# Construire le plan de réalisation d'un objectif complexe

## En utilisant Scratch (V. 3.29), langage de programmation graphique

Sébastien Nowak – Septembre 2023

# Bienvenue !

Vous êtes ici

# Période préparatoire : aborder la formation de formateur numérique avec les bons outils et la bonne posture

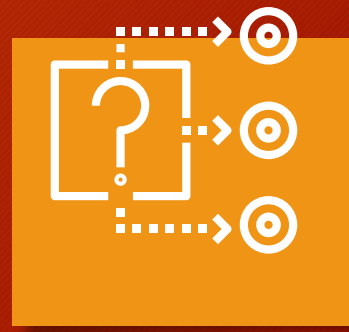
| N° Fiche AT | Activités types | N° Fiche CP | Compétences professionnelles |
|---|---|---|---|
| 1 | Concevoir et préparer la formation | 1 | Elaborer la progression pédagogique d'une formation multimodale à partir d'une demande |
| | | 2 | Concevoir un scénario pédagogique et d'accompagnement en intégrant la multimodalité |
| | | 3 | Concevoir des activités d'apprentissage et d'évaluation en intégrant la multimodalité |
| 2 | Animer une formation et évaluer les acquis des apprenants | 4 | Animer une formation et faciliter les apprentissages selon différentes modalités |
| | | 5 | Evaluer les acquis de formation des apprenants |
| | | 6 | Remédier aux difficultés individuelles d'apprentissage |
| 3 | Accompagner les apprenants en formation | 7 | Accompagner les apprenants dans leur parcours de formation |
| | | 8 | Accueillir un apprenant en formation et co-construire son parcours |
| | | 9 | Tutorer les apprenants à distance |
| | | 10 | Accompagner le développement professionnel des apprenants |
| 4 | Inscrire sa pratique professionnelle dans une démarche de qualité et de responsabilité sociale des entreprises | 11 | Respecter et faire respecter la règlementation en vigueur en formation et dans sa spécialité |
| | | 12 | Réaliser une veille pour maintenir son expertise de formateur et de professionnel dans sa spécialité |
| | | 13 | Analyser ses pratiques professionnelles |

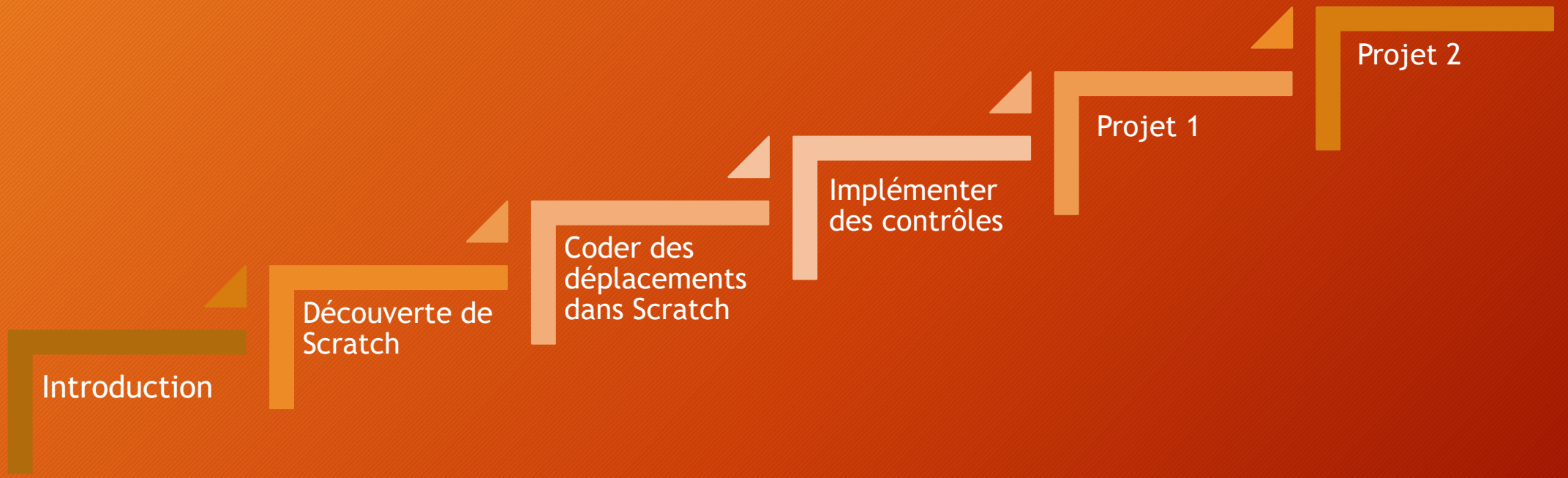
# Période préparatoire : aborder la formation de formateur numérique avec les bons outils et la bonne posture

