# Création du projet global – Scratch-like V5

## 1. Vision du projet

Créer une application inspirée de **Scratch**, permettant aux utilisateurs d'assembler des blocs visuels pour programmer de manière intuitive et voir le résultat en temps réel.

Objectifs principaux : - Apprentissage ludique de la programmation - Interface claire, visuelle et fluide avec blocs designés en images - Architecture modulaire et évolutive

---

## 2. Fonctionnalités clés

### 2.1 Interface utilisateur

Découpée en **3 zones principales** :

1. **Components (Blocs)**
2. Liste de blocs disponibles avec **arrière-plan PNG et icônes**
3. Catégories (Mouvement, Logique, Événements, Variables…)
4. Drag & Drop HTML5
5. **Assemble (Zone d'assemblage)**
6. Zone centrale scrollable
7. Assemblage visuel des blocs
8. Mode focus (expand) pour travailler sans preview
9. **Preview (Résultat)**
10. Visualisation en temps réel
11. Exécution du programme
12. Boutons : Play / Stop / Reset

---

## 3. Architecture technique complète (V5 – Blocs designés avec images)

### 3.1 Structure des fichiers

```
src/
│
├── index.html            # Structure globale 3 zones + bouton focus
│
├── style/
```

```
│   ├── main.css          # Layout global + thèmes + focus mode
│   ├── blocks.css        # Style blocs avec background-image + icônes
│   └── assemble.css      # Scroll, padding, layout Assemble
│
├── assets/
│   ├── blocks/           # Images PNG des formes de blocs
│   └── icons/            # Images icônes des blocs
│
├── core/                 # LOGIQUE PURE (aucun DOM)
│   ├── Block.js          # Modèle Block (type, params, next, children)
│   ├── Program.js        # Programme (liste racines, validation, add/remove)
│   └── Executor.js       # Interpréteur / exécution des blocs
│
├── data/
│   └── blocksCatalog.js  # Définition des blocs avec référence images
│
├── ui/                   # INTERFACE & DOM
│   ├── components/
│   │   └── BlockView.js  # Génération DOM d'un bloc + drag + background
image
│   │
│   ├── panels/           # ⚠️ Contient les panels UI (CategoriesPanel +
BlocksPanel)
│   │   ├── CategoriesPanel.js  # Affiche les catégories et filtre les blocs
│   │   └── BlocksPanel.js      # Affiche les blocs avec images et gère le
drag
│   │
│   ├── assemble/         # ⚠️ Contient AssembleArea et DropManager
│   │   ├── AssembleArea.js     # Zone scrollable où l'utilisateur assemble
les blocs
│   │   └── DropManager.js      # Gère où les blocs sont déposés et met à
jour le core
│   │
│   └── preview/
│       └── Preview.js
│
├── utils/
│   ├── uid.js            # Génération d'ID uniques
│   └── dom.js            # Helpers DOM simples
│
└── main.js               # Point d'entrée
```

**Commentaire sur les panels** :

> Les panels (`CategoriesPanel.js` et `BlocksPanel.js`) sont des composants purement UI. Ils n'ont **aucune logique métier**, et servent uniquement à afficher et filtrer les blocs disponibles. CategoriesPanel filtre les blocs selon la catégorie sélectionnée, tandis que BlocksPanel affiche les blocs (avec images et icônes) et les rend drag & drop pour l'utilisateur.

**Commentaire sur Assemble** :

> `AssembleArea.js` est la zone principale où l'utilisateur construit son programme. Elle est scrollable et peut passer en mode focus. `DropManager.js` s'occupe de détecter où les blocs sont lâchés et met à jour le modèle Program dans `core/`. L'assemble reflète toujours l'état du programme mais ne contient **aucune logique métier**, seulement la représentation DOM des blocs.

### 3.2 Flux de données

```
User action
   ↓
UI (drag, click, visual block images)
   ↓
Program / Block (core)
   ↓
Update state
   ↓
UI redraw (BlockView met à jour DOM avec background PNG et icônes)
```

### 3.3 Fonctionnement de main.js

- Importe tous les modules (core, ui, data)
- Initialise `Program` pour contenir tous les blocs
- Crée et affiche le catalogue de blocs avec images : `BlocksPanel.init(blocksCatalog)`
- Initialise AssembleArea : `AssembleArea.init(program)`
- Initialise Preview : `Preview.init(program)`
- Ajoute écouteurs pour :
- Drag & Drop HTML5 avec images
- Toggle focus mode
- Play / Stop / Reset
- Maintient le flux **UI ↔ core ↔ UI**

### 3.4 BlocksCatalog.js exemple avec images

```
export const blocksCatalog = [
  {
    id: 'move_10',
    type: 'action',
    label: 'Avancer',
    background: 'assets/blocks/move.png',
    icon: 'assets/icons/arrow.png',
    params: { steps: 10 }
  },
  {
    id: 'repeat',
    type: 'loop',
    label: 'Répéter',
    background: 'assets/blocks/loop.png',
```

```
    icon: 'assets/icons/repeat.png',
    params: { times: 5 }
  }
];
```

### 3.5 BlockView.js adaptation

- Crée un élément DOM `<div>` pour le bloc
- Ajoute `background-image` pour la forme du bloc
- Ajoute `<img>` pour l'icône
- Configure `draggable=true`
- Écoute dragstart, dragend

### 3.6 CSS Blocks.css

- Définit taille fixe ou responsive pour chaque bloc
- `background-size: cover;` ou `contain`
- Positionnement icône avec `position: absolute`
- Padding interne pour label et paramètres

### 3.7 main.css reste global

- Layout global (flex pour 3 zones)
- Scroll, focus mode
- Couleurs et thèmes
- Les styles des blocs sont séparés dans `blocks.css`

---

## 4. Roadmap ajustée pour images

- Semaine 1 : UI + scroll + focus mode + intégrer PNG/Icons dans panels et Assemble
- Semaine 2 : Drag & Drop HTML5 avec visuels blocs
- Semaine 3 : Assemblage logique + validation + blocs imbriqués
- Semaine 4 : Executor + Preview + Play/Stop + polish UX visuel

> Cette V5 ajoute maintenant un **commentaire clair sur Assemble**, détaillant la fonction d'AssembleArea et DropManager pour refléter l'état du programme sans inclure de logique métier.