Premier jeu vidéo avec Scratch

Par Stéphane Ngov - Vincent Viale

Date de publication : 27 juin 2016

DÉBUTANT

Dans ce tutoriel, nous allons voir comment construire un jeu vidéo minimaliste. Il vous permettra d'apprendre les principes de base et, par la suite, le jeu pourra être modifié, étendu, amélioré…

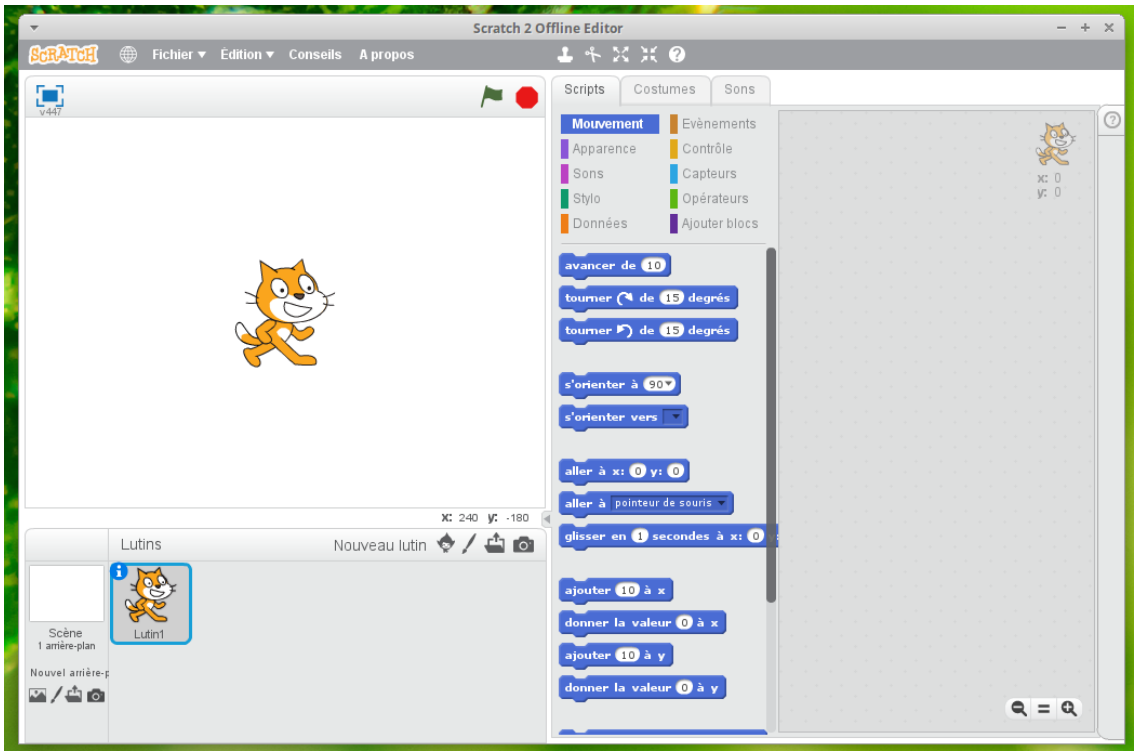
Vous pouvez commenter l'article en suivant le lien suivant : **Commentez**, alors après votre lecture, n'hésitez pas.

I - Introduction

I-A - Scratch : en quelques mots !

Scratch un nouveau langage de programmation visuel et coloré en français pour apprendre, dès l'âge de huit ans, les concepts fondamentaux en mathématiques et en informatique. Il permettra à vos enfants de créer facilement des histoires interactives, des dessins animés, des jeux, des compositions musicales, des simulations numériques, etc. Cela leur apprendra aussi à développer des idées créatives, à avoir un raisonnement cohérent et également à travailler en équipe.

Scratch permet de manipuler des éléments graphiques simples et colorés, pour les imbriquer les uns aux autres de la même façon que lorsque vous construisez un puzzle. Finalement, vous obtenez une série de fonctions, et il vous est alors possible de voir le résultat sur les objets que vous avez insérés. Et il ne nécessite aucune connaissance dans un langage de programmation juste « *un peu d'imagination* ».



Vous trouverez **ici** des explications sur l'interface de Scratch.

Dans ce tutoriel, nous allons créer un jeu de labyrinthe. L'objectif est d'être pédagogique et destiné à des enfants, nous nous concentrerons donc sur la méthode pour réaliser le jeu, plutôt que sur le côté artistique.

I-B - Les différents éléments

Les ingrédients, pour cet exemple, sont :

- un fond d'écran ;
- une boule verte qui correspond au joueur ;
- une flèche pour déterminer la sortie.

Voici les hypothèques que nous allons prendre :

- si je touche la couleur noire qui correspondra au mur, je retourne au point de départ, pour nous cela sera le centre du jeu ;
- si je touche la sortie, j'ai gagné.