| | |
|---|---|
| Séance 0 | Fondements collectifs : référentiels et règles de fonctionnement (1j) |
| Séance 1 | Le métier de formateur numérique (2j) |
| Séance 2 | Pratique de l'écrit et de l'oral (2j) |
| Séance 3 | Démarche de réalisation d'un objectif complexe (2j) |
| Séance 4 | Les outils professionnels du formateur numérique (1j) |
| Séance 5 | Apprendre à apprendre (2j) |

Vous êtes ici

Simuler la réalisation d’un objectif complexe, en utilisant un langage de programmation simple, et en respectant le besoin exprimé




Introduction
Découverte de Scratch
Coder des déplacements dans Scratch
Implémenter des contrôles
Projet 1
Projet 2

# Plan

# Quelques questions vers vous

- Vous a-t-on déjà fixé des objectifs complexes ?
- Si oui, quels étaient-ils ?
- Comment avez-vous procédé pour les relever ?
- Connaissez-vous le codage ?
- Quelle serait votre définition du codage ?
- Connaissez-vous Scratch ?
- Si je vous disais que créer un code est analogue à la construction pédagogique d'une action de formation, me croiriez-vous ?

# Introduction : qu’est-ce que le codage ?

Ensemble des moyens utilisés pour mettre en forme des informations, afin de pouvoir les manipuler, les stocker, ou les transmettre. Le codage ne s’intéresse pas au contenu (le sens) d’une information, mais à sa forme et à son volume.

Par extension, le codage désigne aussi tout procédé qui vise à transposer des données ou des informations d’une forme vers une autre, sans perte de sens.

# Introduction : qu’est-ce que le codage ?

Bienvenue chez Pop School

295142151420
538526161516
1938151512

# Le codage informatique (1/4)

Un ordinateur ne comprend quasiment rien au langage humain. Il possède son propre langage : le langage machine. C’est le seul que le processeur reconnaisse

Le langage machine est basiquement une suite de bits (des 0 et des 1) que le processeur lit et interprète.

# Le codage informatique (2/4)

« *L'ordinateur est complètement con.* »

Gérard Berry

Il ne fait rien d'intelligent par lui-même (pas encore du moins). Il faut lui donner des instructions pour chaque tâche qu'on souhaite lui confier.
Ces instructions doivent être simples. Le champ des possibles d'un processeur reste limité, quelle que soit sa puissance.

# Le codage informatique (3/4)

On est donc dans une situation où l'ordinateur ne reconnait pas notre langage, et où le commun des mortels ne comprend absolument rien au langage machine.

Bref, il va falloir un peu de traduction. C'est la finalité du codage informatique.

