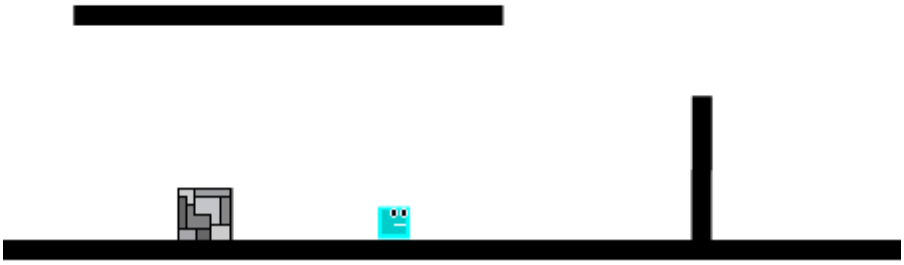
Jeux de plateau avec Scratch

Dans les jeux de plateformes ou plateaux, le joueur contrôle un objet ou un personnage qui doit sauter sur des plates-formes, suspendues ou non dans les airs et éviter des obstacles. Le joueur a un contrôle sur la hauteur et la distance des sauts.

Nous allons voir ici les techniques de base pour réaliser un jeu de plateaux avec Scratch. Nous avons ici 2 cubes qui partent de la gauche de la scène et doivent atteindre la droite de la scène pour passer au niveau suivant. Pour cela ils doivent pouvoir sauter et se déplacer en évitant les murs noirs.



Nous définissons pour chacun de ces cubes des variables :

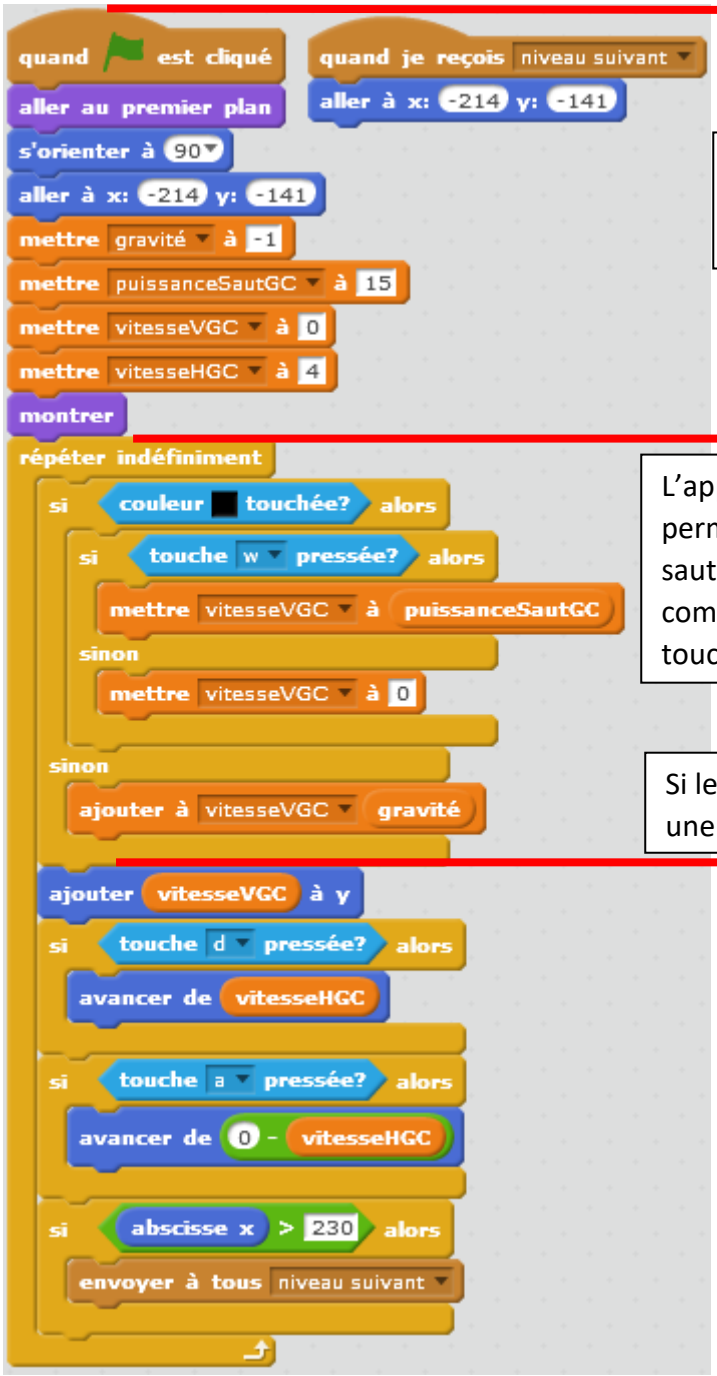
Gravité est utilisée par les deux cubes. Cette variable simule l'attraction terrestre. Si le cube n'est pas sur une plateforme, il est soumis à l'attraction terrestre et tombe.