II - Préparation du jeu

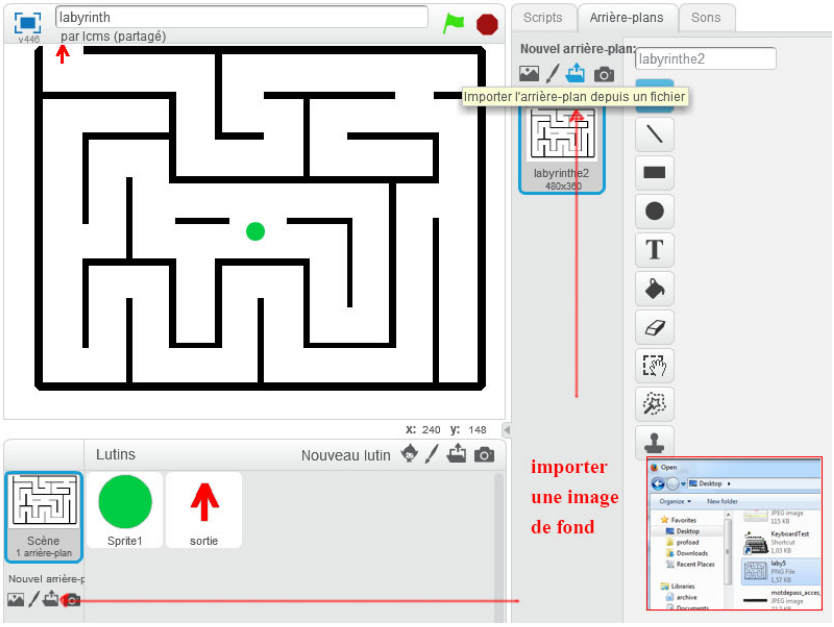
II-A - L'arrière-plan

L'arrière-plan (ou fond d'écran) permet dans un premier temps d'ajouter une forme de présentation, une forme de décor ou de mise en scène.

Mais il a aussi une autre fonctionnalité très intéressante, qui permet d'interagir avec, comme de définir des contraintes, des obstacles, etc.

Dans notre cas, nous nous en servirons comme d'un obstacle qui ne pourra pas être franchi. Avant de commencer à déposer les Scratchs, nous allons télécharger un arrière-plan, ce sera notre **labyrinthe**.

Pour insérer une image en arrière-plan, nous avons le choix entre deux icônes :



Cela ouvrira une fenêtre nous permettant d'aller chercher l'image souhaitée.

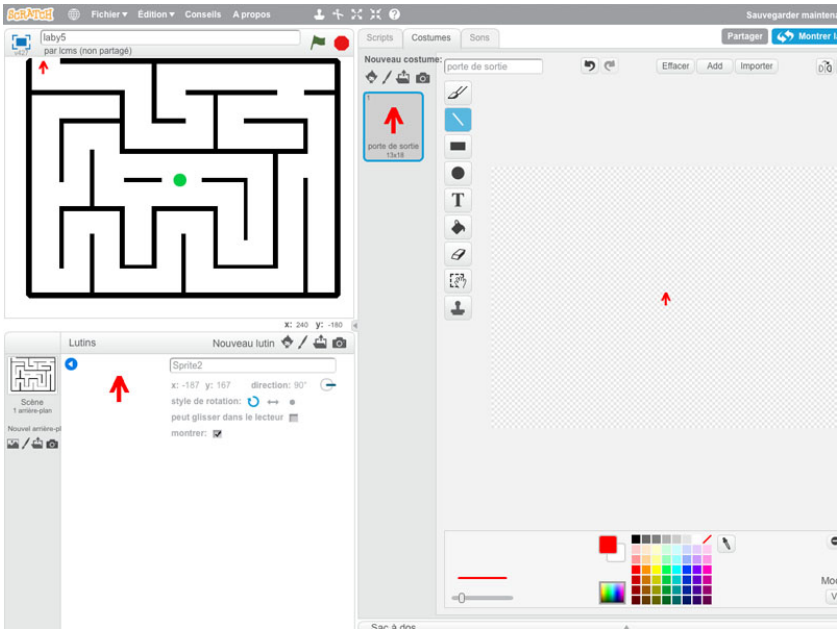
Nous pouvons bien sûr en créer ou en télécharger une autre, mais l'image devra obligatoirement avoir une taille de 480 x 360 pixels.

II-B - Les personnages

Le personnage, sous Scratch, s'appelle un **lutin**. Le programme va lui donner vie pour qu'il sache évoluer, car une image seule ne sait pas se déplacer et interagir avec le reste.

Maintenant que l'arrière-plan est fait, nous allons pouvoir créer un lutin. Dans notre cas, nous allons créer une petite boule verte qui nous servira à nous déplacer dans ce jeu, le but étant de sortir de ce labyrinthe.

Pour détecter la sortie, il nous faut un autre élément, ce sera une flèche rouge, mais qui peut être de n'importe quelle couleur, à vous de choisir :



Maintenant que nous avons terminé avec les éléments de notre jeu, nous allons commencer par travailler sur la boule, qui est le joueur.

