne s'agit ni d'un modèle, ni d'un document officiel.

Nous avons aussi choisi de donner peu d'exemples, afin de permettre aux enseignants et aux élèves de réaliser des documents personnels.

Ce document s'appuie aussi sur deux productions d'autres groupes I.R.E.M., notamment la Commission Inter-Irem Informatique [1] et la Commission Inter-Irem Lycée [2], dont on trouvera les références dans la bibliographie.

# 1 Initiation

## 1.1 Fenêtres

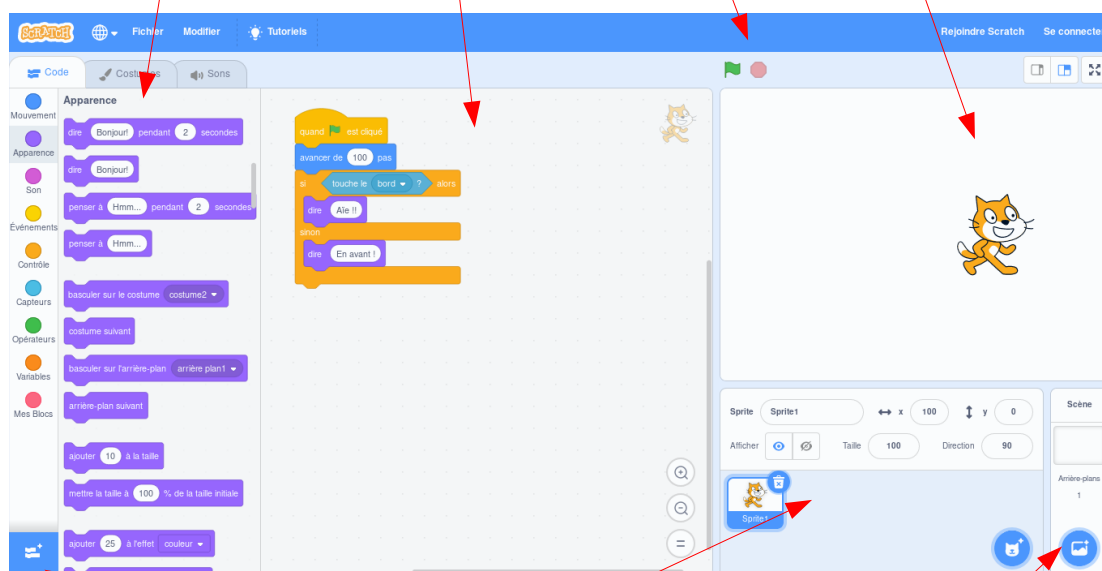
Lorsque l'on ouvre Scratch, on se retrouve devant plusieurs fenêtres dont chacune a sa spécificité.

Les blocs de commandes sont accessibles via différents menus colorés depuis la fenêtre **des blocs**.

**Drapeau vert** et bouton **Stop**.

On glisse des blocs depuis la fenêtre des blocs dans la **fenêtre de programmation**.

Lorsqu'un programme est exécuté, ses effets sont visibles dans la **fenêtre d'exécution** ou **scène**.



C'est dans le **menu des modules** qu'on trouve le **module Stylo**.

Dans la **fenêtre des sprites** on peut sélectionner le sprite dont on modifie les propriétés (script, etc.).