Les variables puissanceSautGC et puissanceSautPC donne la valeur en pixels du saut à effectuer par le grand cube (GC) ou le petit cube (PC).

Les variables vitesseHGC et vitesseHPC donne la valeur du déplacement à l'horizontal pour le grand cube ou le petit cube.

Les variables vitesseVGC et vitesseVPC donne la valeur du déplacement à la verticale pour le grand cube ou le petit cube. Cette vitesse est à 0 lorsque le cube touche une plate-forme, elle est positive si le cube saute et négative si le cube tombe.

Première technique : Le code du grand cube



Mise en place du cube et initialisation des variables

L'appui sur la touche W qui permet au grand cube de sauter n'est prise en compte que si le cube touche une plateforme.

Si le cube ne touche pas une plate-forme, il tombe.

Dans la boucle de jeu, « Répéter indéfiniment », le grand cube ne peut sauter que s'il touche du noir.

Nous testons donc si le grand cube touche du noir.

- S'il touche du noir **et** que la touche W est pressée, on met la variable vitesseVGC à la valeur contenue dans la variable puissanceSautGC
- S'il touche du noir **mais que la touche W n'est pas pressée**, la variable vitesseVGC est mise à 0 puisque le cube ne doit pas bouger verticalement : il est posé sur une plate-forme.
- S'il ne touche pas du noir alors cela veut dire que **le cube est dans le vide**. On ajoute à la variable vitesseVGC la valeur de la gravité qui ici est de -1 : on diminue donc la valeur de la vitesse verticale.

Après ce test, **quelque soit la position du cube, on ajoute à l'ordonnée y du cube, c'est-à-dire à sa position verticale, la valeur de la variable vitesseVGC.**

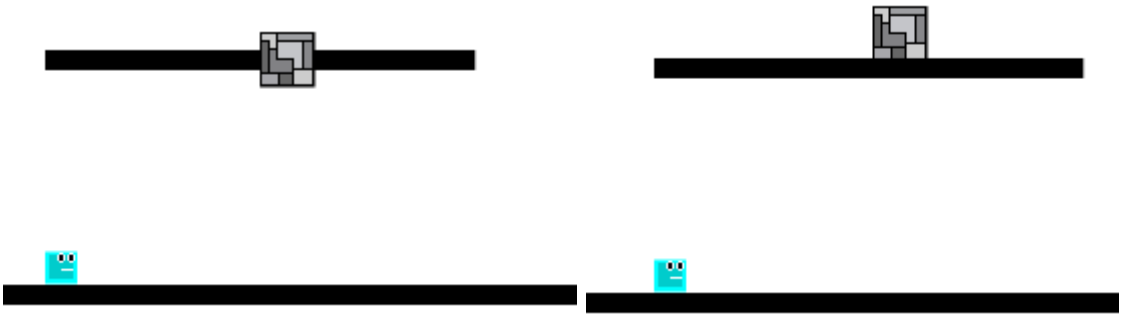
Si cette valeur est nulle, le cube ne change pas de position en vertical, si elle est positive, le cube monte, si elle est négative le cube descend.

Les touches **d** et **a** permettent de déplacer le cube à droite ou à gauche.

Nous testons ensuite si le cube arrive complètement à droite afin de passer au niveau suivant.

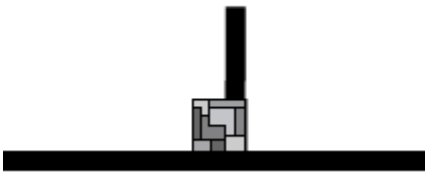
Problèmes :

Si le cube se trouve sous un plateau et qu'il saute, il peut traverser ce plateau pour ensuite se reposer dessus.



En effet, une fois la demande de saut émise, la vitesse du cube à la verticale devient positive et si le plateau au dessus est suffisament près, rien n'empêche le cube de le traverser puisque sa coordonnée « y » est brutalement augmentée.