II-B-1 - Déplacer le personnage

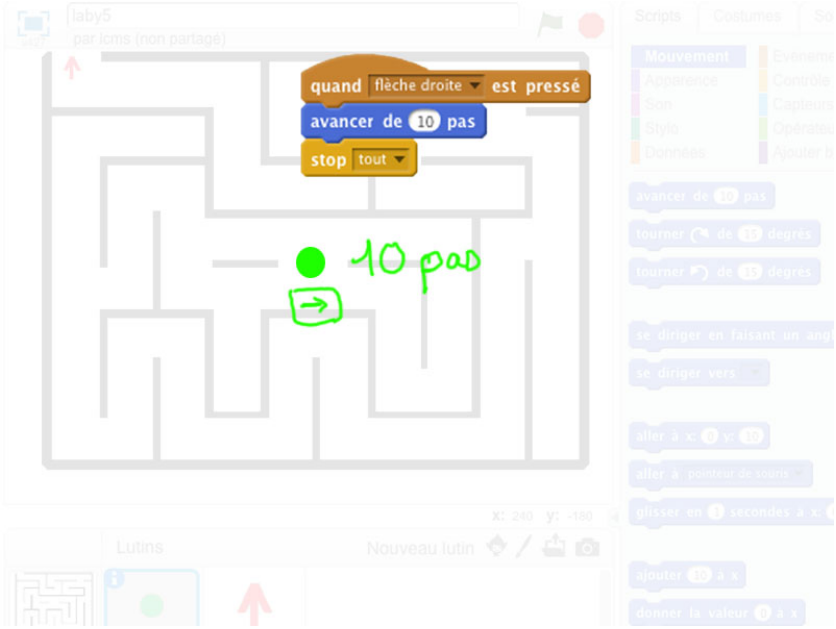
Pour que notre jeu soit interactif, il nous faut des commandes pour le faire évoluer dans le labyrinthe.

Pour déplacer le joueur, nous allons utiliser une série d'événements, dans notre cas, cela sera :

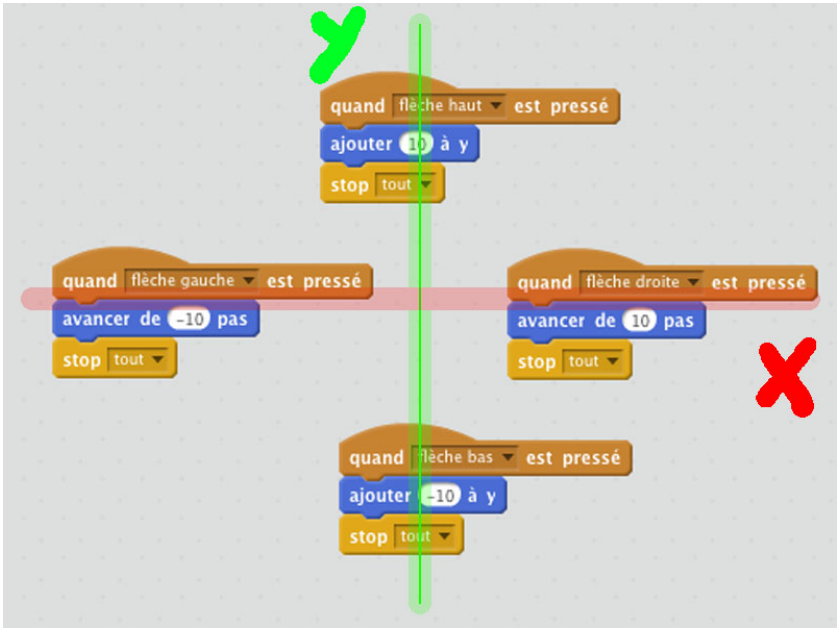
Il existe de multiples événements qui sont classés par catégorie, chaque catégorie correspondant à une couleur :

Chaque événement s'appelle aussi un Scratch.

Une fois que nous avons déposé les trois Scratchs, nous pouvons voir le résultat, à chaque fois que nous appuyons sur la flèche de droite la boule avance de dix pas :



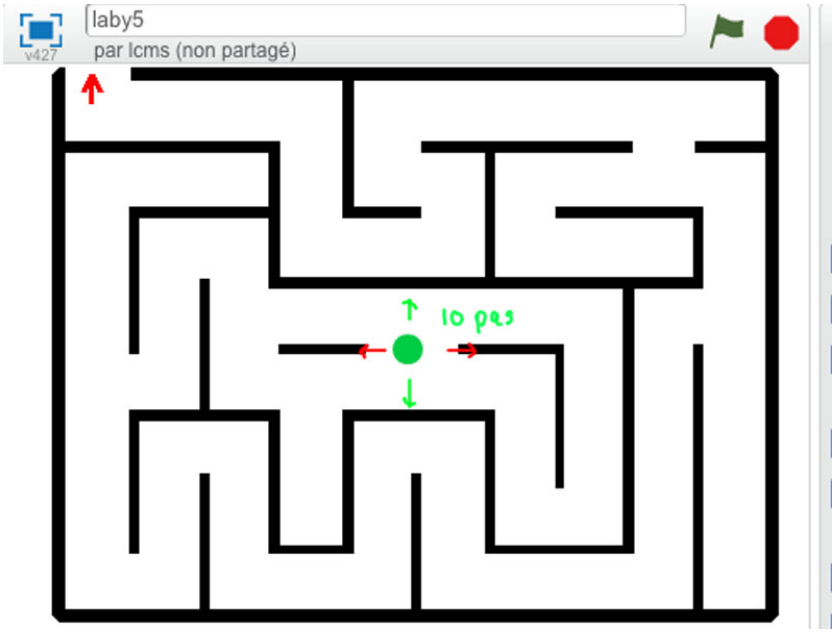
Notre boule ne sait aller que dans une direction, pour qu'elle puisse se promener dans le labyrinthe, il nous faut ajouter les trois autres événements : gauche, haute et bas :



Il vous suffit de cliquer sur la petite flèche noire pour voir apparaître les différentes possibilités :

Comme il n'est pas possible de reculer, il faut mettre une valeur négative au nombre de pas pour avoir une représentation de recul.

