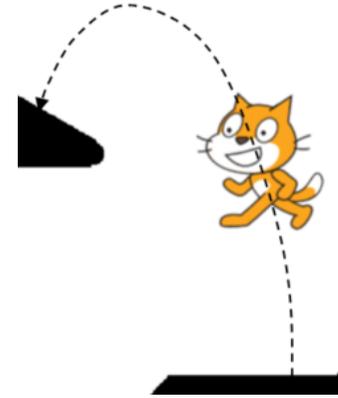


# Jeux de plateforme

## V3 : sauts et gravité

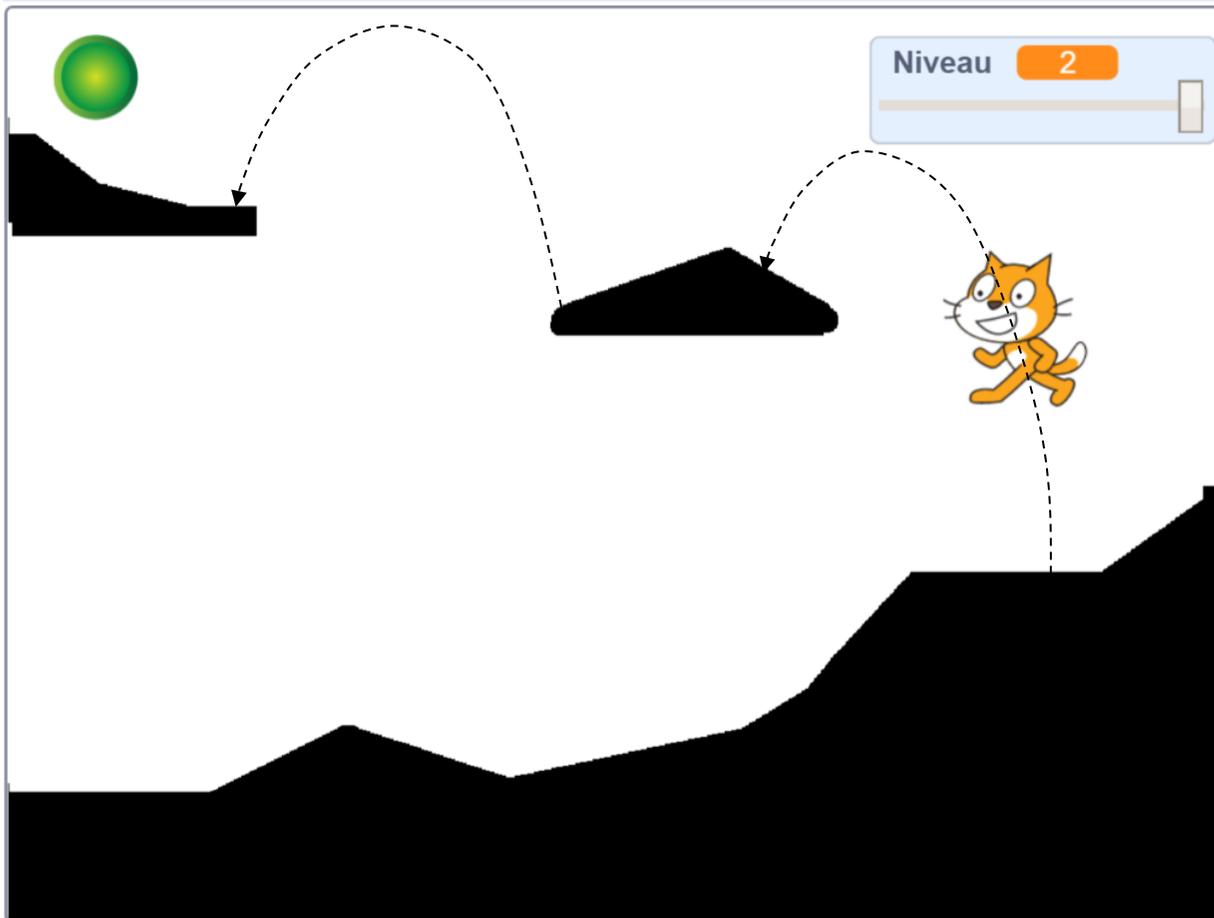


Pierre Huguet

mail : [Pierre@onvaessayer.org](mailto:Pierre@onvaessayer.org)

site : <https://onvaessayer.org/scratch>

## V3 : SAUTER ET REBONDIR



Dans la V3 : on fait sauter le sprite avec la flèche haut :

- s'il touche le relief en montant, il doit rebondir vers le bas,
- s'il touche le relief en descendant, le saut est fini, il repasse en suivi de terrain.

# JEU DE PLATEFORME

## Réalisation d'un algorithme de suivi de terrain :

<https://onvaessayer.org/scratch?app=platform>

V1 : [initialisation et déplacement horizontal](https://onvaessayer.org/scratch?video=platformV1)

<https://onvaessayer.org/scratch?video=platformV1>

V2 : [déplacement avec suivi de terrain](https://onvaessayer.org/scratch?video=platformV2)

<https://onvaessayer.org/scratch?video=platformV2>

V3 : **[déplacement avec saut](https://onvaessayer.org/scratch?video=platformV3)**

<https://onvaessayer.org/scratch?video=platformV3>

V4 : [bonhomme animé et chute ralentie](https://onvaessayer.org/scratch?video=platformV4)

<https://onvaessayer.org/scratch?video=platformV4>

V5 : [panoramique terrain de jeu illimité](https://onvaessayer.org/scratch?video=platformV5)

<https://onvaessayer.org/scratch?video=platformV5>

# V3 : NOUVELLES VARIABLES

Initialiser variables et sprite

envoyer message "niveau" et attendre

aller à x,y départ  
mode rotation gauche-droite  
mettre vitesse horizontale à 5

**mettre VITESSE Y SAUT à 15**  
**mettre GRAVITE à -1**  
**mettre saut? à faux**  
**Mettre vitesseY à 0**

numéro costume à 1  
basculer vers numéro costume

Nouvelles variables suivantes :  
(procédure d'initialisation)

**VITESSE Y SAUT = 15**

**GRAVITE = -1**

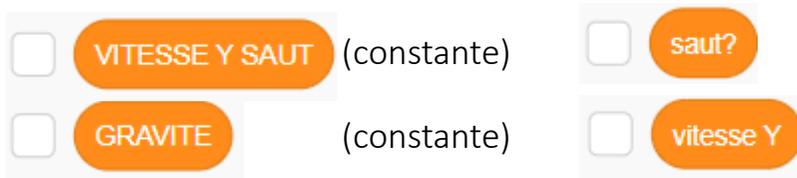
**saut? = faux** au départ

**vitesseY = 0** au départ

La gravité est la quantité que l'on ajoute à la vitesse verticale pour qu'elle diminue, devienne négative et que le sprite retombe

# V3 : NOUVELLES VARIABLES ET NOUVEAU SON

Nouvelles variables



Nouveau son



Définissez ces variables, elles peuvent être "pour ce sprite seulement" et ajoutez le son au sprite.

# V3 : INITIALISER VARIABLES DANS PROCÉDURE INITIALISER

Nouvelles variables



Nouveau son



Définissez ces variables, elles peuvent être "pour ce sprite seulement" et ajoutez le son au sprite.

Ajouts à la procédure d'initialisation



# V3 : INITIALISER VARIABLES DANS PROCÉDURE INITIALISER

Après avoir déclaré les variables, modifiez la procédure "Initialiser" avec les blocs suivants.

The image shows a Scratch script for an 'Initialiser' procedure. The blocks are as follows:

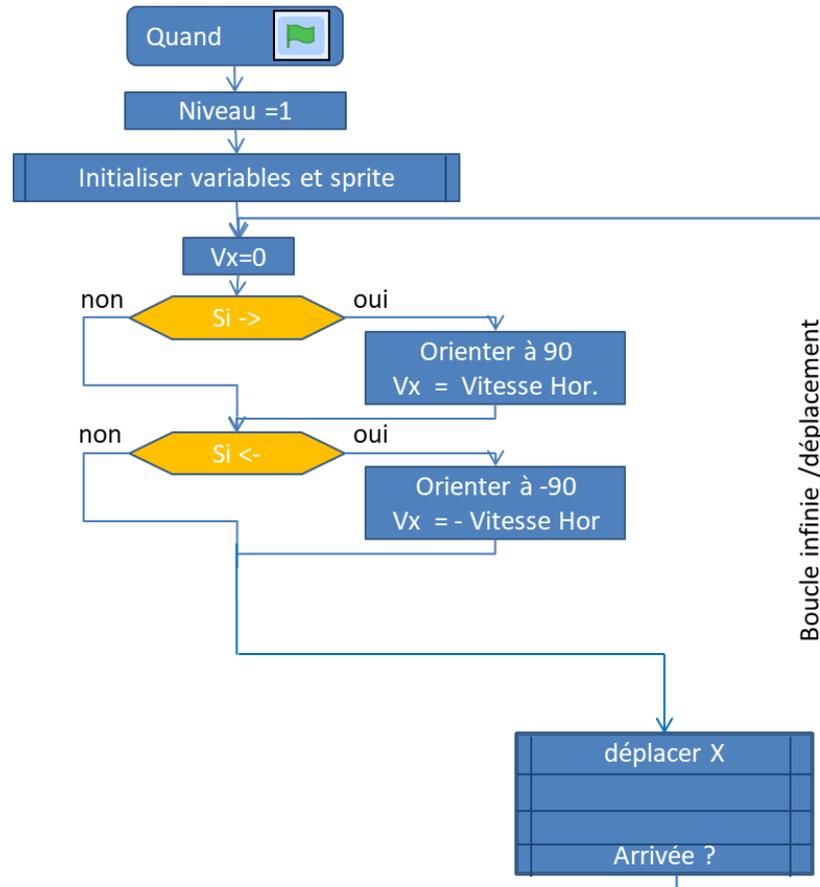
- définir initialiser** (pink block)
- niveau** (pink block)
- aller à x: x départ y: y départ** (blue block)
- fixer le sens de rotation gauche-droite** (blue block)
- mettre VITESSE HORIZONTALE à 5** (orange block) with a tooltip: "c'est la vitesse du lutin. vers la gauche ou la"
- mettre VITESSE Y SAUT à 15** (orange block) with a tooltip: "vitesse initiale du lutin au début de chaque"
- mettre GRAVITE à -1** (orange block) with a tooltip: "quantité déduite de Vitesse Y à chaque pas"
- mettre saut? à faux** (orange block) with a tooltip: "'vrai' ou 'faux' selon que le sprite est dans un saut"
- mettre vitesse Y à 0** (orange block)
- mettre vitesse X à 0** (orange block)
- mettre numero costume à 1** (orange block)
- basculer sur le costume numero costume** (purple block)

Nouveaux blocs

# V3 : PSEUDO CODE ET DIAGRAMME DE LA VERSION 2

Quand clic sur drapeau vert

- Mettre **niveau** à 1
- Appeler la procédure **initialiser**
- Répéter indéfiniment
  - mettre **vitesseX** à 0
  - si flèche droite pressée
    - Orienter à 90°
    - **VitesseX** = vitesse horizontale
  - si flèche gauche pressée
    - Orienter à -90°
    - **VitesseX** = - vitesse horizontale
- appeler procédure **déplacer X**
- Appeler procédure **arrivée ?**



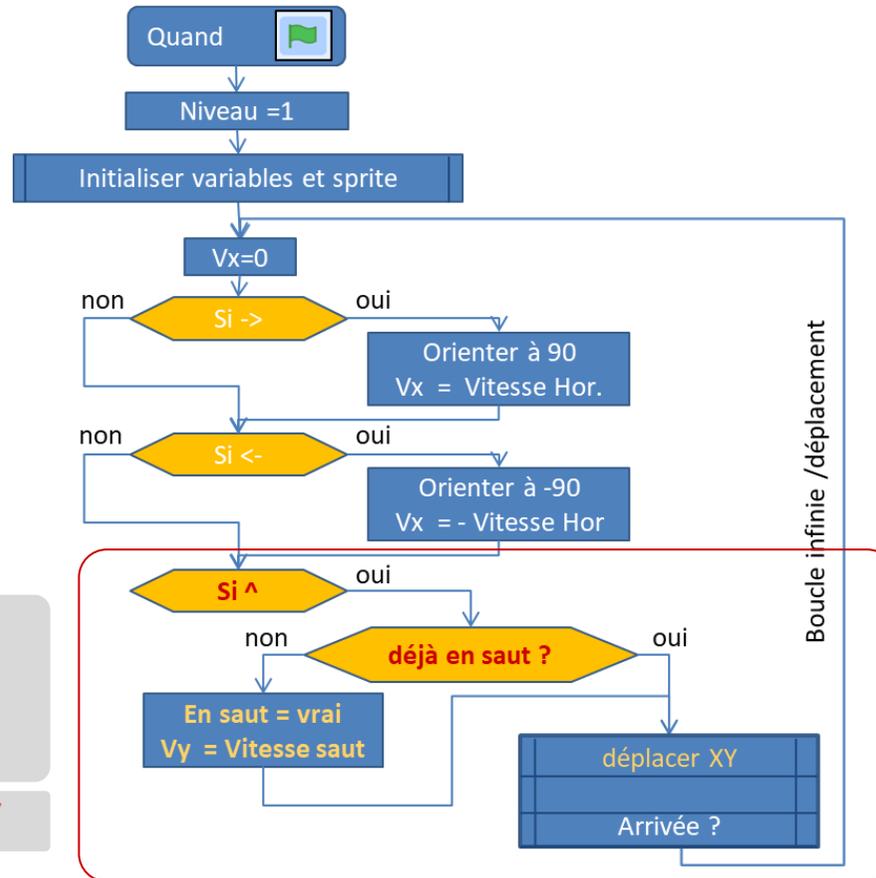
V2

déplacer X

# V3 : AJOUT "FLÈCHE HAUT"

Quand clic sur drapeau vert

- Mettre **niveau** à 1
- Appeler la procédure **initialiser**
- Répéter indéfiniment
  - mettre **vitesseX** à 0
  - si flèche droite pressée
    - Orienter à 90°
    - **VitesseX** = vitesse horizontale
  - si flèche gauche pressée
    - Orienter à -90°
    - **VitesseX** = - vitesse horizontale
  - si flèche haute pressée
    - Si PAS déjà en saut (**saut?** = faux)
      - **Saut?** = vrai
      - **vitesseY** = **VITESSE Y SAUT**
- appeler procédure **déplacer XY**
- appeler procédure **arrivée ?**



V3

déplacer XY

# V3 : AJOUT "FLÈCHE HAUT"

Quand clic sur drapeau vert

- Mettre **niveau** à 1
- Appeler la procédure **initialiser**
- Répéter indéfiniment
  - mettre **vitesseX** à 0
  - si flèche droite pressée
    - Orienter à 90°
    - **VitesseX** = vitesse horizontale
  - si flèche gauche pressée
    - Orienter à -90°
    - **VitesseX** = - vitesse horizontale
  - si flèche haute pressée
    - Si PAS déjà en saut (**saut?** = faux)
      - **Saut?** = vrai
      - **vitesseY** = **VITESSE Y SAUT**
- appeler procédure **déplacer XY**
- appeler procédure **arrivée ?**

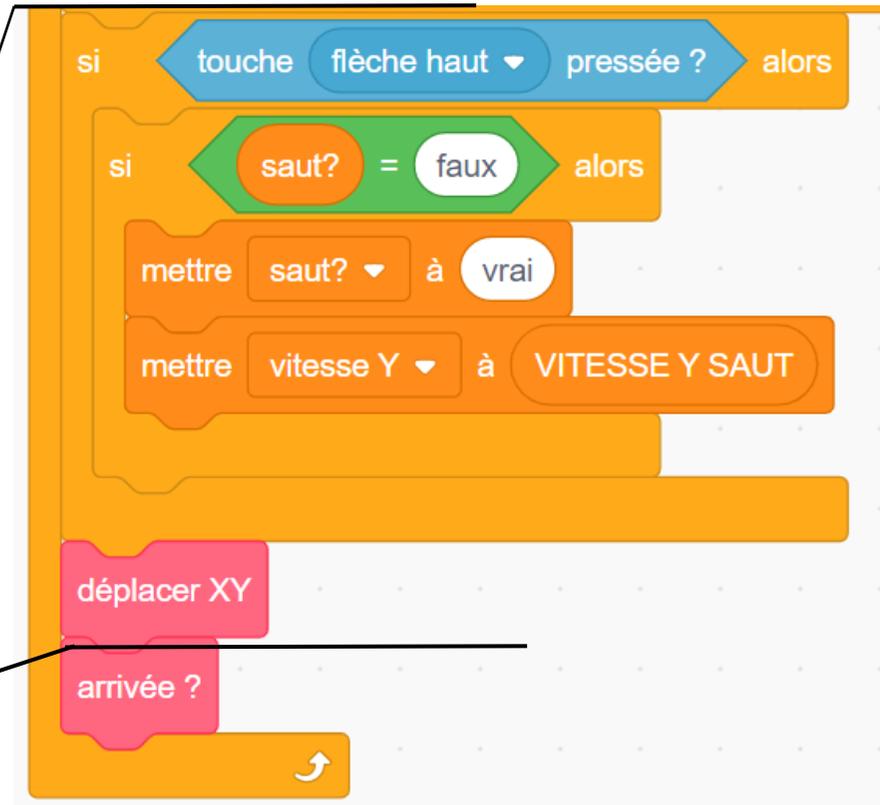
```
quand le drapeau vert est cliqué
mettre Niveau à 1
initialiser
répéter indéfiniment
mettre vitesse X à 0
si touche flèche droite pressée ? alors
  s'orienter à 90
  mettre vitesse X à VITESSE HORIZONTALE
si touche flèche gauche pressée ? alors
  s'orienter à -90
  mettre vitesse X à VITESSE HORIZONTALE * -1
si touche flèche haut pressée ? alors
  si saut? = faux alors
    mettre saut? à vrai
    mettre vitesse Y à VITESSE Y SAUT
déplacer XY
arrivée ?
```