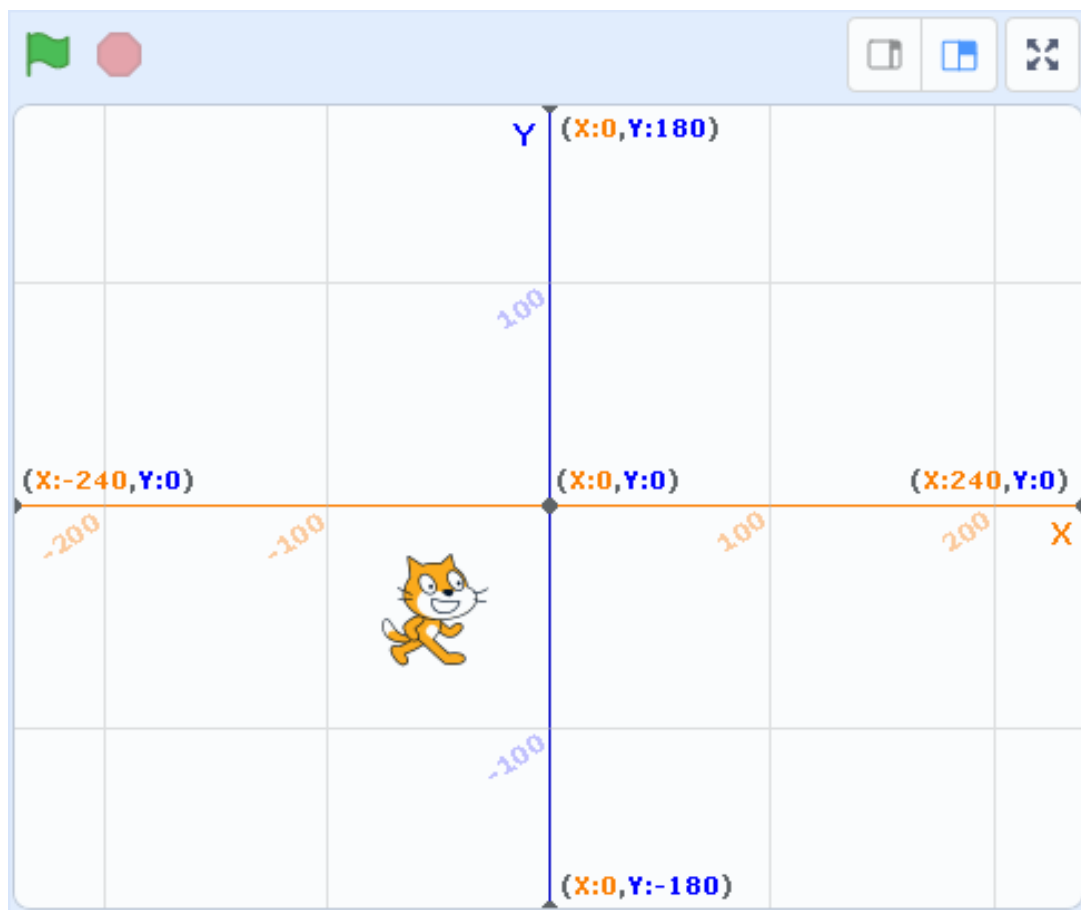
**Sprite** : Personnage à qui on fait exécuter les instructions indiquées par **le script** qui lui est associé.

Dans la **fenêtre des arrière-plans**, on peut sélectionner l'arrière-plan dont on modifie les propriétés (script, etc.).

## 1.2 Repère

La fenêtre d'exécution est un plan repéré, dont l'origine est au centre de la fenêtre. Ses dimensions ne peuvent pas être changées :

- les abscisses vont de  $-240$  à  $240$  ;
- les ordonnées vont de  $-180$  à  $180$ .




Ce repère est utilisé par l'instruction `aller à x:0 y:0`

## 1.3 Stylo

Chaque lutin est muni d'un stylo qui lui permet d'écrire sur la scène (extension à ajouter, voir la capture d'écran ci-dessus). La position de la pointe du stylo correspond aux coordonnées du lutin. Plusieurs paramètres sont associés au stylo (position d'écriture, couleur, taille, intensité). Par exemple :

 mettre la taille du stylo à 1

 relever le stylo

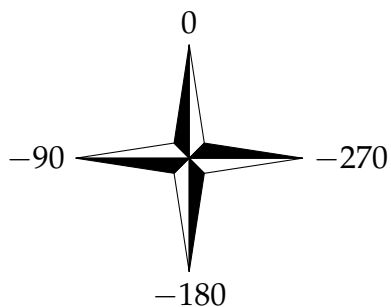
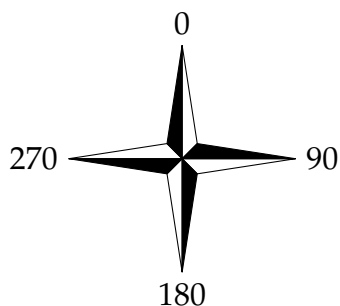
 ajouter 1 à la taille du stylo