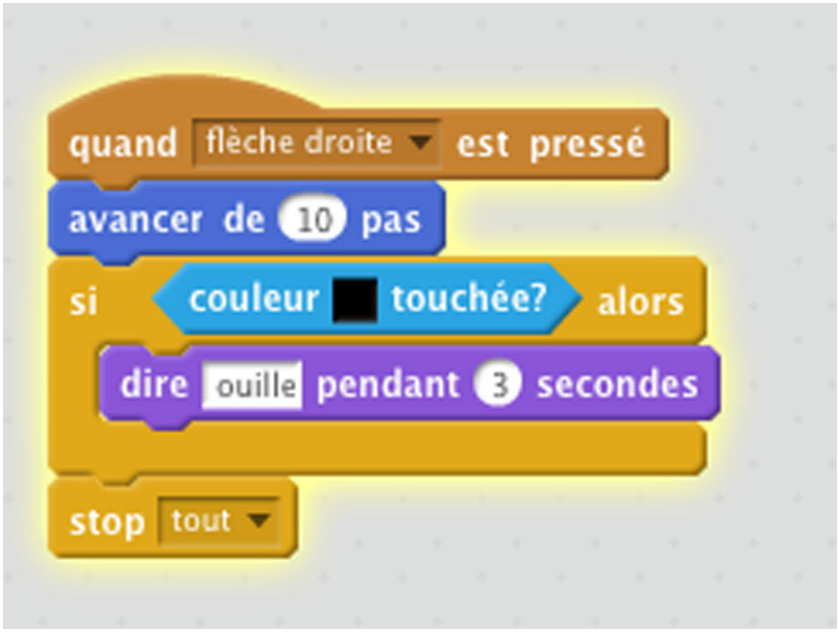
La boule peut maintenant être déplacée de tous les côtés :



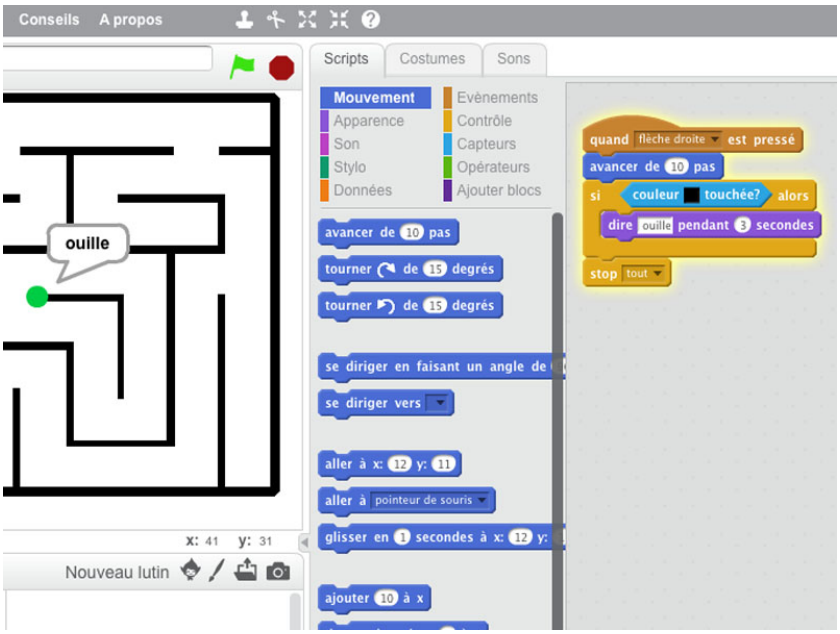
II-C - Les obstacles

II-C-1 - Les bords du labyrinthe

Pour le moment, nous pouvons nous déplacer librement, pour que la boule détecte un mur, il faut deux Scratchs, le capteur et un contrôle « si » :



Si j'avance de dix pas et que je rencontre la couleur noire qui correspond au mur, je dis à la boule de dire « ouille » :

