



# Synthèse : Programmation sous Scratch

## ALGORITHME ET PROGRAMME :

Algorithme d'un programme sous la forme d'un texte et d'un organigramme		Le programme sous forme de blocs avec Scratch/mBlock correspondant à ces algorithmes.
<p><i>Début</i>  <b>SI</b> couleur jaune touchée  <b>ALORS</b> Afficher « Bravo »,  Attendre 2 secondes,  Retour <i>Début</i></p>		

Un algorithme sous la forme d'un organigramme est toujours composé des éléments suivants :

Un ovale qui correspond au <b>Début</b> ou <b>Fin</b> (si fin il y a) de l'organigramme.	Correspond à <b>une question</b> à laquelle on peut <b>répondre uniquement</b> par : <b>OUI</b> ou <b>NON</b>	Correspond à <b>une action</b> à effectuer.

## CONSTITUTION D'UN SCRIPT (PROGRAMME) :

**DÉBUT** du programme

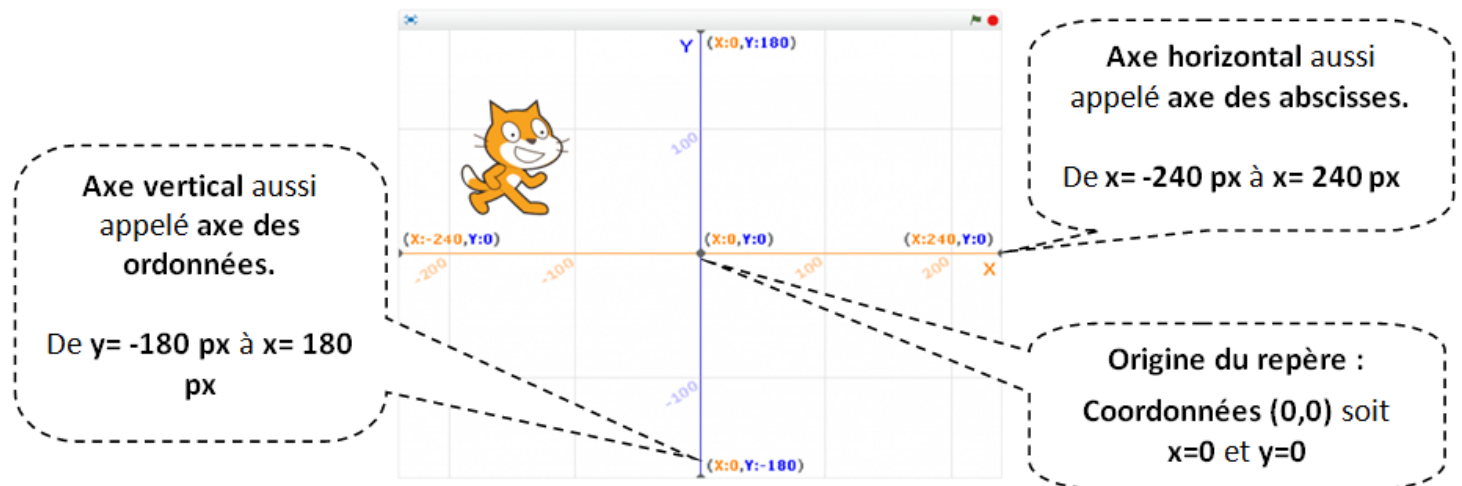
**INITIALISATION**

- permet de positionner le lutin à gauche,
- oriente le lutin vers la droite (90°),
- efface l'écran.

**PROGRAMME PRINCIPAL**

Un programme est toujours constitué : d'un **début**, d'une **initialisation** et du **programme principal**.