De la même façon, si un mur vertical barre la route au cube, il le traverse sans être arrêté. Le cube avance ou recule sans que l'on teste s'il touche ou non la couleur noire.



Cette première technique a donc des limites importantes, mais elle permet tout de même de réaliser des jeux simples, dans lesquels on sera attentif à la position des différents plateaux. Le gros travail consiste donc à créer les différents niveaux en s'assurant que le cube ne puisse jamais passer à travers un mur, à déterminer comment on passe d'un niveau à l'autre.

Voir le programme BlocJumper : dans ce programme le cube doit passer à travers l'anneau pour passer au niveau suivant. S'il touche la lave (zone rouge) ou s'il touche un pic, il perd une vie. S'il perd ses 3 vies, il a perdu. Les costumes des lutins sont tirés d'un jeu trouvé sur la plate-forme Scratch. J'en ai remanié certains.

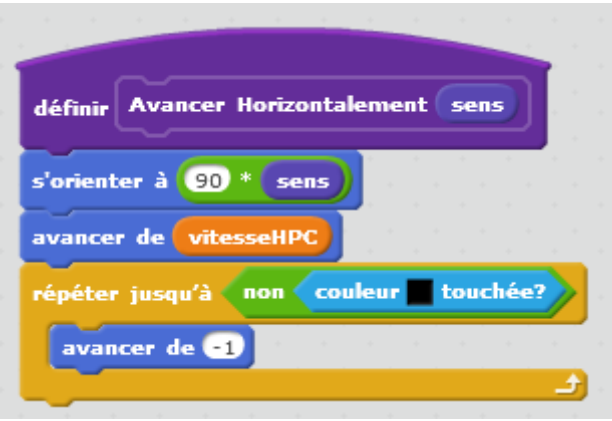
Deuxième technique : Le code du petit cube

Déplacement horizontal

Pour empêcher le cube de traverser les murs verticaux, nous devons tester s'il touche du « noir » lors de son déplacement à l'horizontal.

Si c'est le cas nous le renvoyons en arrière de 1 pixel pour qu'il ne touche plus le mur à gauche ou à droite, suivant le sens de son déplacement.

Nous écrivons un bloc dont le paramètre indiquera le sens du déplacement. Ce paramètre vaudra -1 si le cube se déplace vers la gauche et 1 si le cube se déplace vers la droite.

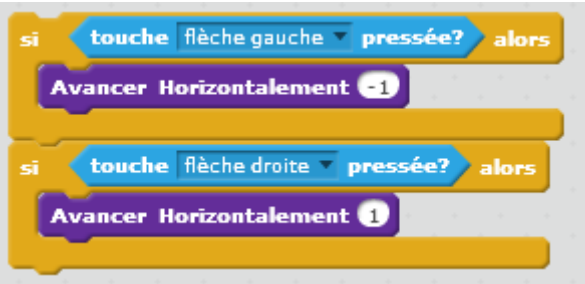


Dans ce bloc, nous orientons le cube à droite ou à gauche suivant la valeur de « sens »

Nous avençons de la valeur du déplacement en x contenue dans la variavle vitesseHPC et nous testons si le cube touche la couleur noire. **Tant que le cube touche le noir**, nous le faisons reculer.

Nous sortons de la boucle, lorsque le cube ne touche plus la couleur noire.

Appel de ce bloc :



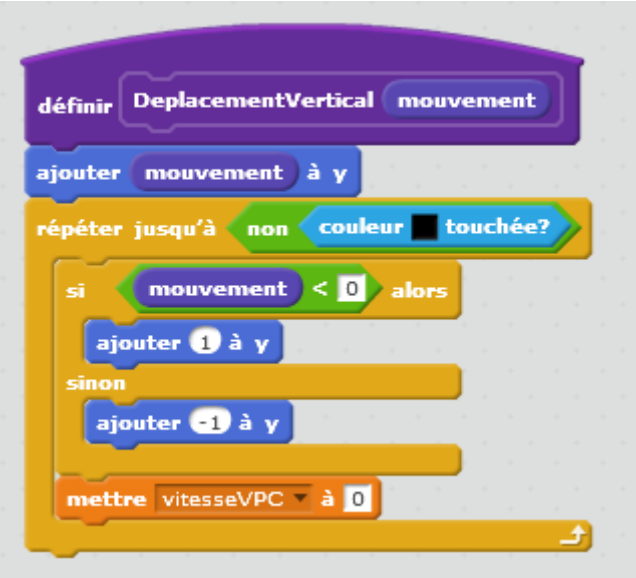
Déplacement vertical

Pour empécher le cube de traverser les murs situés au dessus de lui, nous devons tester s'il touche du « noir » lors de son déplacement vertical.