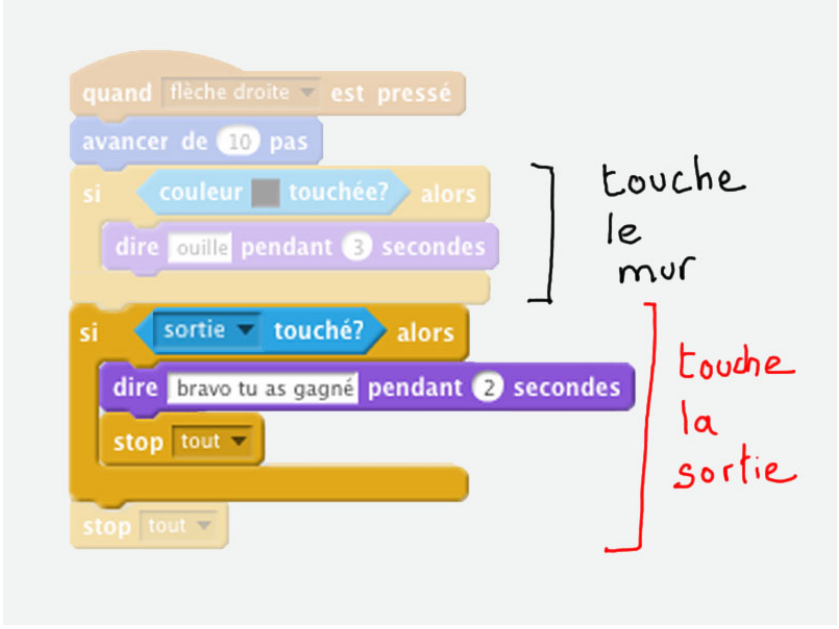


II-C-2 - La sortie

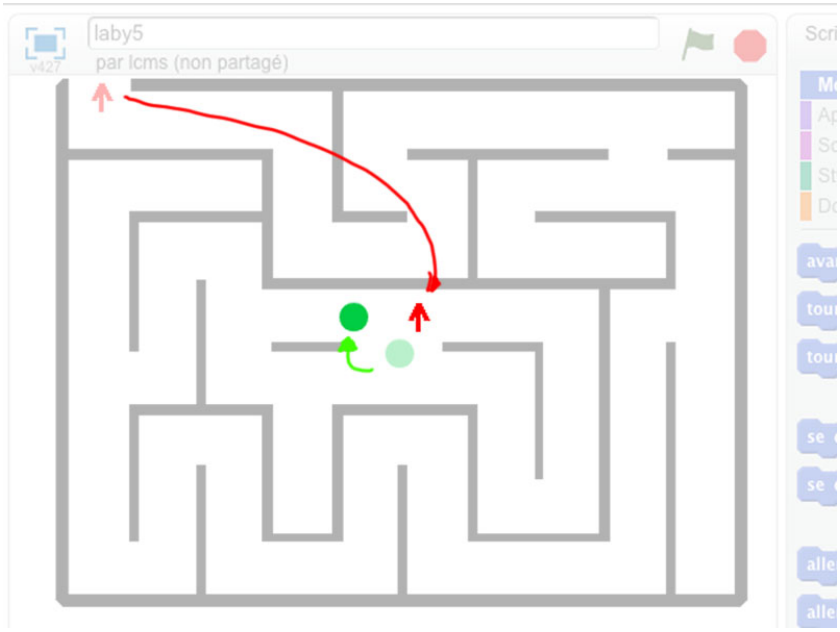
Maintenant que nous savons détecter un mur, il faut se poser la question si la boule est sortie.

Cette fois-ci on n'utilisera pas le Scratch de détection d'une couleur, mais un élément qui a la forme d'une flèche rouge (que nous avons définie précédemment comme étant un lutin), mais nous aurions pu très bien dessiner une porte, une coupe, etc.

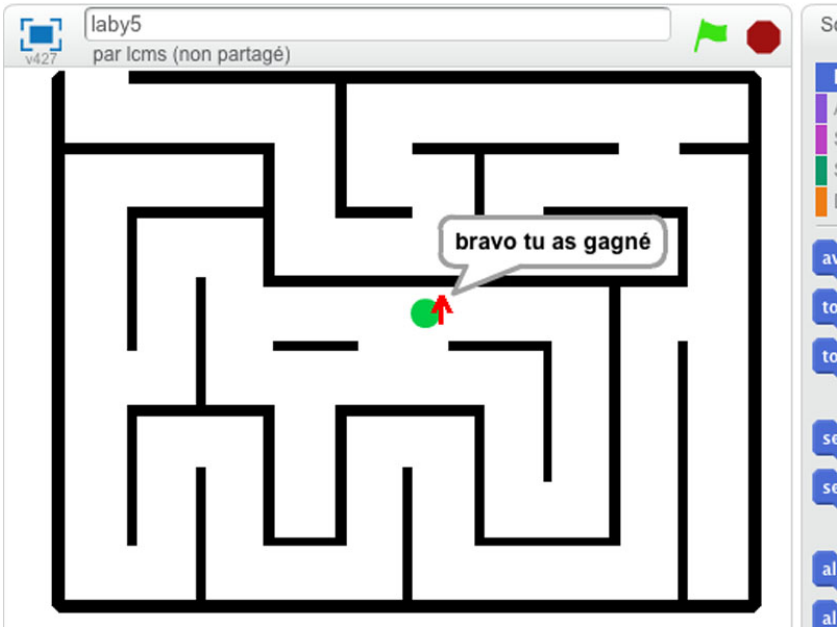
Voici le bout de programme :



Pour tester si ce bout de programme marche, il suffit de regrouper la boule verte et la flèche rouge :



Puis, de se déplacer jusqu'à la flèche rouge :



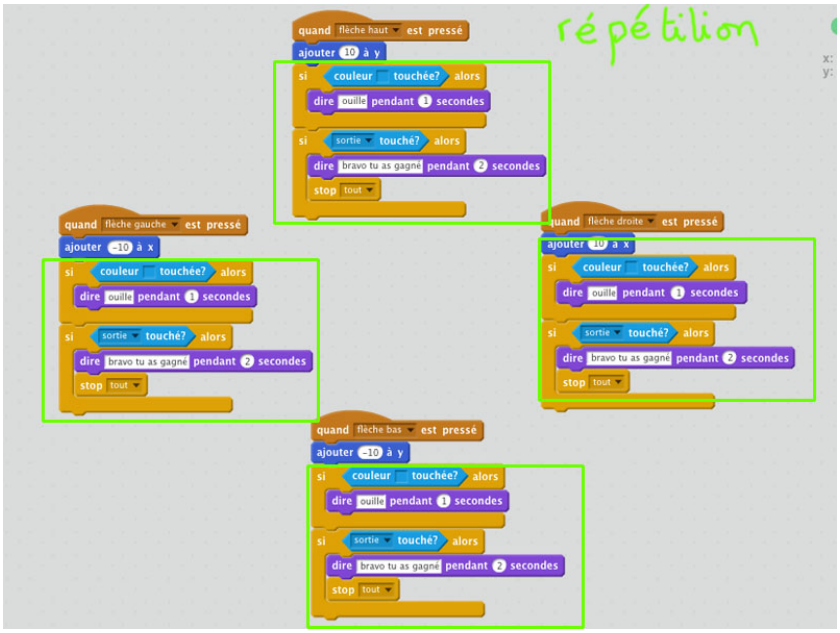
Voilà, notre programme sait maintenant :

- détecter un mur ;
- avancer à droite ;
- détecter la sortie.