V3

# V3 : AJOUT "FLÈCHE HAUT"

Quand clic sur drapeau vert

- Mettre **niveau** à 1
- Appeler la procédure **initialiser**
- Répéter indéfiniment
  - mettre **vitesseX** à 0
  - si flèche droite pressée
    - Orienter à 90°
    - **VitesseX** = vitesse horizontale
  - si flèche gauche pressée
    - Orienter à -90°
    - **VitesseX** = - vitesse horizontale
  - si flèche haute pressée
    - Si PAS déjà en saut (**saut?** = faux)
      - **Saut?** = vrai
      - **vitesseY** = **VITESSE Y SAUT**
- appeler procédure **déplacer XY**
- appeler procédure **arrivée ?**



V3

# V3 : MODIFICATION DU SCRIPT PRINCIPAL (DRAPEAU VERT)

Modification de la fin du script principal

et on renomme la  
procédure déplacerX

The image shows a Scratch script and a procedure call. The script is as follows:

```
si [touche flèche haut ▼ pressée ?] alors
  si [saut? = faux] alors
    mettre [saut?] à [vrai]
    mettre [vitesse Y ▼] à [VITESSE Y SAUT]
  déplacer XY
  arrivée ?
```

The procedure call is:

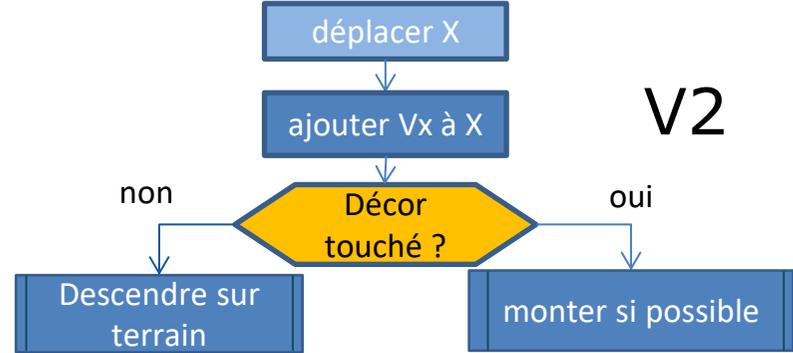
```
définir [déplacer XY]
```

Two callout boxes provide explanations:

- The first callout points to the 'si saut? = faux' block: "Si flèche haut est enfoncée, on vérifie d'abord que le sprite n'est pas déjà en train de sauter et ça n'est pas le cas : on met la vitesse verticale (vitesse Y) à sa valeur de saut et on indique que le lutin est en train de sauter en mettant la variable "Saut?" à vrai".
- The second callout points to the 'déplacer XY' block: "appel à la procédure "déplacer XY" qui va déplacer le lutin horizontalement et verticalement en suivi de terrain et pour les sauts. (procédure où on coche l'option "exécuter sans rafraichir")".

## V3 : MODIFIER DÉPLACER X => DÉPLACER XY AVEC SAUTS

- ajouter Vx à X
- si décor touché
  - appeler "monter si possible"
- sinon:
  - appeler "descendre sur terrain"

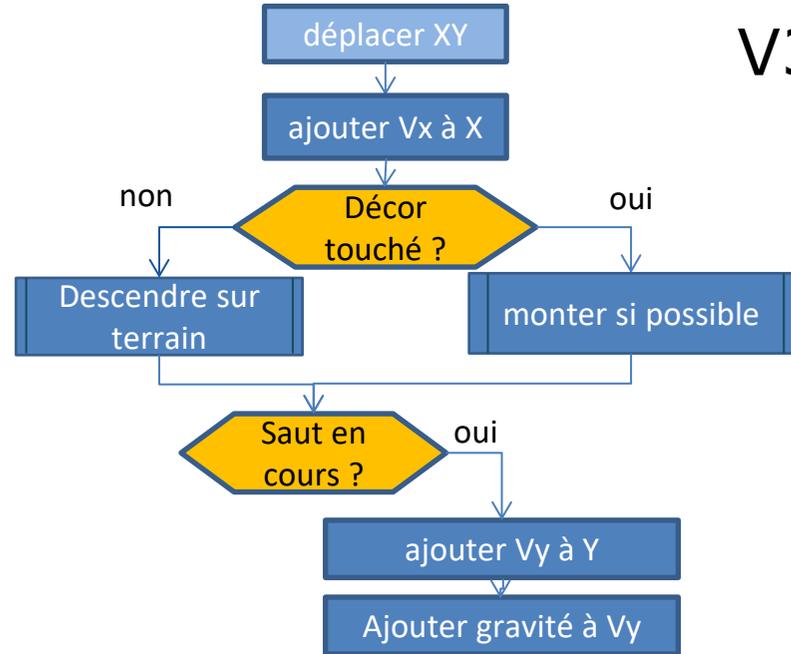


# V3 : PASSER DE DÉPLACER X À DÉPLACER XY AVEC SAUTS

V3



- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- sinon:
  - appeler "descendre sur terrain"
- si saut en cours
  - ajouter  $vitesseY$   $Y$
  - ajouter **GRAVITE** à  $vitesseY$

