Fiche

I. Analyser et modifier un programme informatique

Sans programme, les ordinateurs, consoles de jeux, robots, smartphones ou objets connectés ne pourraient pas fonctionner correctement. Un programme est une suite d'instructions logiques qui permettent à une machine d'exécuter une tâche précise. Ces instructions sont écrites en lignes comme dans un traitement de texte, dans des langages comme Python, ou écrites en blocs visuels comme dans le langage Scratch.

Analyser un programme signifie observer toutes ses lignes, c'est-à-dire sa structure, ses instructions et ses résultats afin de vérifier qu'il correspond bien au besoin défini. Par exemple, un programme en langage Scratch peut faire avancer un personnage (*sprite*) de 10 pas lorsque l'utilisateur appuie sur la touche flèche droite. Le besoin est alors clair : déplacer le personnage sur l'écran. L'analyse consiste à tester le programme, observer le déplacement et vérifier qu'il est conforme aux attentes.

On peut par exemple analyser un jeu programmé en Scratch. Pour cela il faut identifier les blocs utilisés : blocs de mouvement (avancer, tourner), blocs de contrôle (répéter, attendre), blocs d'événements (lorsqu'on clique sur le drapeau vert). Chaque bloc est articulé l'un avec l'autre pour former le programme complet.

Cette analyse permet aussi de repérer les erreurs du programme (bugs ou bogues) : par exemple, un personnage qui avance sans s'arrêter car aucune condition n'a été définie pour limiter son déplacement. C'est une étape obligatoire avant de passer à la modification ou à la création.

Modifier un programme permet de l'adapter à de nouveaux besoins ou de corriger des erreurs. Cette étape montre que la programmation est un processus évolutif, où les idées s'améliorent progressivement. Par exemple, un programme qui fait avancer un *sprite* peut être enrichi pour qu'il change de couleur ou d'aspect lorsqu'il court ou pour qu'il saute lorsqu'on appuie sur la touche espace. Le plus souvent, la modification consiste en un débogage qui teste le programme, identifie les erreurs et les corrige.

II. Réaliser des boucles dans un programme simple

Dans le monde professionnel, la programmation ne se limite pas à Scratch. Elle s'étend à des langages plus complexes comme Python, C++ ou Java. Mais les principes restent les mêmes : analyser, modifier, tester, corriger. Dans la vie quotidienne, les applications de domotique, les voitures autonomes ou les thermostats connectés fonctionnent tous grâce à des programmes qui suivent cette logique.

Pour commencer à apprendre et comprendre la programmation, Scratch est un excellent moyen. Une des principales fonctions de Scratch, et de tous les programmes informatiques, est la boucle.

Voici un exemple de programmation de boucle avec un programme simple qui permet à son utilisateur de créer graphiquement un carré en faisant avancer un *sprite* au moyen des touches « a » et « espace » du clavier. La touche « a » permet de tourner et la touche « espace » de tracer un trait.