# Le codage informatique (4/4)

Langage naturel

Code/programme source

Code exécutable, lisible et traitable par le processeur

Codage informatique
Programmation

Compilation

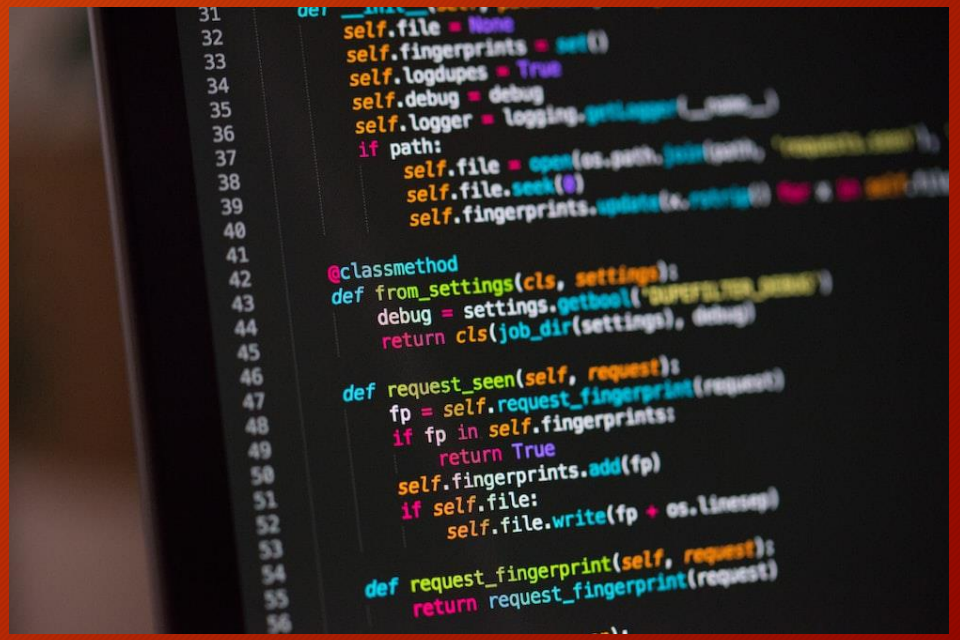
Langage de programmation

# En résumé…

Le codage informatique est une première étape de formalisation des instructions données à un ordinateur.

On utilise pour coder un langage de programmation. Chaque langage possède ses règles et ses conventions. Chaque langage a ses utilisations préférentielles.

Parmi les langages les plus connus et utilisés : Python, html, Java, JavaScript, C, C++…



# Un peu de C++

```cpp
#include <iostream>
#include <string>

template <typename Traits>
class Information{
    private:
        Traits pf_name;         // fp == public field name
        Traits pf_family;       // fp == public field family

    public:
        void set_information(Traits arg_name, Traits arg_family){
            pf_name = arg_name;
            pf_family == arg_family;
        }

        void get_information(){
            std::cout << "Your name is " << pf_name << " " << pf_family << std::endl;
        }
};

auto main(int argc, char* argv[]) -> decltype(0){
    Information<std::string> o_person;

    o_person.set_information("Milad", "Kahsari Alhadi");
    o_person.get_information();

    return 0;
}
```

# Un peu de html

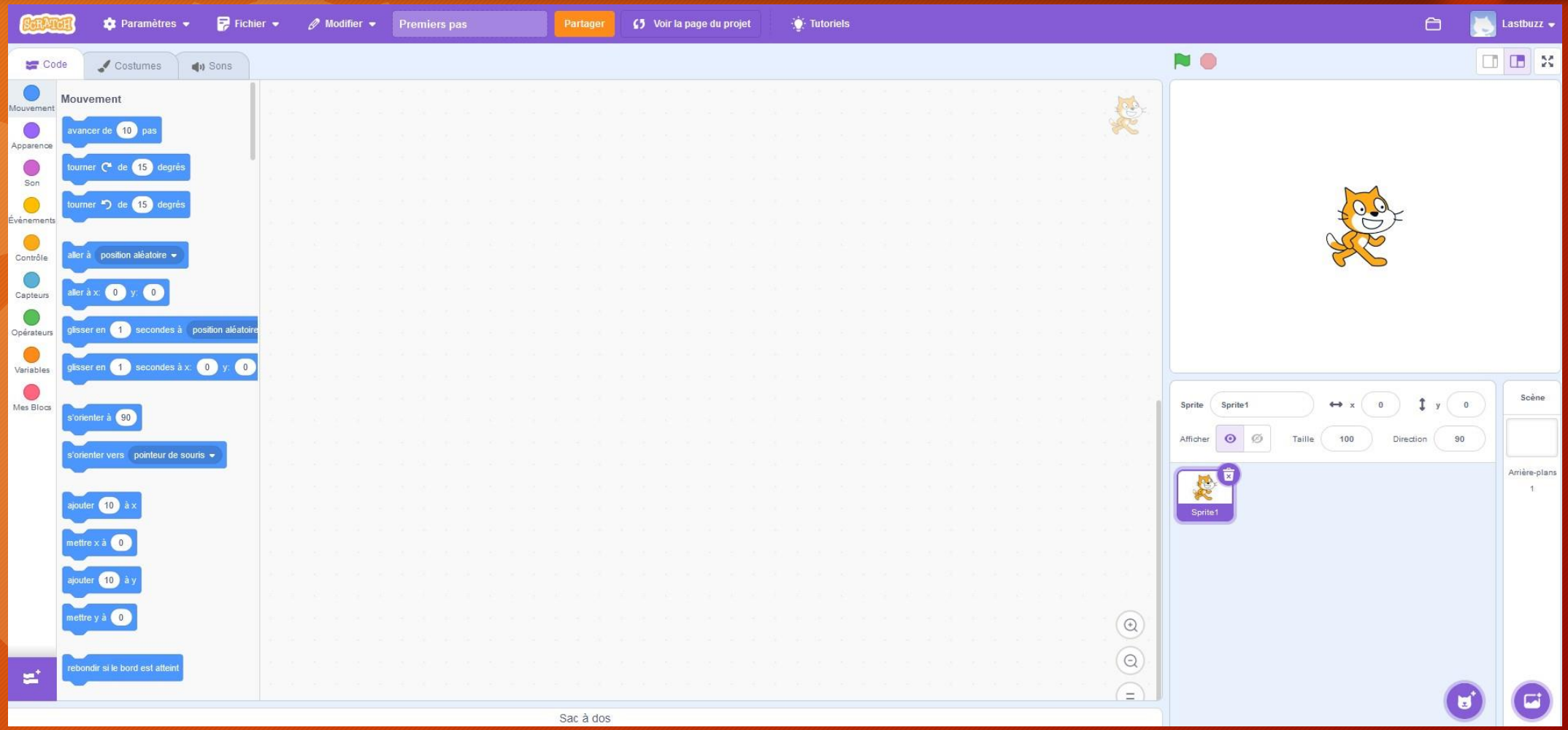
```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="utf-8" />
5       <title>Ma page</title>
6       <link href="/style.css" rel="stylesheet" />
7     </head>
8     <body>
9      Ma ligne à commenter
10    </body>
11  </html>
12
```