## 1.4 Angles

Il existe dans Scratch trois instructions relatives aux angles :

```
s'orienter à 90 degrés
```

Le plan est orienté comme ci-dessous. L'instruction permet de mettre le lutin dans une position fixée par rapport à ce repère.



```
tourner ↻ de 15 degrés
tourner ↻ de 15 degrés
```

Le lutin tourne dans le sens et du nombre de degrés indiqués par le bloc, par rapport à son orientation actuelle.

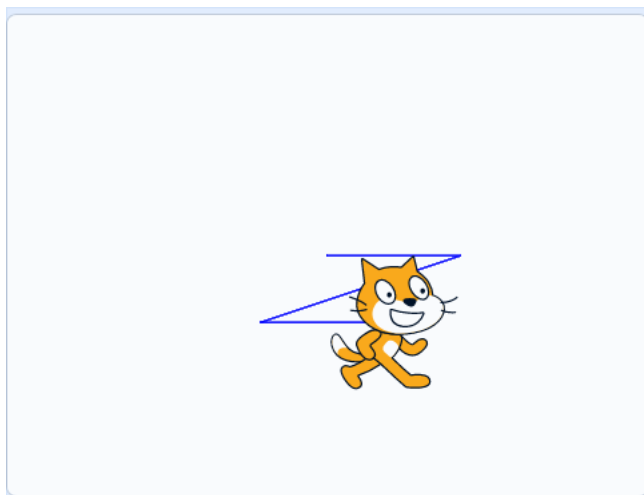
**⚠ À savoir**  
 Pour tracer avec Scratch un angle donné, il faut demander au lutin de pivoter du supplémentaire de cet angle.

*Exemple :* Pour tracer un triangle équilatéral de côté 100, dont on rappelle que tous les angles mesurent  $60^\circ$ , on peut utiliser le programme suivant :

```
1 quand 🚩 est cliqué
2 stylo en position d'écriture
3 répéter 3 fois
4   avancer de 100
5   tourner ↻ de 120 degrés
```

## 1.5 État du système

Lors de l'ouverture du logiciel Scratch, le lutin est un chat, positionné en  $(0;0)$ , orienté à  $90^\circ$  et le stylo est relevé. Par contre, lors de l'ouverture d'un fichier Scratch déjà existant, l'apparence, la position, l'orientation du lutin, etc. sont les mêmes que celles de la sauvegarde. De même, à la fin de l'exécution d'un script, ces valeurs internes de Scratch ne sont pas remises à zéro.



L'image ci-dessus a été obtenue avec l'enchaînement des deux scripts suivants, chacun traçant un segment à un emplacement spécifique.

```

1 quand [drapeau cliqué] est cliqué
2 aller à x: 0 y: 0
3 stylo en position d'écriture
4 avancer de 100

1 quand [espace] est cliqué
2 aller à x: -50 y: -50
3 avancer de 100
  
```

L'état du stylo n'ayant pas changé, le déplacement du lutin pour aller au point  $(-50; -50)$ , est tracé.

## 2 Variables

Généralement, les résultats des opérations effectuées par un ordinateur ne sont pas conservés en mémoire. Si on veut utiliser ou réutiliser une valeur plus tard, il faut la stocker en mémoire à un endroit précis. En informatique cet emplacement est appelé variable.

### 2.1 Qu'est ce qu'une variable informatique ?

**! Définition**

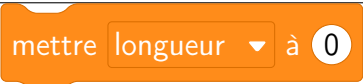



Une variable est un emplacement dans la mémoire d'un ordinateur qui permet de stocker des informations.