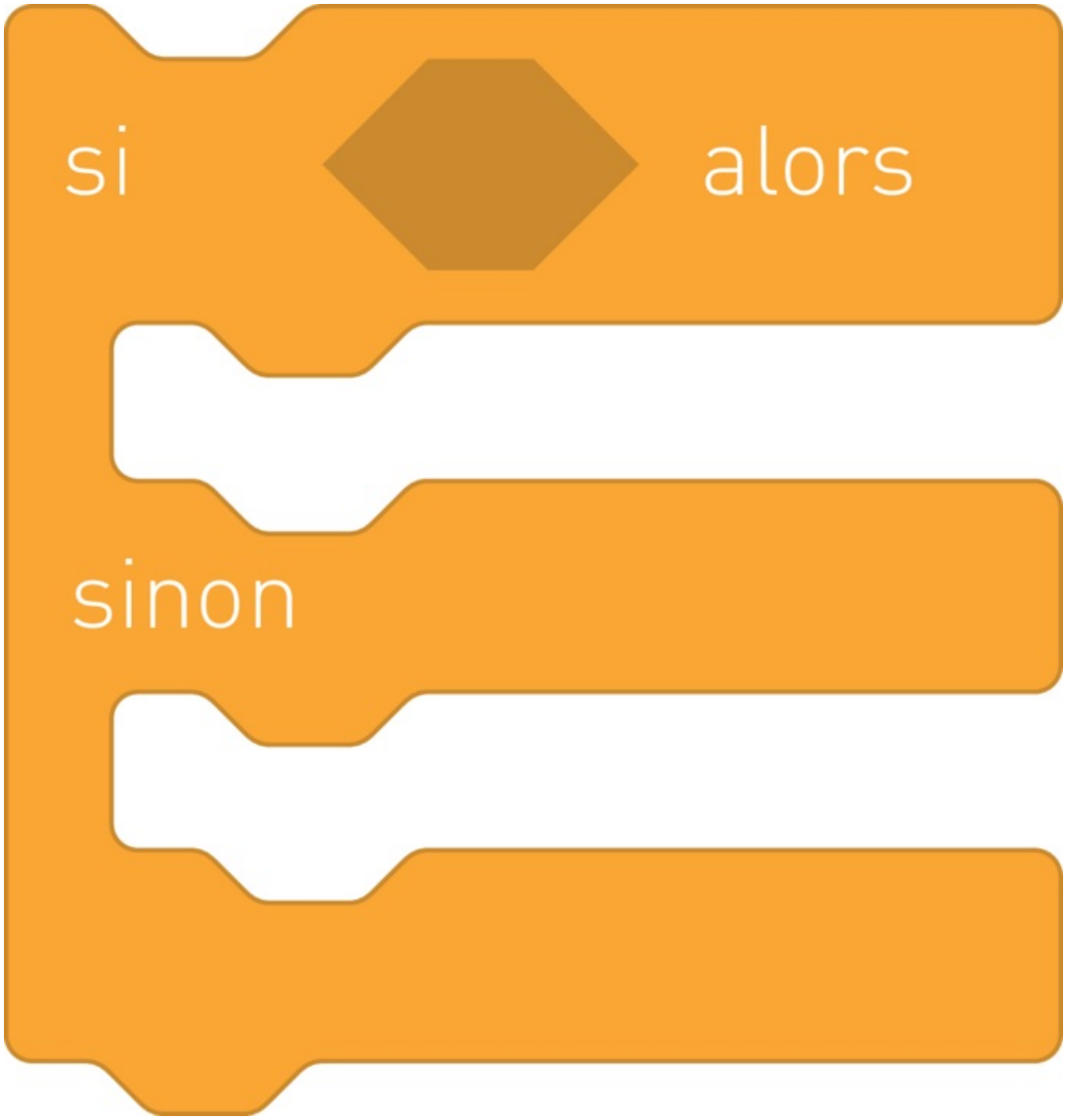
1. Création d'une boucle-répétition

On utilise la fonction « répéter 10 fois » en paramétrant le nombre de répétitions, puis on place les actions à répéter à l'intérieur de la « boucle ». Par exemple, pour dessiner un carré, on répétera 4 fois le tracé du même segment et la rotation de 90°.