## LES COORDONNÉES :

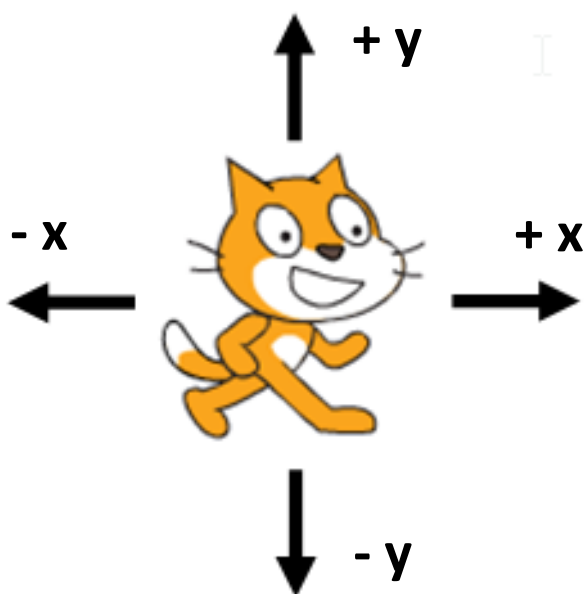


Le **pixel** (souvent abrégé **px**) est l'unité de base permettant de mesurer la définition d'une image numérique.



## LES MOUVEMENTS

s'orienter à 90 ▼	s'orienter à -90 ▼	s'orienter à 0 ▼	s'orienter à 180 ▼


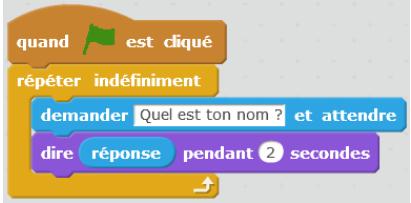
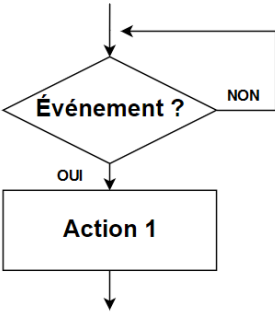
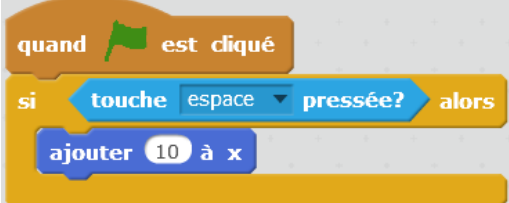

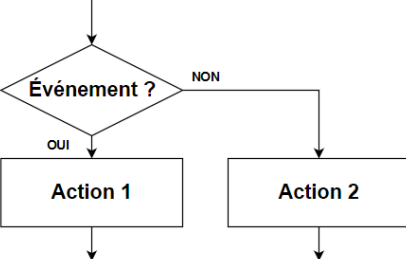
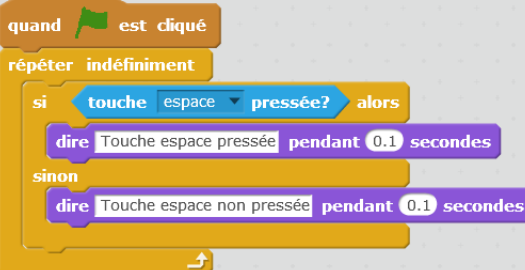
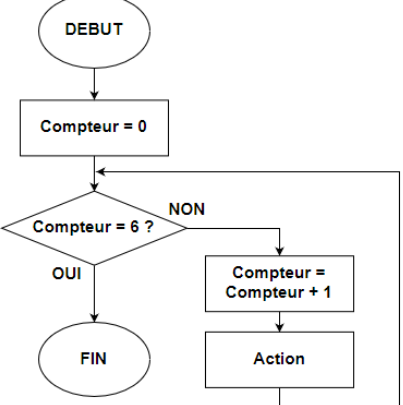