Une variable porte un nom permettant de retrouver l'information stockée à cet emplacement. Pour que vos programmes soient plus facile à lire il est important de nommer vos variables avec des noms précis (cote, longueur, angle,...).

### 2.2 Dans Scratch

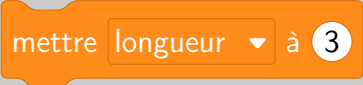
On trouvera les blocs dont il est question dans le menu « Variables » .

Au départ, le menu Variables en contient une, appelée « ma variable », qu'il est possible de renommer.


Bloc	Utilisation
	<b>Affectation</b> : « 0 » est la valeur à mettre en mémoire à l'emplacement nommé « longueur » « 0 » peut être remplacé par le résultat d'un calcul ( $2 + 3$ ), une chaîne de caractères (« cerise ») ou une autre variable (cote).
	<b>Incrémentation</b> : La variable longueur est ici numérique, et on lui ajoute 1.
	La variable et sa valeur sont affichée dans la fenêtre d'exécution.
	La variable et sa valeur sont cachées dans la fenêtre d'exécution.

### 2.3 Propriétés de la variable

**! Affectation d'une valeur**

Lors d'une affectation du type  , la nouvelle valeur remplace l'ancienne , qui, elle, disparaît de la mémoire.

**! Affectation d'une autre variable**

Lors d'une affectation du type  , le contenu de la variable longueur est remplacé par le contenu de la variable cote .  
Pour sa part, le contenu de la variable cote n'a pas changé, il a simplement été copié.

## 3 Structures conditionnelles

En informatique, on utilise une structure conditionnelle lorsqu'il est nécessaire de faire un choix.








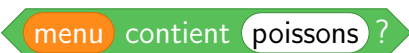
**⚠ Formes**

Il existe deux formes principales :

<p><b>Si</b> « condition » est vraie <b>Alors</b> « séquence d'instructions 1 »</p> <p><b>Sinon</b> « séquence d'instructions 2 »</p> <p><i>Exemple</i> <b>Si</b> le nombre se termine par 0 <b>Alors</b> dire le nombre est divisible par 10</p> <p><b>Sinon</b> dire le nombre n'est pas divisible par 10</p>	<p><b>Si</b> « condition » est vraie <b>Alors</b> « séquence d'instructions »</p> <p><i>Exemple</i> <b>Si</b> le nombre se termine par 0 <b>Alors</b> dire le nombre est divisible par 10</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 3.1 Conditions

Pour faire fonctionner une structure conditionnelle, il faut évaluer si une condition est vraie ou fausse. Dans Scratch, les opérateurs de comparaison ont une forme hexagonale. On les nomme « Prédicats » .

<i>Menu Opérateurs</i>	
	Teste l'égalité des deux valeurs, qui peuvent être des variables, et renvoie ici « faux ».
	Teste si la première valeur est plus petite que la seconde valeur et renvoie ici « vrai ».
	Teste si la valeur de la variable « cote » est plus grande que la valeur de la variable « longueur ».
	Teste la présence du caractère « r » dans la chaîne de caractères « patate » et renvoie ici « faux ».
	Teste si les deux conditions sont vraies en même temps (pour 1972 dans « annee », renvoie « vrai »).
	Teste si au moins l'une des deux conditions est vraie (pour 1900 dans « annee », renvoie « vrai »).
	Teste si la condition est fausse (pour 1972 dans « annee », renvoie « faux »).
<i>Menu Variables</i>	
	Teste si la liste « menu » contient la chaîne de caractères « poissons ».



Menu Capteurs	
	Teste si le lutin touche le pointeur de la souris et renvoie « vrai » si c'est le cas.
	Teste si le lutin touche un élément de la couleur choisie et renvoie « vrai » si c'est le cas.
	Teste si la partie colorée d'un lutin touche une autre partie colorée d'un autre objet et renvoie « vrai » si c'est le cas.
	Teste si la touche du clavier sélectionnée est pressée et renvoie « vrai » si c'est le cas.
	Teste si un bouton de la souris est pressé et renvoie « vrai » si c'est le cas.