Si le cube touche du « noir » alors qu'il monte, nous le renvoyons en arrière de 1 pixel pour qu'il ne touche plus le mur au dessus de lui.

Si le cube touche du « noir » alors qu'il est en cours de descente, nous devons là aussi le renvoyer en arrière, car sinon il traversera le mur situé sous lui.

Nous écrivons un bloc dont le paramètre « mouvement » indiquera la valeur du déplacement en y à effectuer au moment où il est appelé.



Dans ce bloc, nous ajoutons à y la valeur contenue dans le paramètre mouvement. Si « mouvement » est positif le cube monte, s'il est négatif le cube descend.

Les instructions de la boucle « répéter » sont exécutées tant que le cube touche du noir.

Tant que le cube touche du noir, on regarde si le mouvement était montant ou descendant. Si le cube était en montée (mouvement positif) et qu'il touche du noir, alors il faut le faire redescendre d'un pixel.

Si le cube était en descente (mouvement négatif) et qu'il touche du noir, alors il faut le faire remonter d'un pixel.

Dans cette boucle nous remettons la vitesse verticale du cube à 0, ce qui annule l'effet d'un saut.

Nous sortons de la boucle, lorsque le cube ne touche plus la couleur noire, ni au dessus ni en dessous.

Le petit cube se déplace donc toujours 1 pixel au dessus d'une ligne noire.

La boucle de jeu du petit cube

Problème :

Lorsqu'on appuie en continu sur la flèche « haut », le cube s'envole.

Il faut donc empêcher que le joueur appuie trop longtemps sur cette touche. En effet s'il appuie en continu sur cette touche, on ajoute la valeur de la puissance du saut plusieurs fois à y avant que le cube ne puisse recommencer à descendre.

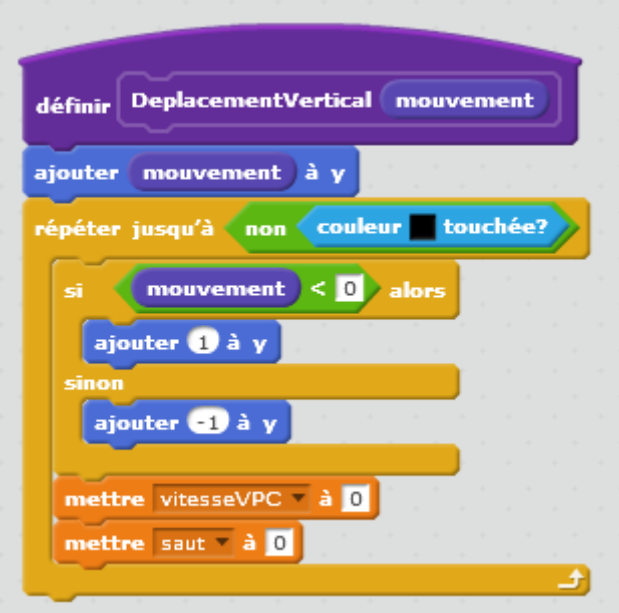
Nous allons définir 2 nouvelles variables : nbSautMax qui indiquera pour tout le jeu combien de fois on autorise la prise en compte de la « flèche

haut » avant que le cube ne redescende et saut qui indiquera à chaque instant le nombre de sauts déjà effectués avant que le cube ne redescende ou qu'il ne touche du noir.

Au démarrage « saut » vaut 0 et « nbSautMax » est initialisée à une valeur qui doit être ajustée par le programmeur, en fonction des niveaux de jeu qu'il a créés.