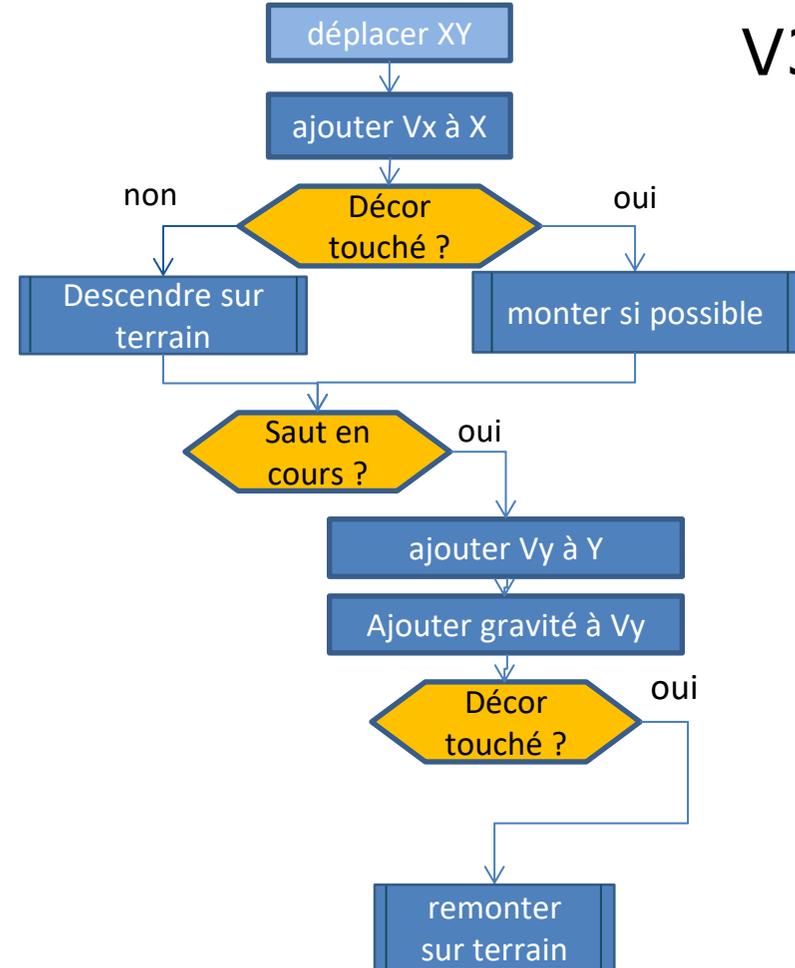


# V3 : PASSER DE DÉPLACER X À DÉPLACER XY AVEC SAUTS

V3



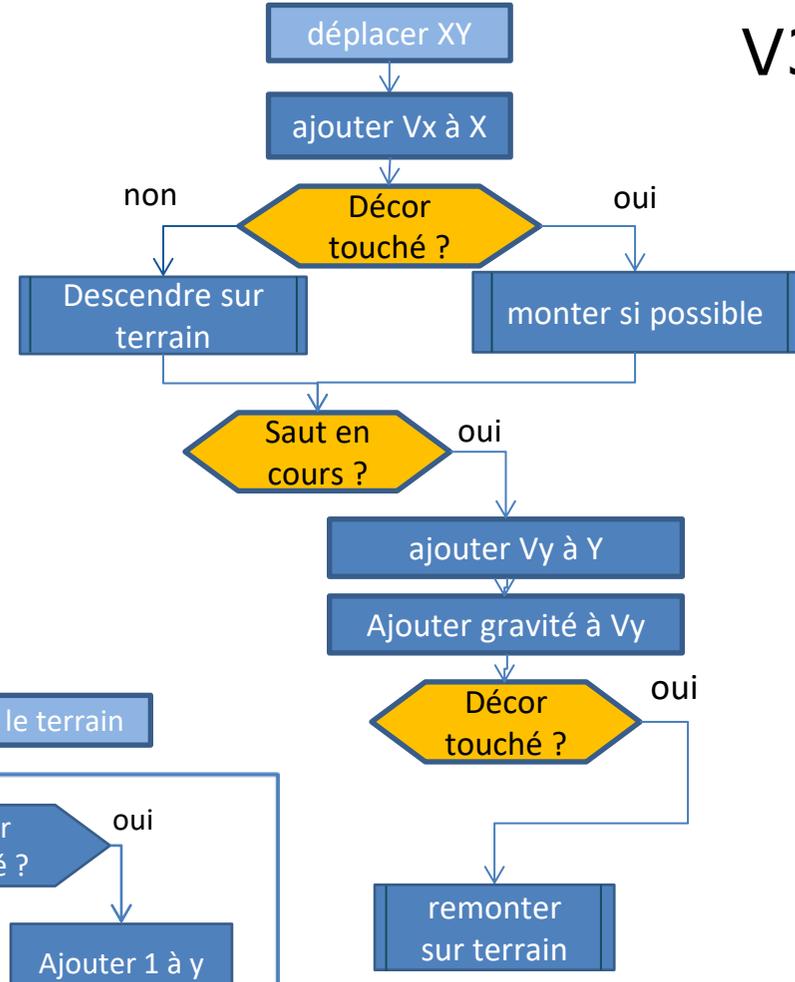
- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- sinon:
  - appeler "descendre sur terrain"
- si saut en cours
  - ajouter **vitesseY**  $Y$
  - ajouter **GRAVITE** à **vitesseY**
  - si décor touché (noir)
    - remonter sur le terrain



# V3 : PASSER DE DÉPLACER X À DÉPLACER XY AVEC SAUTS

V3

- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- sinon:
  - appeler "descendre sur terrain"
- si saut en cours
  - ajouter **vitesseY**  $Y$
  - ajouter **GRAVITE** à **vitesseY**
  - si décor touché (noir)
    - remonter sur le terrain



Remonter sur le terrain

- répéter jusqu'à décor **NON** touché
  - ajouter 1 à  $Y$
- mettre **vitesseY** à 0
- mettre **saut?** à faux

# V3 : CODAGE ET TESTS

- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- sinon:
  - appeler "descendre sur terrain"
- si saut en cours
  - ajouter  $vitesseY$   $Y$
  - ajouter **GRAVITE** à  $vitesseY$
  - si décor touché (noir)
    - appeler "remonter sur le terrain"



Remonter sur le terrain

- répéter jusqu'à décor **NON** touché  
ajouter 1 à  $Y$
- mettre  $vitesseY$  à 0  
mettre  $saut?$  à faux

```
définir déplacer XY
ajouter vitesse X à x
si couleur [noir] touchée ? alors
  monterSiPossible
sinon
  descendreSurTerrain
si saut? = vrai alors
  ajouter vitesse Y à y
  ajouter GRAVITE à vitesse Y
  si couleur [noir] touchée ? alors
    remonterSurTerrain
```

```
définir remonterSurTerrain
répéter jusqu'à ce que non couleur [noir] touchée ?
  ajouter 1 à y
mettre vitesse Y à 0
mettre saut? à faux
```

# V3 : CODAGE ET TESTS



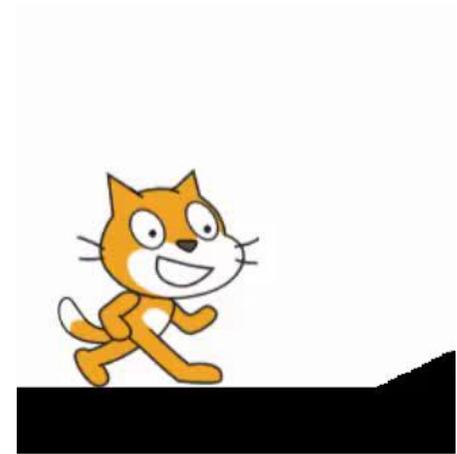
- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- sinon:
  - appeler "descendre sur terrain"
- si saut en cours
  - ajouter **vitesseY**  $Y$
  - ajouter **GRAVITE** à **vitesseY**
  - si décor touché (noir)
    - appeler "remonter sur le terrain"



Remonter sur le terrain

- répéter jusqu'à décor **NON** touché
  - ajouter 1 à  $Y$
- mettre **vitesseY** à 0
- mettre **saut?** à faux

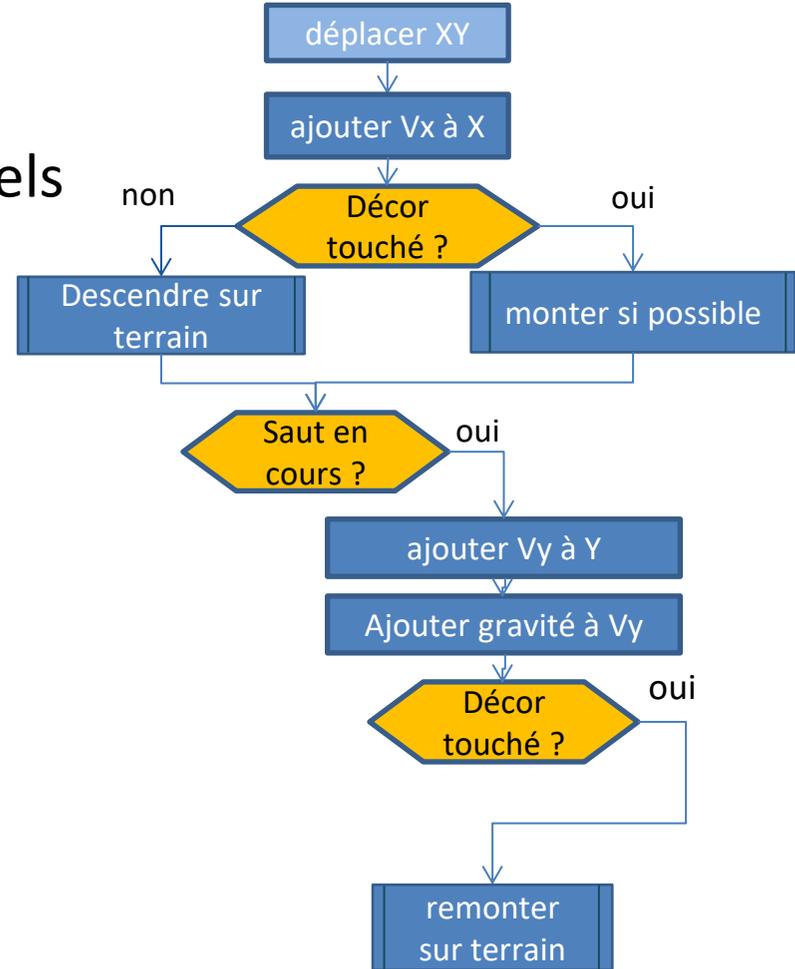
```
définir déplacer XY
ajouter vitesse X à x
si couleur [noir] touchée ? alors
  monterSiPossible
sinon
  descendreSurTerrain
si saut? = vrai alors
  ajouter vitesse Y à y
  ajouter GRAVITE à vitesse Y
  si couleur [noir] touchée ? alors
    remonterSurTerrain
```



```
définir remonterSurTerrain
répéter jusqu'à ce que non couleur [noir] touchée ?
  ajouter 1 à y
mettre vitesse Y à 0
mettre saut? à faux
```