### 3.2 Structures conditionnelles

On trouvera les blocs dont il est question dans le menu « Contrôle » .

*Exemple :* Écrire un programme qui demande à l'utilisateur le produit de 5 par 7.

```

1 quand [drapeau cliqué]
2 demander [Quel est le produit de 5 par 7 ?]
3 Si [réponse = 35] alors
4   dire [Bravo !]
5 sinon
6   dire [Désolé !]
  
```

```

1 quand [drapeau cliqué]
2 demander [Quel est le produit de 5 par 7 ?]
3 Si [réponse = 35] alors
4   dire [Bravo !]
  
```

*Exemple :* Écrire un programme Scratch qui demande un nombre entier à l'utilisateur et qui fait dire au lutin si le nombre est strictement positif ou non.

```

1 quand [drapeau cliqué]
2 demander [Entrer un nombre !]
3 mettre [nombre] à [réponse]
4 si [nombre > 0] alors
5   dire [Ce nombre est strictement positif !] pendant 2 secondes
6 sinon
7   si [nombre = 0] alors
8     dire [Ce nombre est nul !] pendant 2 secondes
9   sinon
10    dire [Ce nombre est strictement négatif !] pendant 2 secondes
  
```

## 4 Boucles

Une boucle est une structure de contrôle qui permet de répéter une instruction (ou une séquence d'instructions) plusieurs fois.

Dans Scratch, les blocs « Boucles » se trouvent dans l'onglet **Contrôle**.

### 4.1 Les boucles pour répéter des instructions un nombre de fois fixé

Le bloc utilisé dans ce cas est : répéter  $n$  fois



Toutes les instructions placées à l'intérieur de ce bloc seront répétées  $n$  fois où  $n$  peut être une variable.

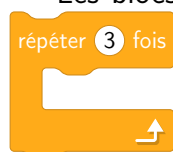
*Exemple :*

Pour réaliser la frise ci-dessous, on peut envisager la construction de deux façons :



Méthode 1	Méthode 2
1 Quand  est cliqué	1 Quand  est cliqué
2  stylo en position d'écriture	2  stylo en position d'écriture
3 tourner  de 60 degrés	3 tourner  de 60 degrés
4 avancer de 100	4 répéter 3 fois
5 tourner  de 120 degrés	5 avancer de 100
6 avancer de 100	6 tourner  de 120 degrés
7 tourner  de 120 degrés	7 avancer de 100
8 avancer de 100	8 tourner  de 120 degrés
9 tourner  de 120 degrés	
10 avancer de 100	
11 tourner  de 120 degrés	
12 avancer de 100	
13 tourner  de 120 degrés	
14 avancer de 100	
15 tourner  de 120 degrés	

Les blocs 4 à 7 de la méthode 1. se répètent trois fois et peuvent donc être insérés à l'intérieur du bloc



## 4.2 Les boucles pour répéter une instruction jusqu'à ce qu'une condition d'arrêt soit vérifiée

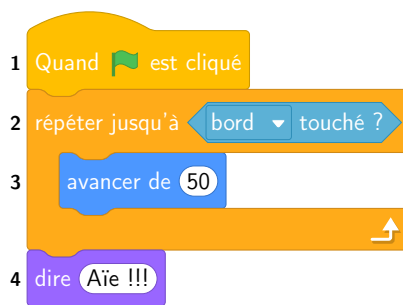
Le bloc utilisé dans ce cas est :



Toutes les instructions placées à l'intérieur de ce bloc seront répétées jusqu'à ce que la condition soit vérifiée.

*Exemple :*

Quelles que soient la position et l'orientation du lutin, si l'on répète l'instruction **avancer de 50**, celui-ci touchera l'un des bords après un certain nombre de répétition. On ne sait pas, *a priori*, combien de fois il faut répéter l'instruction **avancer de 50** avant de toucher un bord.