II-C-3 - Les autres directions

Pour que le jeu marche bien, il nous faut avoir le même programme de détection pour le déplacement vers le haut, le bas et la gauche.

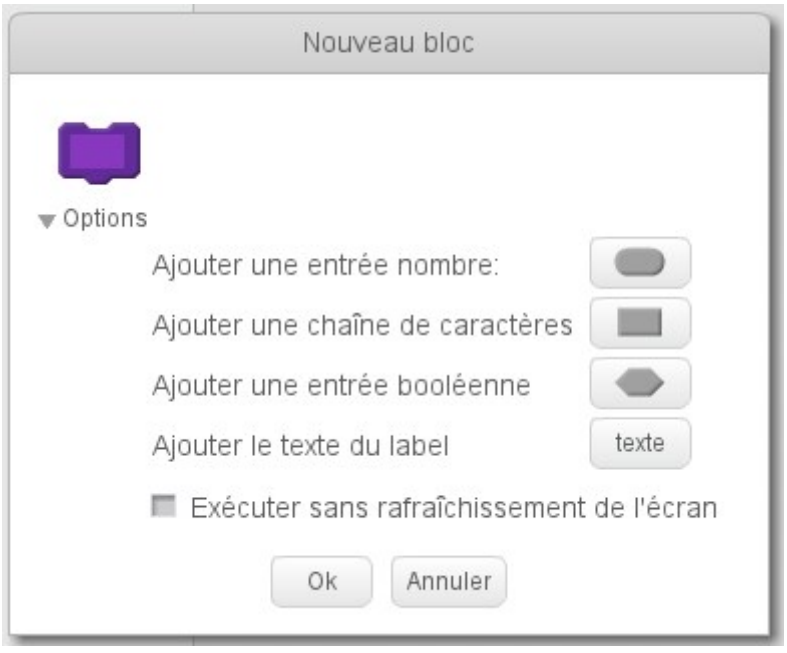
Faisons la même chose pour les trois autres directions, et voici ce que cela nous donne :



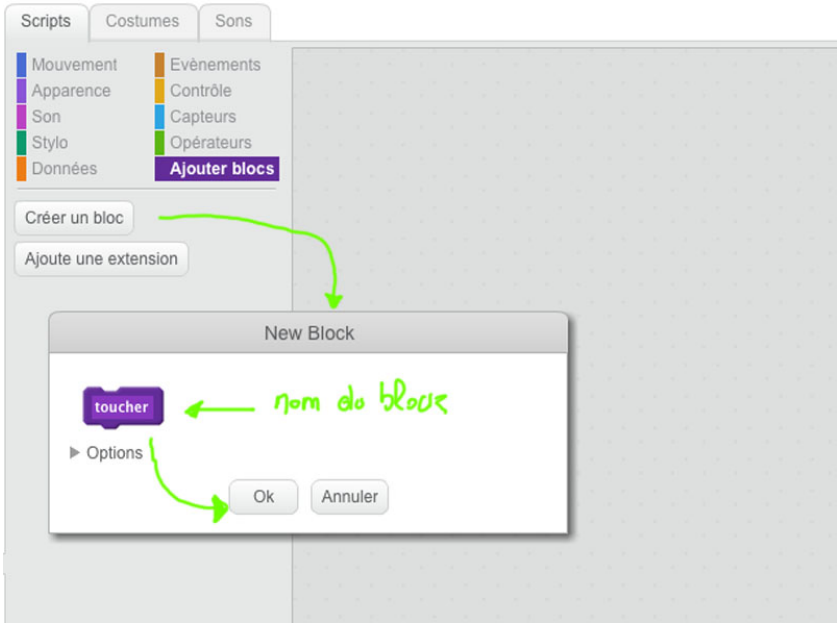
II-D - Amélioration du code

Nous pouvons remarquer que les scratchs encadrés en vert (dans l'image précédente) sont identiques. Avec Scratch nous allons pouvoir utiliser une fonctionnalité pour éviter ces répétitions, elle s'appelle **bloc**.

> *Un **bloc** doit être construit quand vous réutilisez les mêmes séquences dans différentes parties de votre programme.*
>
> *Il peut être constitué de séquences simples (mouvements, boucles, etc.) ou de séquences plus compliquées avec l'utilisation des variables.*
>
> *Et vous verrez que les possibilités sont nombreuses :*