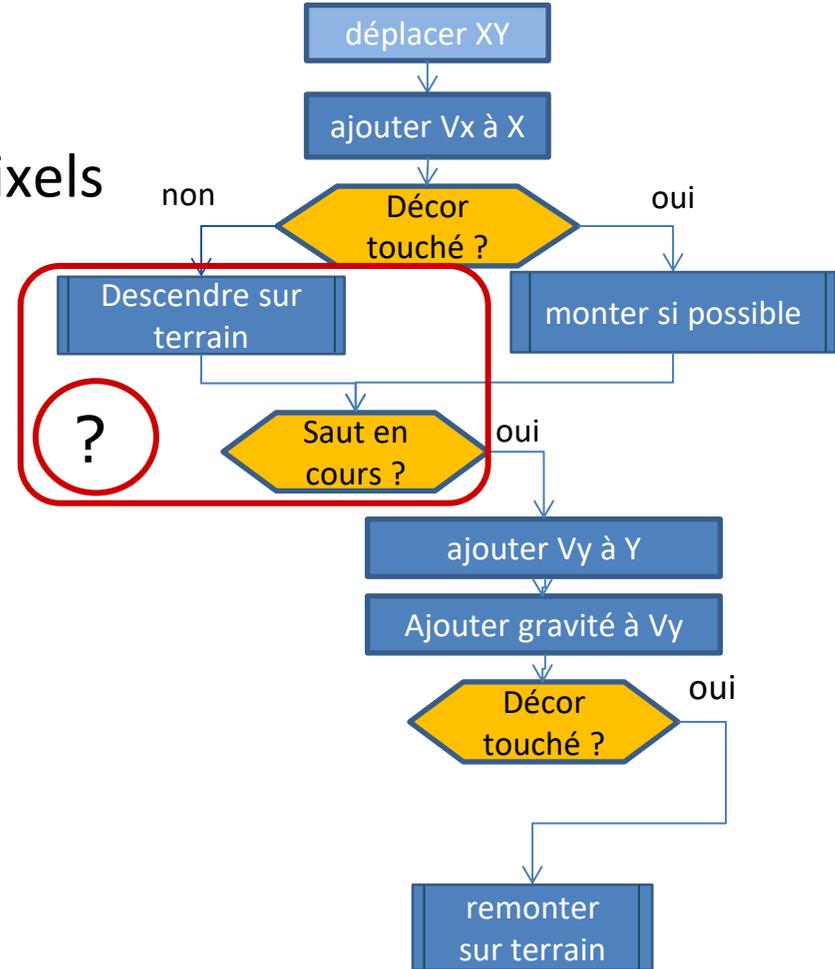
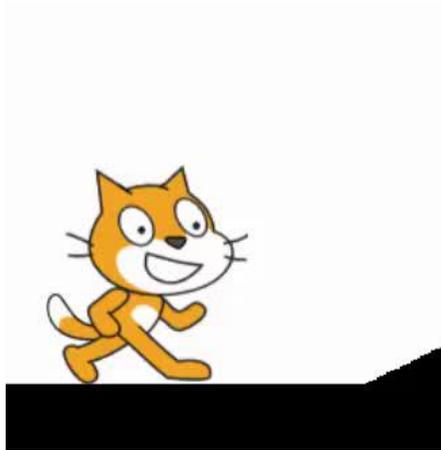
# V3 : BUG / SAUT DU SPRITE

- Bug :
  - Le sprite devrait monter de 120 pixels  
33% de l'écran :  $15+14+13+12+11+\dots$
  - Il monte de  $\sim 15$  pixels ( $< 5\%$  écran)
- Pourquoi ?



# V3 : CORRECTION DU BUG / SAUT DU SPRITE

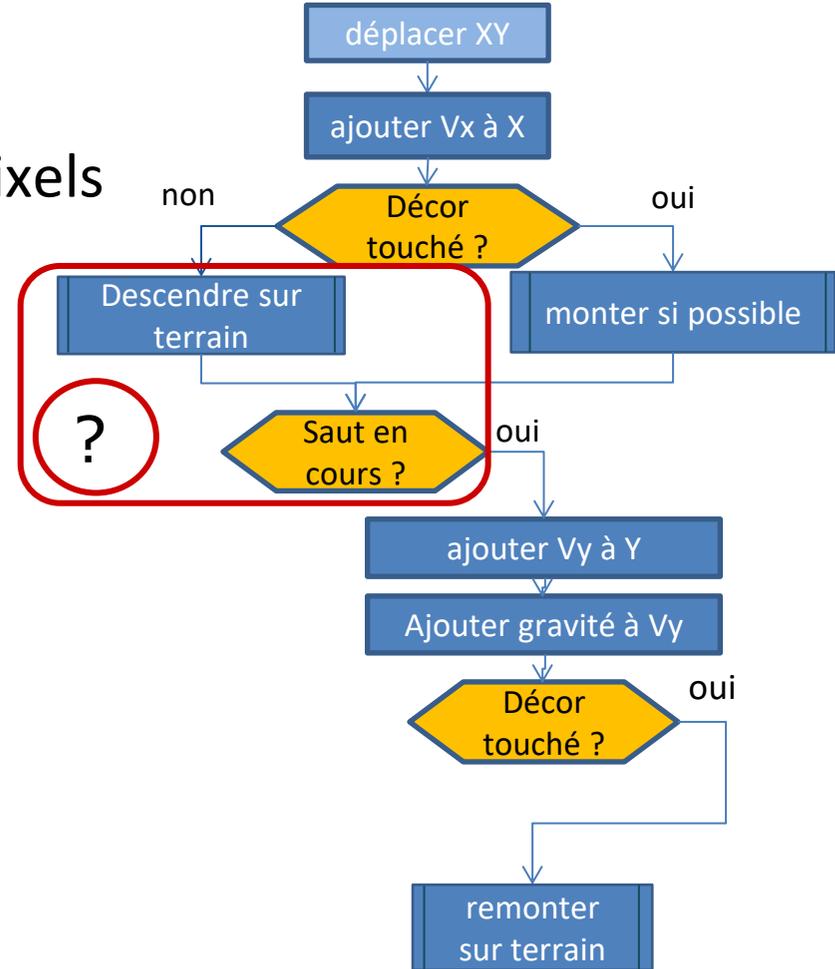
- Bug :
  - Le sprite devrait monter de 120 pixels  
33% de l'écran :  $15+14+13+12+11+\dots$
  - Il monte de  $\sim 15$  pixels ( $< 5\%$  écran)
- Pourquoi ?



# V3 : CORRECTION DU BUG / SAUT DU SPRITE

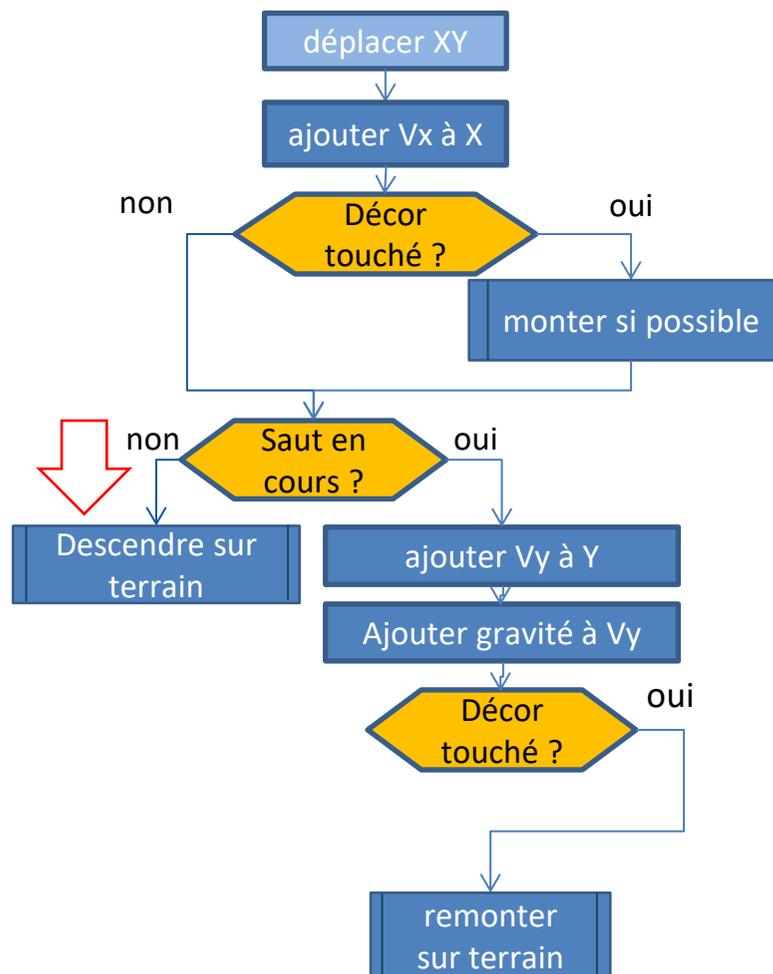
- Bug :
  - Le sprite devrait monter de 120 pixels  
33% de l'écran :  $15+14+13+12+11+\dots$
  - Il monte de  $\sim 15$  pixels ( $< 5\%$  écran)

## • Pourquoi ?



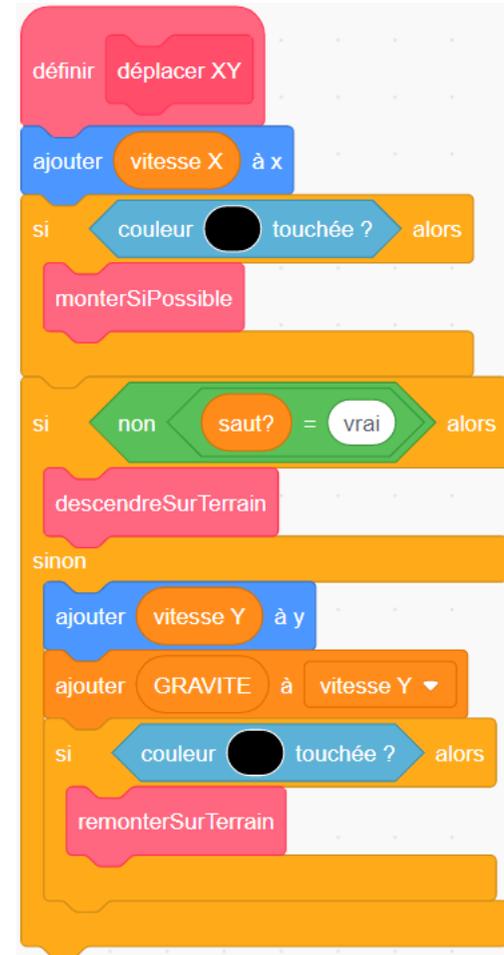
# V3 : CORRECTION DU BUG / SAUT DU SPRITE

- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- ~~• sinon :~~
- ~~• appeler "descendre sur terrain"~~
- si saut n'est pas en cours
  - appeler "descendre sur terrain"
- Sinon :
  - ajouter **vitesseY**  $Y$
  - ajouter **GRAVITE** à **vitesseY**
  - si décor touché (noir)
    - appeler "remonter sur le terrain"



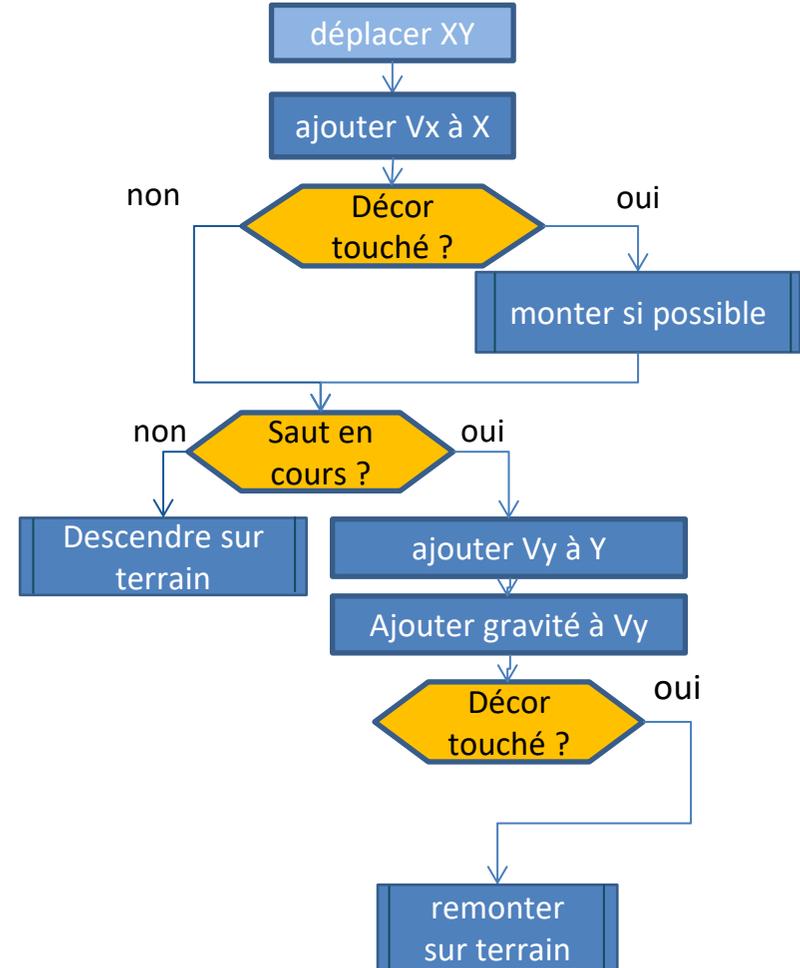
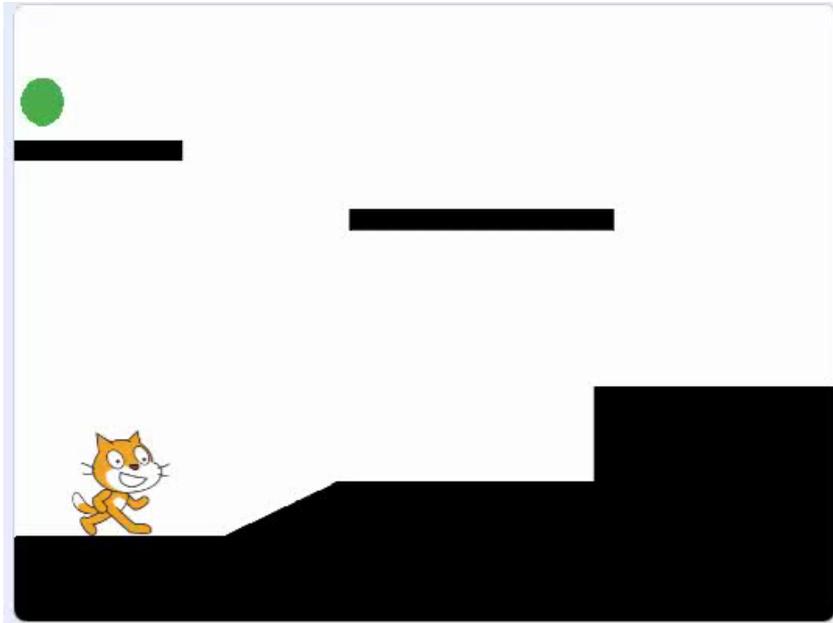
# V3 : VERSION CORRIGÉE / SAUT DU SPRITE

- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- si saut n'est pas en cours ( $saut?$  n'est pas égal à vrai)
  - appeler "descendre sur terrain"
- Sinon ( $saut? = vrai$ )
  - ajouter  $vitesseY$   $Y$
  - ajouter **GRAVITE** à  $vitesseY$
  - si décor touché (noir)
    - appeler "remonter sur le terrain"



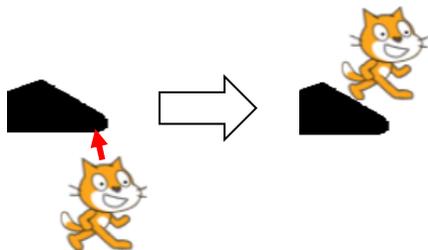
# V3 : TESTS DE LA VERSION CORRIGÉE

- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"

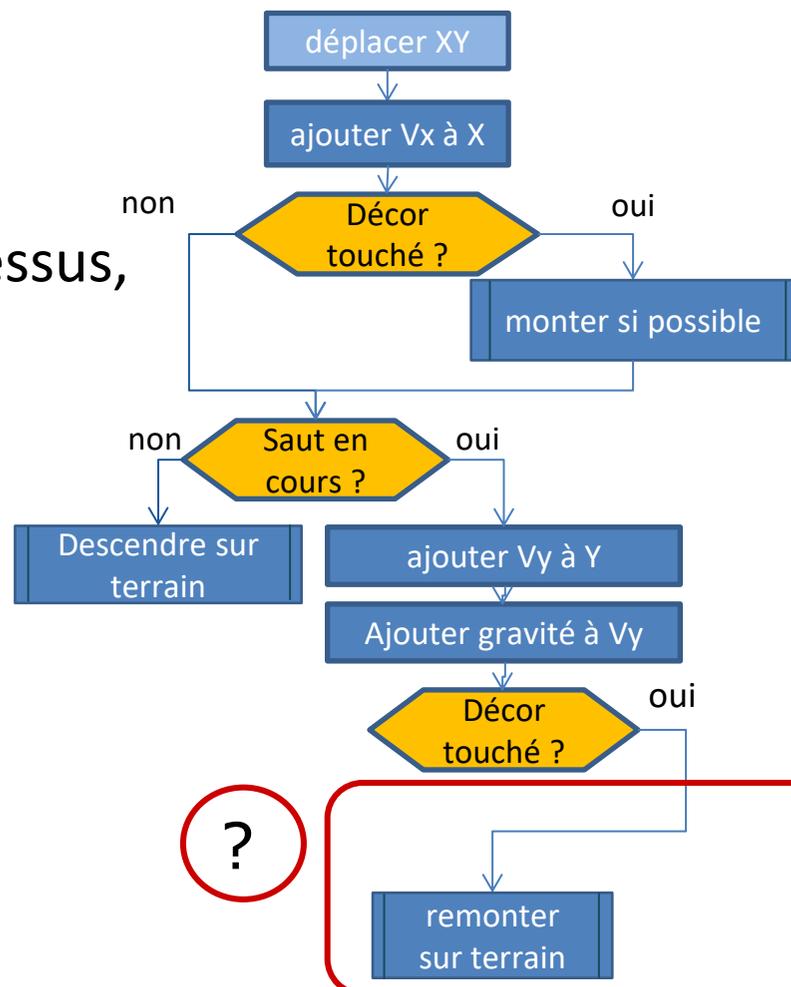


# V3 : BUG / LE SPRITE PASSE AU DESSUS DES OBSTACLES

- Bug :
  - Quand le sprite se cogne contre un obstacle en montant il passe par dessus, alors qu'il devrait rebondir ...