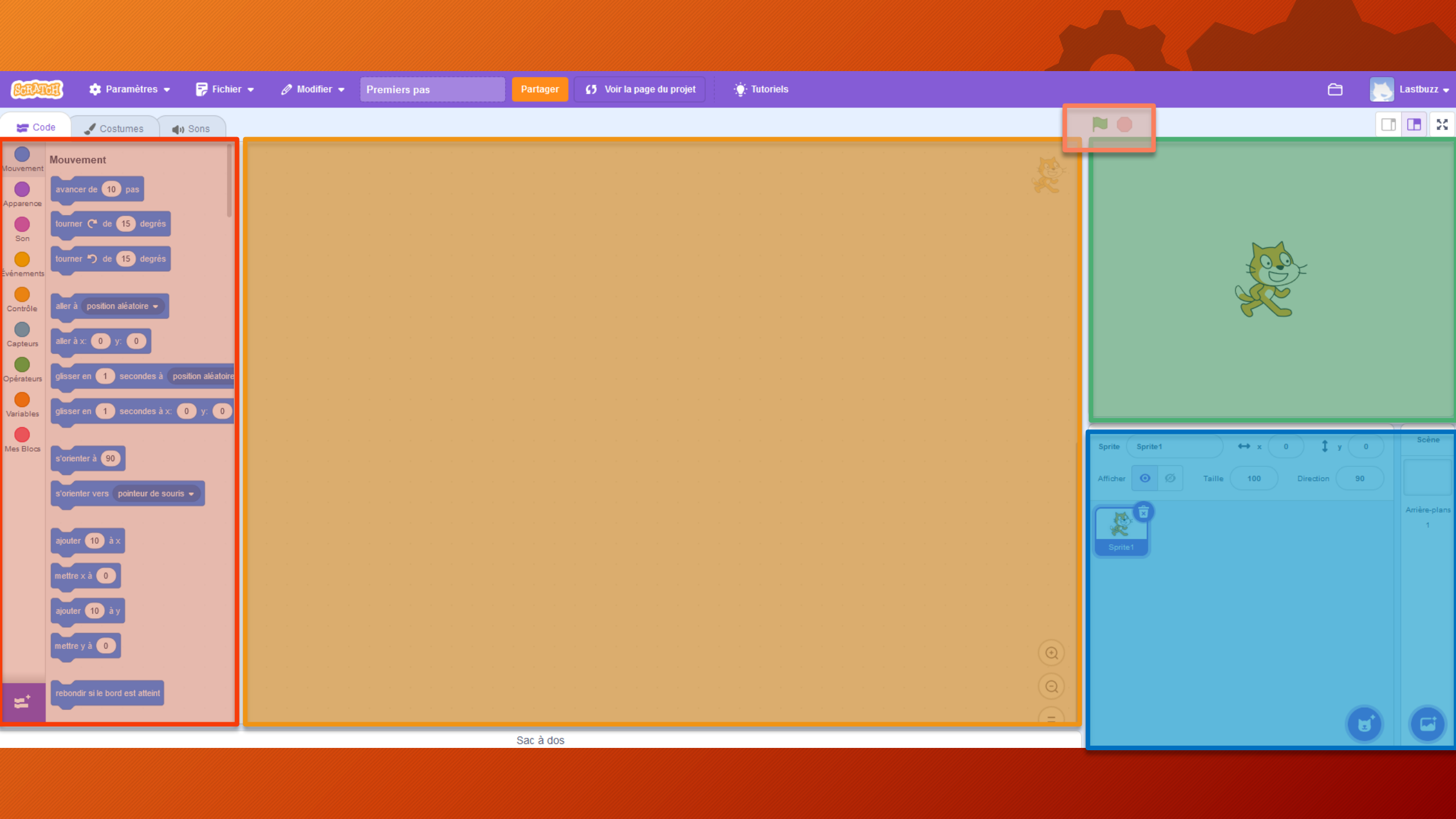
# Aller sur Scratch

https://scratch.mit.edu/

Puis créez-vous un compte

# Votre premier écran de projet Scratch




Paramètres
Fichier
Modifier
Premiers pas
Partager
Voir la page du projet
Tutoriels
Lastbuzz
Code
Costumes
Sons
Mouvement
Apparence
Son
Événements
Contrôle
Capteurs
Opérateurs
Variables
Mes Blocs
Mouvement
avancer de 10 pas
tourner de 15 degrés
tourner de 15 degrés
aller à position aléatoire
aller à x: 0 y: 0
glisser en 1 secondes à position aléatoire
glisser en 1 secondes à x: 0 y: 0
s'orienter à 90
s'orienter vers pointeur de souris
ajouter 10 à x
mettre x à 0
ajouter 10 à y
mettre y à 0
rebondir si le bord est atteint
Sprite
Sprite1
x 0
y 0
Afficher
Taille 100
Direction 90
Scène
Arrière-plans 1
Sprite1
Sac à dos

# La zone rouge : les blocs d'instructions


Ils sont classés en catégories : mouvements, apparence, événements, contrôle…