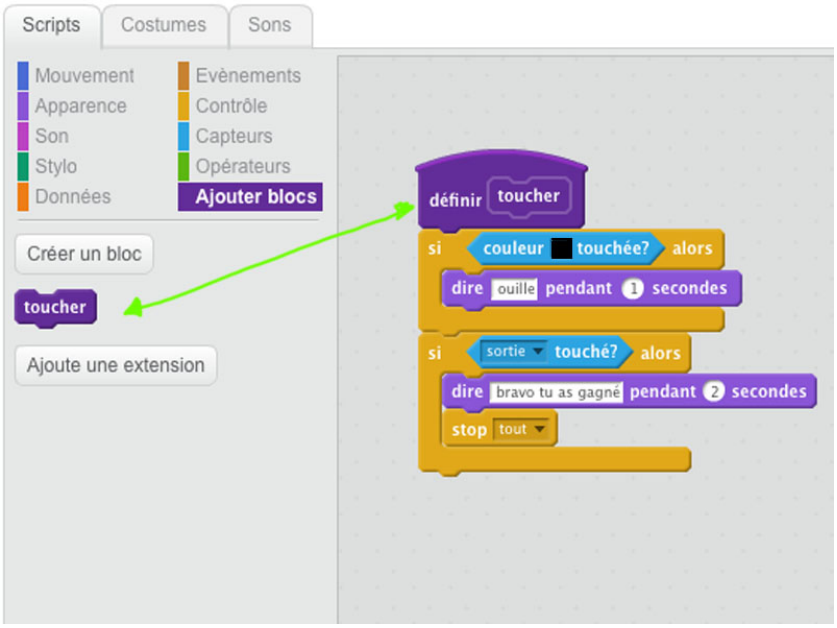


Ajoutons un bloc que nous appellerons « toucher » :

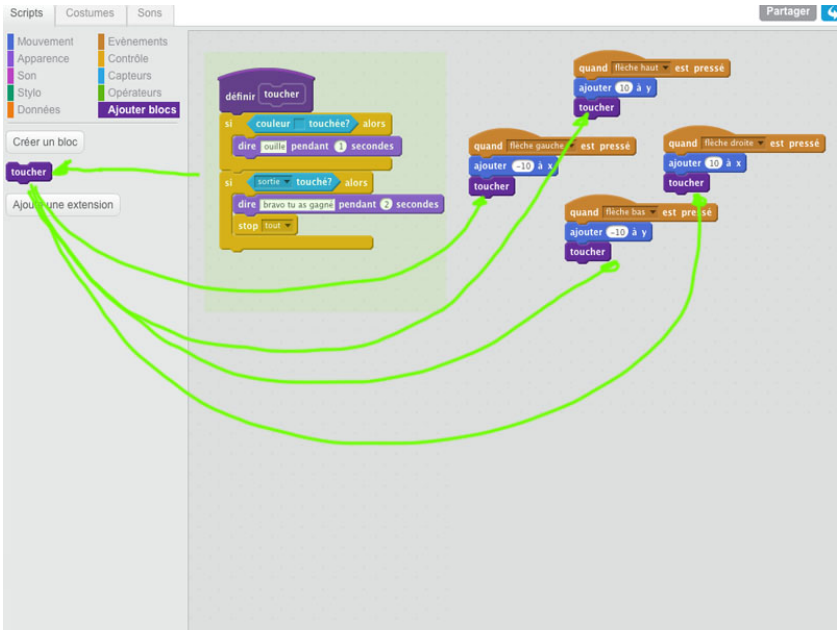


Une fois que le bloc est prêt, un Scratch « toucher » est créé dans le menu « Block », et sur la surface de travail, il y a un « **gros** » Scratch dont le chapeau est en forme de courbe.

Il suffit maintenant d'accrocher le programme que nous avons identifié comme répétitif :



Une fois que nous avons notre **block**, il ne reste plus qu'à supprimer l'ancien programme par le bloc « toucher » sur les quatre événements :



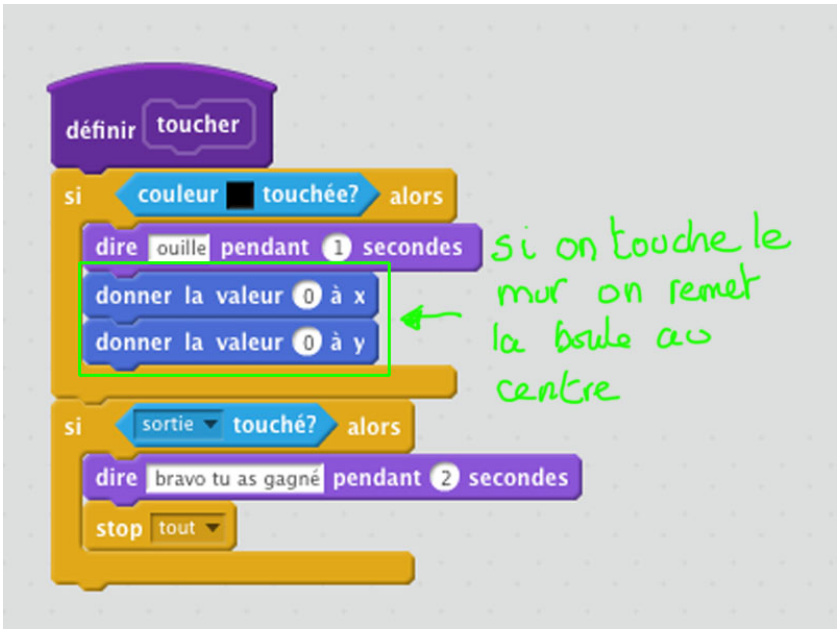
Avec cette méthode, nous réduisons la longueur du programme et si nous souhaitons modifier un élément, il ne sera pas utile de le répercuter sur chaque déplacement, ce qui peut éviter des erreurs par la suite.

Par exemple, « tu as gagné » par « gagné », nous aurons juste à changer un seul message et pas à intervenir sur les quatre mouvements.