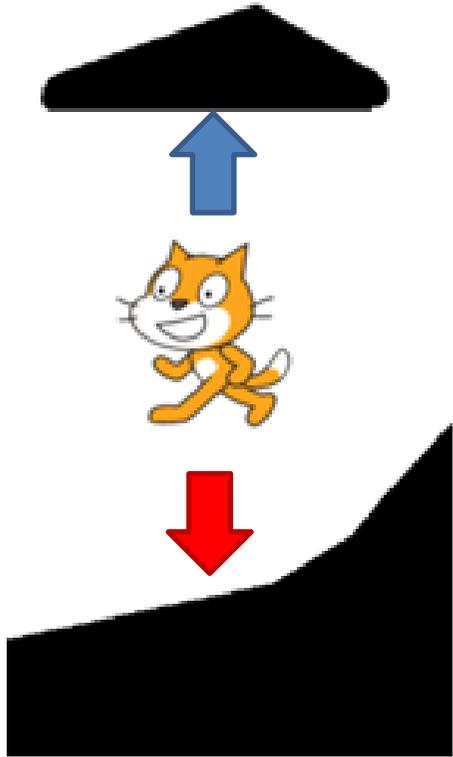


- **Comment faire ?**
  - Selon le signe de Vitesse Y



## V3 : CORRECTION DE BUG / COLLISIONS AVEC OBSTACLES

Pendant un saut, si le joueur touche le décor

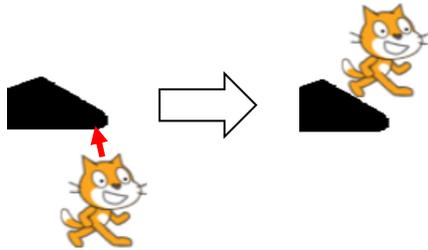


- en montant :  $V_y > 0$  
  - il doit redescendre, jusqu'à ce qu'il ne touche plus (puis il va tomber avec la gravité)
- en descendant :  $V_y \leq 0$  
  - il doit remonter, jusqu'à ce qu'il ne touche plus
  - et le saut est fini
- et on met Vitesse  $V_y = 0$

# V3 : CORRECTION DE BUG / COLLISIONS AVEC OBSTACLES

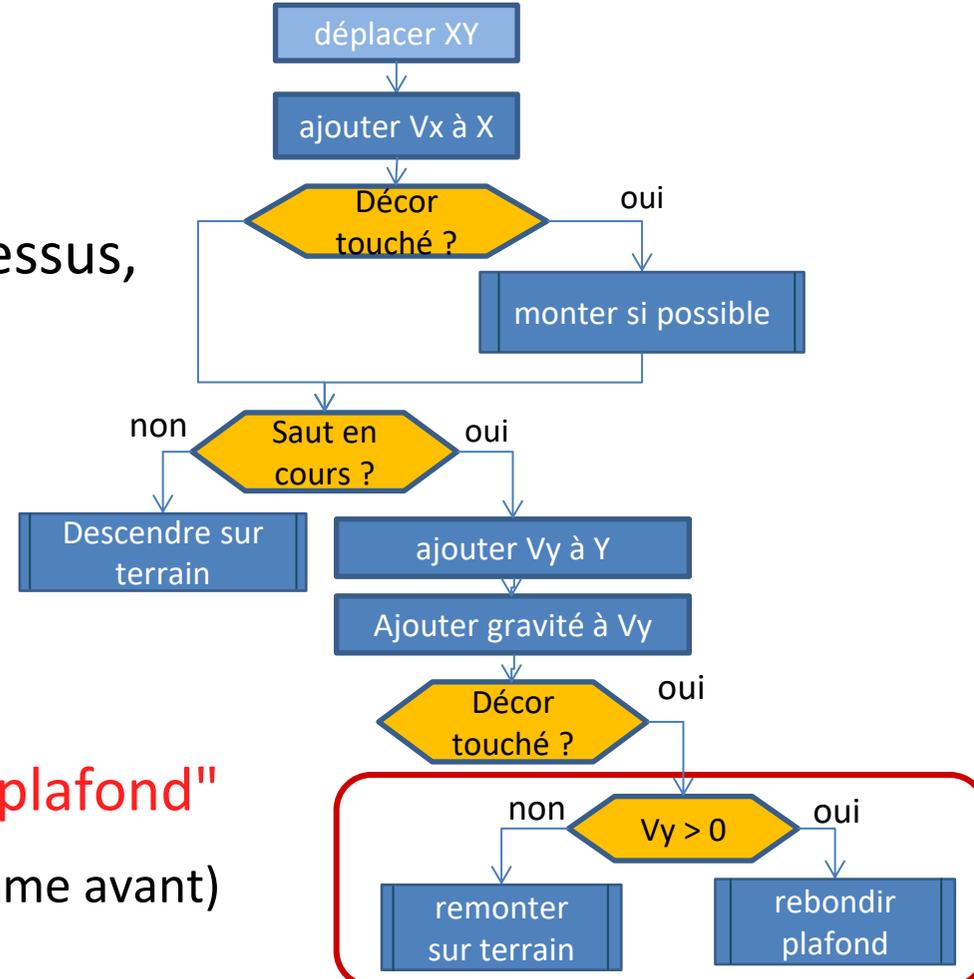
- Bug :

- Quand le sprite se cogne contre un obstacle en montant il passe par dessus, alors qu'il devrait rebondir



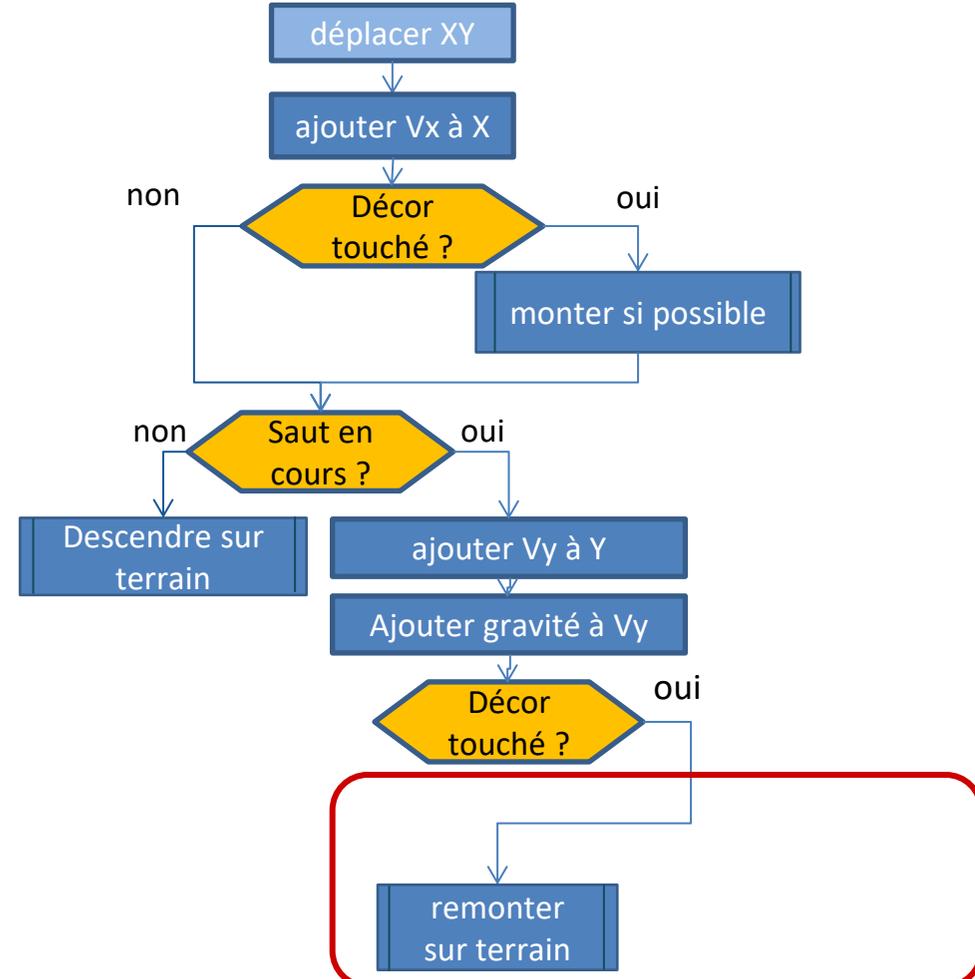
- **Solution**

- si Vitesse Y > 0 : appeler "rebondir plafond"
- sinon : "remonter sur terrain" (comme avant)