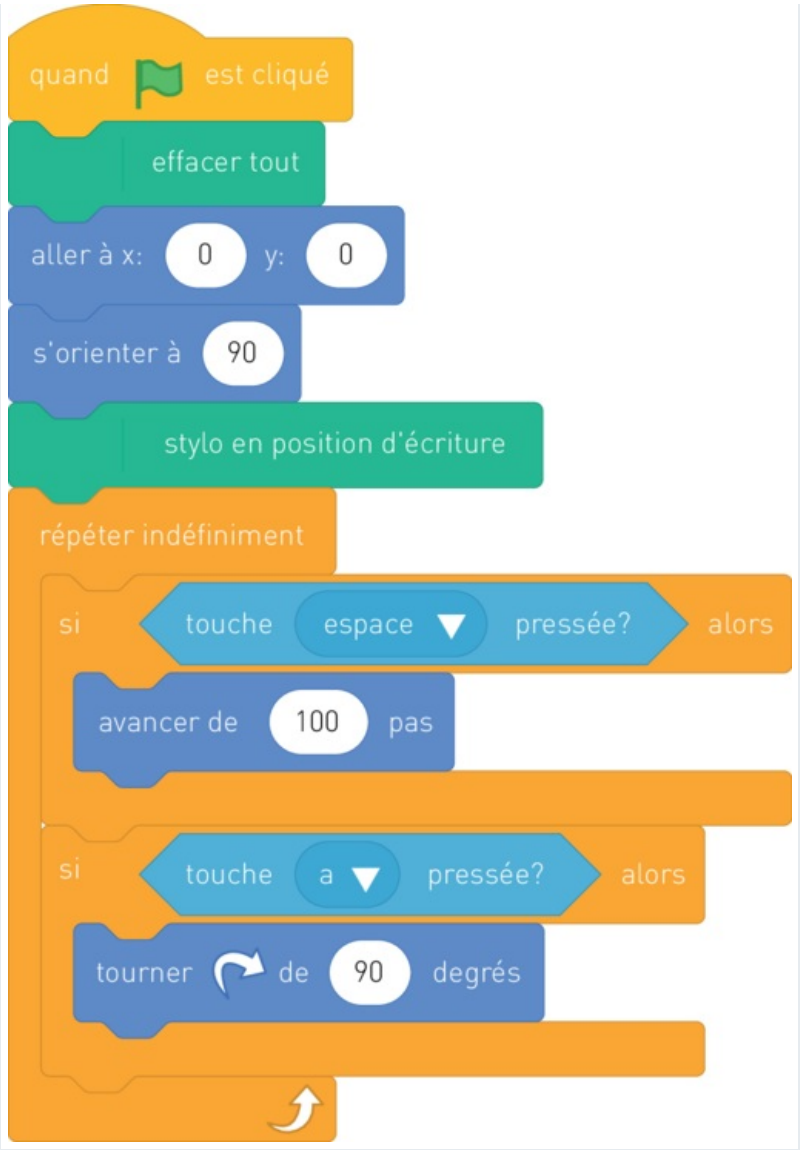
2. Création d'une boucle avec condition

On utilise la fonction « répéter indéfiniment » : les actions à répéter à l'intérieur seront répétées indéfiniment, il faut donc établir une « condition » à valider à l'intérieur de cette boucle pour réaliser les actions.

Pour établir une condition, on utilise :



- Soit une fonction « si » : si la condition est vraie, exécute la pile de commandes situées à l'intérieur de la boucle. Sinon, rien ne se passe.
- Soit une fonction « si-sinon » : si la condition est vraie, exécute la pile de commandes situées à l'intérieur de la boucle de la partie supérieure, sinon, exécute la pile de commandes situées à l'intérieur de la boucle de l'autre partie.



À retenir :

1. Analyser un programme aide à comprendre sa logique et à détecter les erreurs éventuelles.
2. **Modifier** un programme permet de **l'adapter** à de nouveaux **besoins** et de le rendre plus **riche et interactif**.
3. **Réaliser** un programme simple développe des **compétences logiques** et montre le lien entre **théorie et pratique**.
4. Le **débogage** est une **étape** essentielle qui apprend à **corriger** ses erreurs et à **améliorer** ses productions.

Définitions importantes :

Programme informatique : Suite d'instructions permettant à un ordinateur d'accomplir une tâche précise.
Scratch : Langage de programmation visuel et intuitif (sous forme de blocs d'instructions), utilisé pour initier les élèves à la programmation.
Sprite : Traduction de l'anglais « lutin ». Personnage ou objet animé utilisé dans Scratch, consistant en un élément pouvant se déplacer à l'écran selon les instructions programmées.
Débogage : Action de repérer et corriger les erreurs d'un programme (bug ou bogue).
Algorithme : Partie d'un programme informatique consistant en une suite d'instructions logiques organisées pour résoudre un problème ou effectuer une tâche précise au sein du programme.
Condition : Instruction qui permet d'exécuter une action seulement si une situation donnée est vraie.
Boucle : Instruction qui répète une série d'actions tant qu'une condition est remplie.