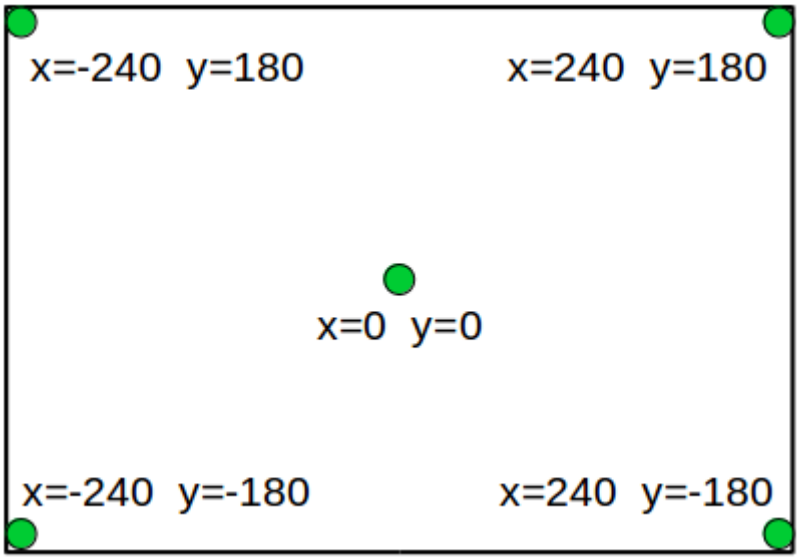
II-E - Pour aller plus loin

Nous pouvons « punir » le joueur d'avoir touché le mur, nous allons donner l'ordre de revenir au centre du jeu, il devra donc recommencer.

Prenons le Scratch du mouvement (le bloc créé), il nous suffira donc de mettre les valeurs X et Y à 0 dans la suite de la condition, si nous touchons le mur noir :



La position X = 0 et Y = 0 correspond au centre de l'image, voici une représentation de l'écran de Scratch :

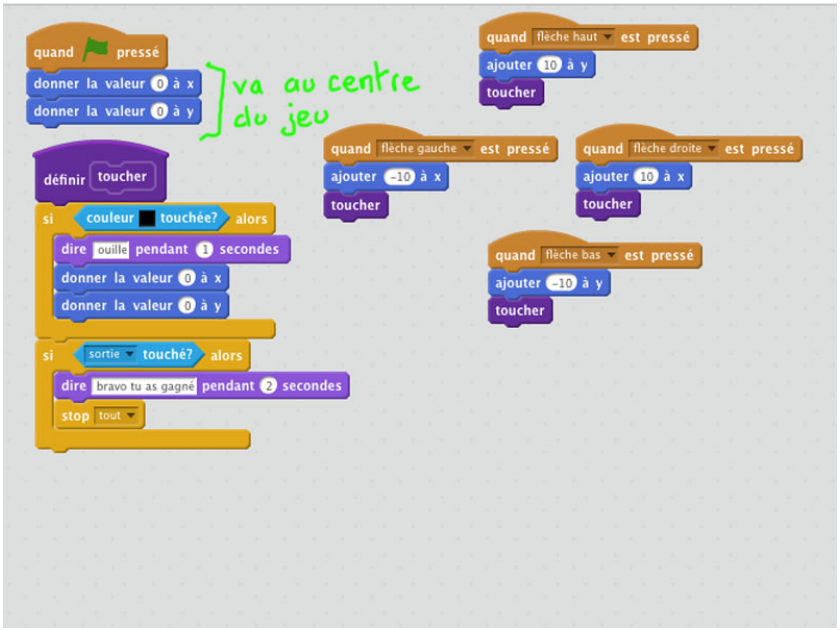


II-F - Finaliser le programme

Nos quatre événements fonctionnent, mais notre programme n'est pas tout à fait terminé.

Un programme doit toujours avoir une entrée, cela se matérialise par le drapeau vert.

Lorsque nous demandons de jouer, nous allons faire en sorte que le départ de la boule soit au centre du jeu, nous allons donc initialiser le jeu en mettant les valeurs de X et Y à 0 :



III - Améliorer le jeu

Voici le Source **fichier Scratch** du jeu que nous venons de créer.

Le jeu de base du labyrinthe est terminé. Mais nous l'avons fait de façon « *simpliste* », ce qui nous a permis de voir une partie des possibilités qu'offrent Scratch.

Maintenant à vous de l'améliorer :

- en rajoutant des obstacles comme des lutins méchants ou des cloisons qui apparaissent ou qui peuvent se déplacer ;
- en mettant un personnage à la place de la flèche ;
- en définissant des points de vie au personnage ;
- en définissant un positionnement aléatoire du personnage au départ ;
- en rajoutant un chronomètre ;
- etc.

Vous constatez que les améliorations sont diverses, elles dépendront de l'imagination de votre enfant.

IV - Pour partir sur de « bonnes bases »

Dans ce tutoriel, nous avons vu comment réaliser un programme de jeu simple. Nous n'y avons pas mis beaucoup d'éléments comme des personnages supplémentaires, différents décors, des actions, etc.

Si vous souhaitez en rajouter d'autres, il vous faudra établir une chronologie des événements. Ensuite, il vous faudra adopter une certaine méthodologie pour identifier les interactions que peuvent avoir un élément avec les autres. Tout ceci, afin, d'avoir finalement quelque chose qui sera cohérent et évolutif.

V - Remerciements

Nous remercions **LittleWhite** pour ses remarques, **jacques_jean** pour la relecture orthographique.