## 4.3 Les boucles en programmation événementielle

Le bloc utilisé dans ce cas est :



Toutes les instructions placées à l'intérieur de ce bloc seront répétées lors de l'exécution du programme jusqu'à son interruption par l'utilisateur via le bouton « Arrêt ».

*Exemple :*

Déplacer le lutin à l'aide des flèches directionnelles :










## 5 Listes

En informatique, une liste contient des informations rangées dans un certain ordre, ce qui permet de les retrouver plus facilement.

On trouvera les blocs dont il est question dans le menu « Variables » .

Dans Scratch, il faut créer une liste dans le menu associé avant de voir apparaître les instructions que l'on peut utiliser.

Bloc	Explications
	Ajoute la chaîne de caractère « lait » en dernière position de la liste « Courses ».
	Insère en position 4 de la liste « Somme » le résultat de « 2+5 » et renumérote la suite de la liste.
	Remplace la valeur en position 7 de la liste « Telephone » par la valeur « 04 73 01 02 03 ».
	Supprime l'élément en position 2 et renumérote la suite de la liste « nombres ».
	Teste si le nombre 18 est un élément de la liste « nombres ».
	Renvoie la valeur en position 4 de la liste « Telephone ».
	Renvoie la longueur de la liste « Telephone », c'est à dire le nombre d'élément dans cette liste.

*Exemple :*

On souhaite créer un petit programme qui conjugue les verbes du deuxième groupe au présent de l'indicatif.

On crée donc une liste des terminaisons, et une variable compteur pour la boucle. On utilise alors le programme suivant.

```

quand [drapeau] est cliqué
demander [Donne le radical d'un verbe du 2e groupe] et attendre
mettre [compteur] à 1
répéter 6 fois
  dire [regrouper reponse] élément [compteur] de terminaisons
  ajouter à [compteur] 1
  
```

terminaisons	
1	s
2	s
3	t
4	issons
5	issez
6	issent
+ longueur 6 =	

## 6 Évènements

### 6.1 Qu'est-ce qu'un évènement en programmation ?

Dans le cas de Scratch, on parle de programmation évènementielle, c'est à dire que le programme attend qu'il se passe quelque chose avant d'exécuter une séquence d'instructions programmée.

Les évènements les plus simples sont **Quand  est cliqué** ou **Quand  est pressé**.

Dans Scratch, les évènements sont toujours placés en début de script, leur forme imposant ce fonctionnement.

### 6.2 Les différents types d'évènements dans Scratch.

#### 6.2.1 Évènements déclenchés par l'utilisateur.

##### Interventions de l'utilisateur

Cette catégorie regroupe les déclencheurs qui imposent une action de l'utilisateur, qui peut être un clic sur le drapeau vert, ou sur un lutin précis, ou alors un appui sur une touche du clavier.

**Attention !** Lorsque l'on crée deux séquences différentes dans le même programme avec le même évènement déclenchant, l'ordre dans lequel Scratch effectue les instructions n'étant pas toujours le même, le résultat de l'exécution peut ne pas être celui qui était attendu.

#### 6.2.2 Évènements déclenchés par un programme : messages.

##### Déclenchement interne

Scratch permet l'envoi de messages entre lutins. Il suffit d'insérer la commande **envoyer à tous**  dans un script. Tous les lutins que l'on veut faire réagir à l'envoi du message doivent avoir un script comportant l'évènement **quand je reçois**  en première instruction.

*Exemple :*

Dans un jeu, on souhaite que le lutin 2 avance lorsque le lutin 1 a touché le bord de la scène. Voici un exemple de séquences d'instructions possibles pour cette situation.