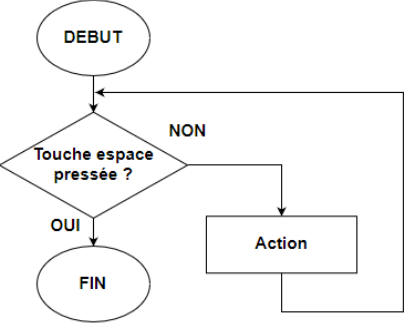



```

quand [drapeau] est cliqué
  s'orienter à 90 ▼
  aller à x: 0 y: 0
  répéter indéfiniment
    si [touche flèche droite ▼] pressée? alors
      ajouter 10 à x
    si [touche flèche gauche ▼] pressée? alors
      ajouter -10 à x
    si [touche flèche haut ▼] pressée? alors
      ajouter 10 à y
    si [touche flèche bas ▼] pressée? alors
      ajouter -10 à y
  
```

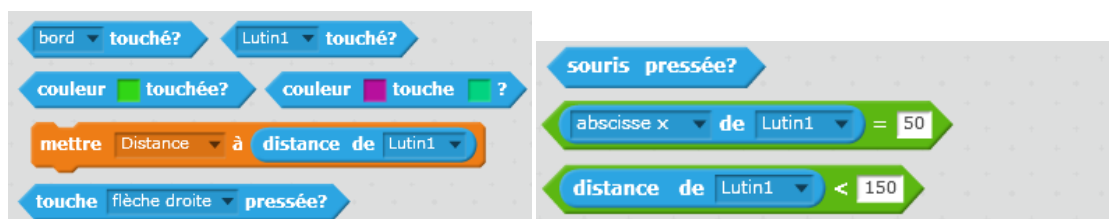
## LES BOUCLES :

Les boucles constituent un moyen de répéter un certain nombre de fois des instructions d'un programme.

<p><b>BOUCLE INFINIE</b></p>		
<p><b>BOUCLE DE TEST « SI...ALORS... »</b></p> <p>Si ... « Évènement » Alors ... « Action »</p>		 <p>On peut rajouter « RÉPÉTER INDÉFINIMENT » pour répéter l'action sans avoir à cliquer sur le </p>
<p><b>BOUCLE DE TEST « SI...ALORS...SINON... »</b></p> <p>Si ... « Évènement » Alors ...« Action 1 » Sinon ...« Action 2 »</p>		
<p><b>BOUCLE QUI COMPTE</b></p> <p>Elle utilise un <b>COMPTEUR</b> (de 1 à 6 dans notre exemple) pour répéter 6 fois l'action.</p>		 <p>Le lutin avancera 10 fois de 25 px soit 250 px en tout.</p>
<p><b>BOUCLE QUI SE RÉPÈTE JUSQU'À...</b></p> <p>Cette boucle se <b>répète jusqu'à ce qu'un évènement arrive</b>.</p>		 <p>Tant que la touche espace ne sera pas pressée, le lutin tounera de 15°.</p>

## LES TESTS :

Les tests permettent de détecter des évènements et selon les résultats d'enchaîner des actions.



## LES OPERATEURS :

Deux formes de blocs :

Des hexagonaux.		Pour comparer des valeurs.
Et d'autres avec les coins arrondis.		Pour faire une opération.

	<p>Si « abscisse x » du lutin « lutin1 » &gt; 160  <b>Alors</b> dire « Gagné » pendant 2 secondes.</p>
	<p>Répéter 5 fois (pour une figure de 5 cotés)  <b>Avancer</b> de 10 puis  <b>Tourner</b> vers la gauche 360°/5</p>

## LES VARIABLES :

Les variables permettent de stocker une valeur qui pourra être utilisée par le programme.

Variables créées	Manipulations sur les variables possibles
	 Stocker « 100 » dans la variable « Distance »
	 Stocker « la distance jusqu'au lutin Banane » dans la variable « Distance »
	 Stocker « la variable Nombre de tours » dans la variable « Distance »
	 Ajouter la réponse à la variable « Distance »
	 Mettre la variable distance à 0 (ou autre).
	 Comparer une variable.