Nous modifions la prise en compte de la touche « flèche haut » :

Et nous remettons la valeur de saut à 0, lorsque le cube touche du noir lors d'un déplacement vertical :

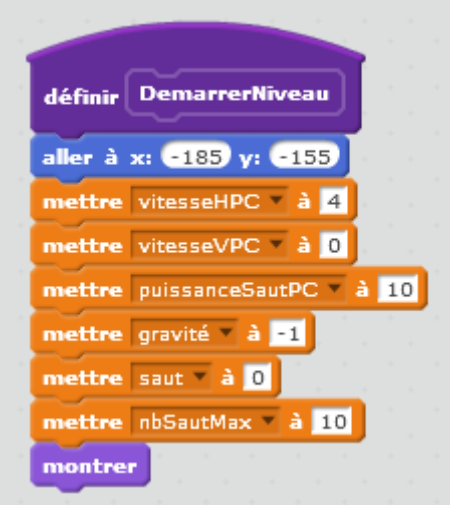


La restructuration du programme

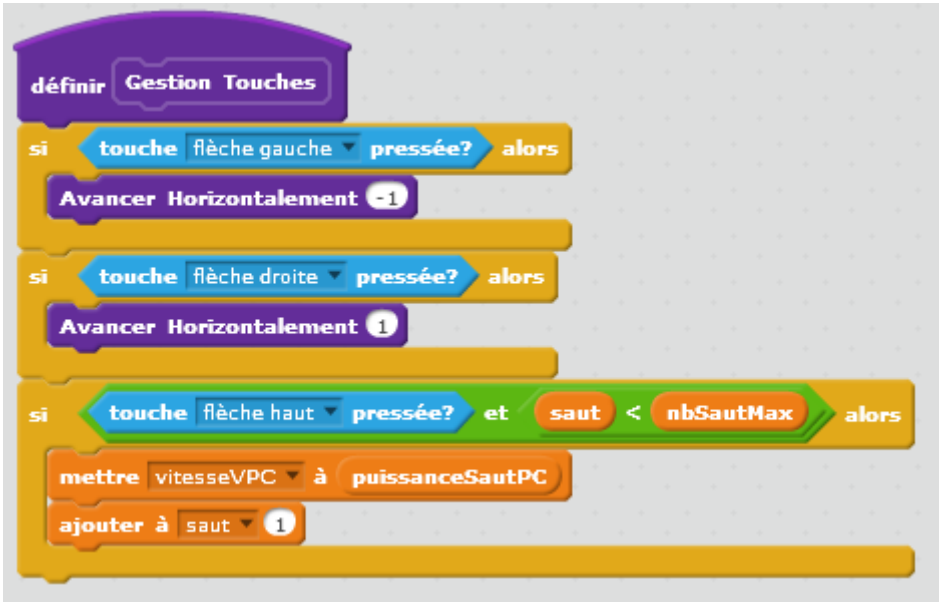
Pour permettre une réutilisation facile des différentes parties d'un jeu de plateforme, nous structurons notre ébauche de jeu en séparant les différentes phases dans des blocs.

Nous créons trois nouveaux blocs :

- Le bloc « Démarrer Niveau » qui regroupe toutes les instructions d'initialisation et de mise en place du cube, lorsque l'on passe au niveau suivant.



- Le bloc « Gestion des touches » qui s'occupe de capturer l'appui sur une des trois touches utilisées.



- Un bloc « Particularités du jeu » qui gère tous les autres éléments du jeu : ici nous ne gérons que le passage au niveau suivant, mais nous ajouterons certainement le cas où sur le parcourt, il y a de la lave à sauter, ou de l'eau ou d'autres obstacles.

- La boucle de jeu devient :

Création d'un nouveau jeu

Il suffit maintenant de créer dans le lutin « niveaux » des costumes avec des difficultés que le cube devra franchir.

Dans le lutin cube, les ajustements au nouveau jeu, se feront uniquement dans les blocs : « Démarrer Niveau » et « Particularités du jeu ». Tous les autres blocs, restent inchangés quelque soit le jeu. (Si l'on choisit d'autres touches pour faire bouger le cube, il faudra bien sur modiffier le bloc de gestion des touches)

Dès qu'un nouveau niveau est créé, il faut tester s'il est jouable, c'est-à-dire si le joueur a une chance d'arriver à passer au niveau suivant. Si un obstacle est trop haut pour que le cube puisse le franchir, on peut ajuster les variables « puissanceSaut » et « nbSautMax » pour que cela devienne possible, ou bien on met une plateforme intermédiaire permettant de franchir l'obstacle.

Si l'on remplace le cube par un lutin ayant une forme moins régulière, on peut être amené à ajuster le recul lorsque le lutin touche un mur, ajuster aussi l'endroit où les coordonnées du lutin sont calculées.

Voici par exemple comment est placée l'origine du petit cube :