Lutin 1	Lutin 2
<ol style="list-style-type: none"> <li>1 répéter jusqu'à ce que <input type="text" value="touche le bord"/> ?</li> <li>2 avancer de <input type="text" value="100"/></li> <li>3 envoyer à tous <input type="text" value="arretL1"/></li> </ol>	<ol style="list-style-type: none"> <li>1 quand je reçois <input type="text" value="arretL1"/></li> <li>2 avancer de <input type="text" value="100"/></li> </ol>

**Remarque :** On peut créer de nouveaux messages, par exemple en leur donnant des noms explicites, comme pour l'exemple ci-dessus « ArretL1 » au lieu de « message 1 » pour faciliter la lecture.

## 7 Blocs

En programmation, il est utile de simplifier son code le plus possible, pour qu'il soit lisible et compréhensible par quelqu'un d'autre.

Plutôt que de réécrire des séquences d'instructions identiques à plusieurs endroits d'un script, dans Scratch, on va utiliser les « blocs » pour faire ce travail.

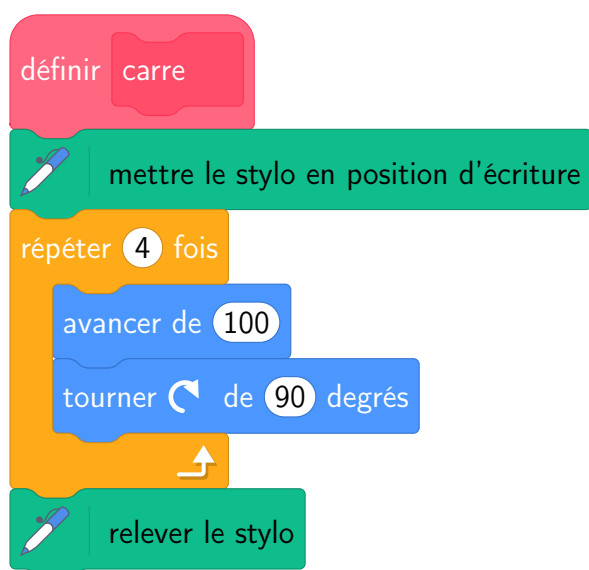
### 7.1 Qu'est-ce qu'un bloc ?

#### Définition

Un bloc est une partie de programme que l'on a nommée et mise à part du programme principal.

Dans Scratch, les blocs sont à définir dans le menu « Mes Blocs ». Ensuite, pour construire le nouveau bloc on utilise les mêmes instructions que dans un script normal. Ce nouveau bloc génère une nouvelles instruction utilisable pour écrire des scripts.

Le bloc ci-dessous permet de faire tracer un carré au lutin.



On peut l'utiliser dans le programme principal de la manière suivante.



## 7.2 Bloc simple, bloc à paramètre

Dans l'exemple précédent, le bloc est figé, et il est simplement exécuté, sans être modifié, 10 fois par le programme principal. On peut introduire un paramètre afin d'obtenir un bloc dont le comportement diffère selon la valeur donnée en entrée.

Le bloc ci-dessous permet de faire tracer au lutin un carré de côté « cote\_carre », qui représente la longueur donnée par l'utilisateur.

```

définir carre cote_carre
mettre le stylo en position d'écriture
répéter 4 fois
  avancer de cote_carre
  tourner de 90 degrés
relever le stylo
  
```

On peut l'utiliser dans le programme principal de la manière suivante :

```

quand est cliqué
relever le stylo
effacer tout
aller à x: -200 y: 0
s'orienter à 90
demander Longueur du côté ? et attendre
répéter 10 fois
  carre réponse
  tourner de 36 degrés
  
```

## Références

- [1] Sylvie ALAYRANGUES et al. "Une analyse des exercices d'algorithmique et de programmation du brevet 2017." In : *Repères IREM* Numéro 116 (2019). URL : <https://hal.archives-ouvertes.fr/hal-02077738>.
- [2] Emmanuel BEFFARA, Malika MORE et Cécile PROUTEAU. *Algorithmique et programmation au cycle 4 : Commentaires et recommandations du groupe Informatique de la CII Lycée*. 2017. URL : <http://www.univ-irem.fr/IMG/pdf/algoetprogaucycle4ciilycee-2.pdf>.