Pour chaque catégorie, ils sont présentés dans la colonne de droite.

Chaque bloc est écrit « en langue humaine », pas dans un langage de programmation dont il faudrait maîtriser la syntaxe et les conventions. Il correspond à une instruction.

Certains paramètres des blocs d'instruction sont modifiables

Le principe de fonctionnement est : on glisse et on dépose les blocs qu'on souhaite utiliser dans la console de programmation (zone orange), puis on les emboîte, comme un puzzle, pour créer une suite d'instructions = un code

# La zone orange : la console de programmation


C’est la grande zone vide (au début) située au ventre de votre écran.

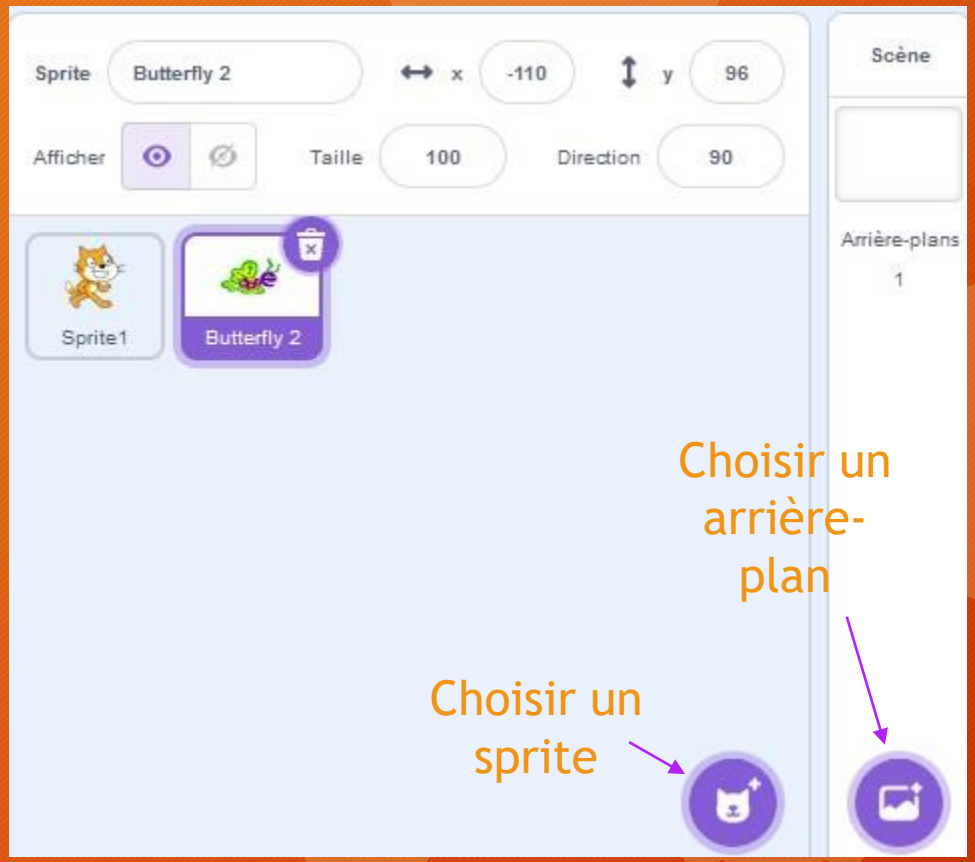
C’est dans cette zone qu’on va faire glisser et emboîter les blocs d’instruction, pour créer du code.

Quand vous avez fait glisser un bloc que vous ne vouliez pas dans cette zone, il suffit de le sélectionner (clic dessus), puis « suppr »

Chaque sprite que vous allez utiliser aura sa propre console de programmation pour donner des instructions. Vous basculerez de la console d’un sprite à celle d’un autre en cliquant sur le sprite que vous désirez dans la zone bleue.

Une petite icône en transparence en haut à droite de la console vous indique à quel sprite elle est rattachée.

# La zone bleue : la gestion des sprites et des arrière-plans




Par défaut, il n'y a qu'un sprite de Scratch dans votre projet.

C'est ici que vous allez pouvoir ajouter des sprites et des arrière-plans à votre projet.

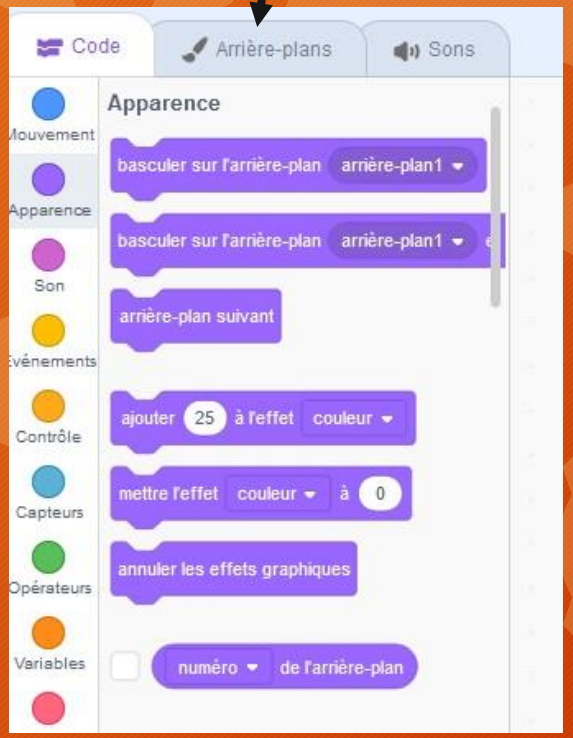
C'est ici que vous allez pouvoir ajuster leur taille, leur orientation initiale, les rendre invisibles (temporairement) le temps que vous travaillez sur un autre sprite.

Quand vous supprimez un sprite (la poubelle), tout le code saisi pour lui disparaît aussi.

En cliquant sur un sprite, vous accédez à sa console de programmation

# La zone bleue : la gestion des sprites et des arrière-plans

C’est ici

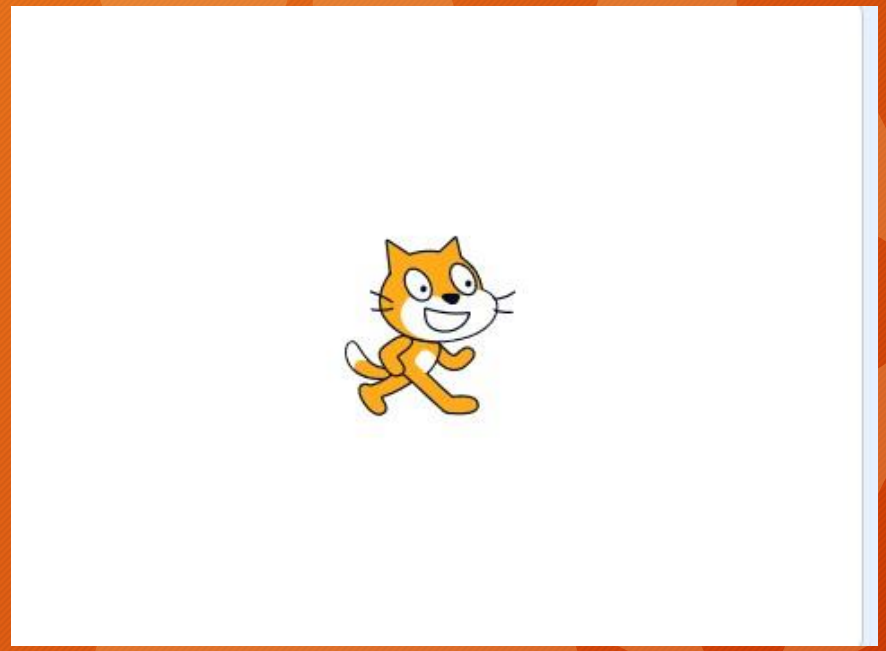



Nota : il y a toujours un arrière-plan blanc par défaut.

Quand vous ajoutez un arrière-plan, il remplace celui présent par défaut, lequel reste néanmoins stocké dans votre répertoire d’arrière-plans.

Si vous souhaitez supprimer définitivement un arrière-plan de votre projet, il faut cliquer sur la colonne « scène » dans la zone bleue, puis dans l’onglet « arrière-plan » (au-dessus de la zone rouge)

# La zone verte : la visualisation de votre projet et de l'exécution de votre code

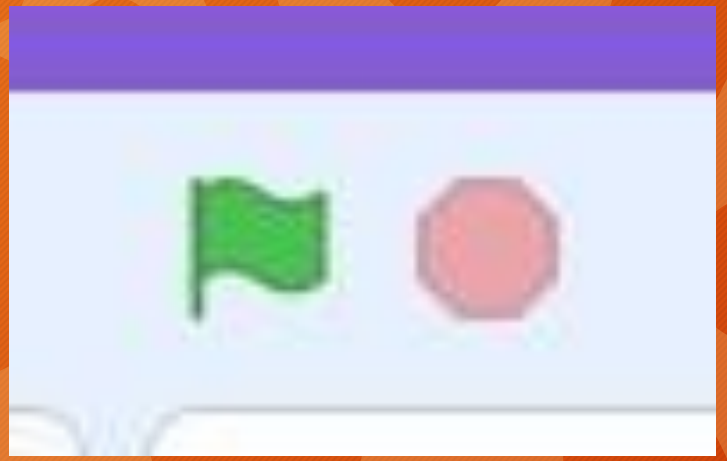


Dans cette zone apparaissent tous vos sprites (si vous ne les invisibilisez pas) et l'arrière-plan retenu. Par défaut, il n'y a que le sprite de Scratch et un arrière-plan blanc.

Quand vous exécuterez votre code, c'est ici que l'ensemble des instructions saisies dans la console de programmation vont se manifester… Si vous les avez bien saisies ☺

Notez qu'il est possible de manipuler les sprites dans cette zone (glisser/déposer), indépendamment de toute instruction, si vous souhaitez agencer votre fenêtre pour plus de clarté.

# La zone rose : exécuter le code/arrêter l'exécution du code



Ce sont deux boutons qui servent à lancer l'exécution du code (drapeau vert) et à l'interrompre (bouton rouge).

Attention : l'exécution ne va se faire que si l'instruction de lancement du code avec le drapeau vert a été spécifiée dans la console de programmation

Si votre code prévoit des itérations illimitées d'une instruction (boucle), le bouton rouge est le seul moyen d'arrêter le programme.

# Principe de manipulation de Scratch

Je sélectionne des blocs d'instructions dans la galerie de blocs

Je glisse et dépose ces blocs dans ma console de programmation

J'assemble mes blocs d'instructions pour réaliser des suites d'instructions

Dans une suite d'instructions, Scratch va toujours lire les éléments du haut vers le bas

J'exécute mes instructions (drapeau vert)

Je vérifie sur la scène ce qui fonctionne ou pas dans l'exécution de mon code

Et je répète cette suite d'actions pour tous les sprites (et éventuellement les arrière-plans) qui composent mon projet