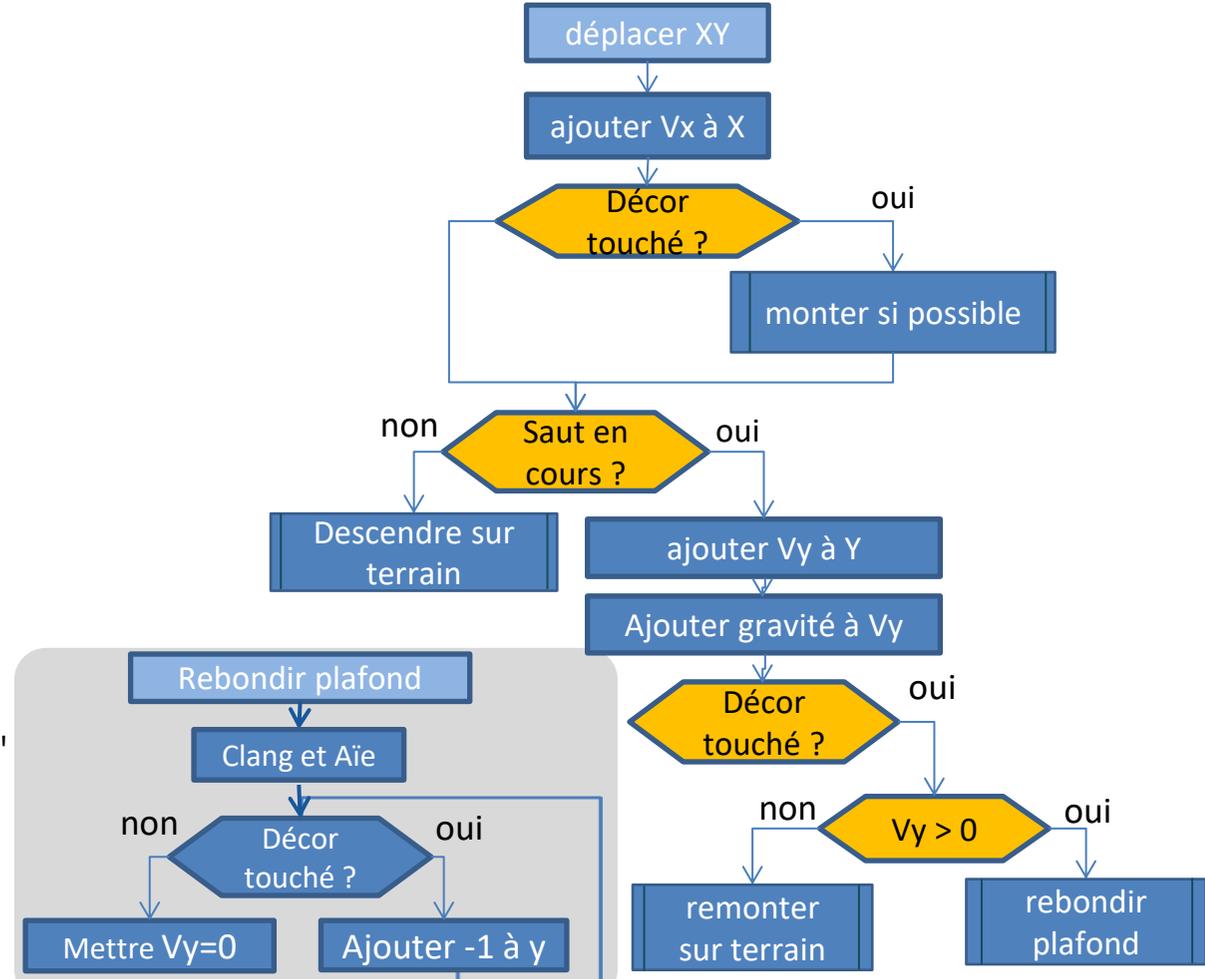
# V3 : CORRECTION DE BUG / COLLISIONS AVEC OBSTACLES

- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- si saut en cours
  - ajouter **vitesseY**  $Y$
  - ajouter **GRAVITE** à **vitesseY**
  - si décor touché (noir)
    - appeler "remonter sur le terrain"
- sinon:
  - appeler "descendre sur terrain"



# V3 : CORRECTION DE BUG / COLLISIONS AVEC OBSTACLES

- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- si saut en cours
  - ajouter **vitesseY**  $Y$
  - ajouter **GRAVITE** à **vitesseY**
  - si décor touché (noir)
    - si **vitesseY**  $> 0$ 
      - appeler "rebondir plafond"
    - sinon
      - appeler "remonter sur terrain"
- sinon:
  - appeler "descendre sur terrain"



# V3 : CORRECTION BUG

- ajouter  $V_x$  à  $X$
- si décor touché
  - appeler "monter si possible"
- si saut en cours (si **saut?** = vrai)
  - ajouter **vitesseY**  $Y$
  - ajouter **GRAVITE** à **vitesseY**
  - si décor touché (noir)
    - si **vitesseY**  $> 0$ 
      - appeler "rebondir plafond"
    - sinon
      - appeler "remonter sur terrain"
- sinon:
  - appeler "descendre sur terrain"

```
définir déplacer XY
ajouter vitesse X à x
si couleur [noir] touchée ? alors
  monterSiPossible
si non saut? = vrai alors
  descendreSurTerrain
sinon
  ajouter vitesse Y à y
  ajouter GRAVITE à vitesse Y
  si couleur [noir] touchée ? alors
    si vitesse Y > 0 alors
      rebondirPlafond
    sinon
      remonterSurTerrain
```

```
définir rebondirPlafond
jouer le son Clang
dire Aïe! pendant 0.1 secondes
répéter jusqu'à ce que non couleur [noir] touchée ?
  ajouter -1 à y
mettre vitesse Y à 0
```

# V3 : BLOCS DE LA VERSION 3

## Script du drapeau vert

quand le drapeau vert est cliqué

- mettre Niveau à 1
- initialiser
- répéter indéfiniment
  - mettre vitesse X à 0
  - si touche flèche droite pressée ? alors
    - s'orienter à 90
    - mettre vitesse X à VITESSE HORIZONTALE
  - si touche flèche gauche pressée ? alors
    - s'orienter à -90
    - mettre vitesse X à  $VITESSE HORIZONTALE * -1$

Annotations :

- On commence par le niveau 1
- On appellera cette procédure "initialiser" au début de chaque niveau
- boucle infinie de déplacement du sprite joueur
- Vitesse X = 0 avant de tester les flèches
- quand le joueur appuie sur la flèche droite mettre vitesse X à la vitesse horizontale (positive) et orienter le lutin à droite (90°) et le déplacer de vitesse X
- quand le joueur appuie sur la flèche gauche mettre vitesse X à l'opposé de vitesse horizontale (\* -1), et orienter le lutin à gauche (-90°)

## 2° partie

si touche flèche haut pressée ? alors

- si saut? = faux alors
  - mettre saut? à vrai
  - mettre vitesse Y à VITESSE Y SAUT
- déplacer XY
- arrivée ?

Annotations :

- Si flèche haut est enfoncée, on vérifie d'abord que le sprite n'est pas déjà en train de sauter et ça n'est pas le cas : on met la vitesse verticale (vitesse Y) à sa valeur de saut et on indique que le lutin est en train de sauter en mettant la variable "Saut?" à vrai
- appel à la procédure "déplacer XY" qui va déplacer le lutin horizontalement et verticalement en suivi de terrain et pour les sauts. (dans cette procédure où on coche l'option "exécuter sans rafraichir")

# V3 : BLOCS DE LA VERSION 3

## Script du drapeau vert (zoom)

```
quand le drapeau vert est cliqué
mettre Niveau à 1
initialiser
répéter indéfiniment
mettre vitesse X à 0
si touche flèche droite pressée ? alors
  s'orienter à 90
  mettre vitesse X à VITESSE HORIZONTALE
si touche flèche gauche pressée ? alors
  s'orienter à -90
  mettre vitesse X à VITESSE HORIZONTALE * -1
```

```
si touche flèche haut pressée ? alors
  si saut? = faux alors
    mettre saut? à vrai
    mettre vitesse Y à VITESSE Y SAUT
déplacer XY
arrivée ?
```

# V3 : BLOCS DE LA VERSION 3

### Procédure initialiser

The code for 'Procédure initialiser' consists of the following blocks:

- définir initialiser
- niveau
- aller à x: x départ y: y départ
- fixer le sens de rotation gauche-droite
- mettre VITESSE HORIZONTALE à 5
- mettre VITESSE Y SAUT à 15
- mettre GRAVITE à -1
- mettre saut? à faux
- mettre vitesse Y à 0
- mettre vitesse X à 0
- mettre numero costume à 1
- basculer sur le costume numero costume

Callouts explain the variables:

- VITESSE HORIZONTALE: c'est la vitesse du lutin, vers la gauche ou la droite
- VITESSE Y SAUT: vitesse initiale du lutin au début de chaque saut
- GRAVITE: quantité déduite de Vitesse Y à chaque pas
- numero costume: "vrai" ou "faux" selon que le sprite est dans un saut

### Procédure niveau

The code for 'Procédure niveau' consists of the following blocks:

- définir niveau
- basculer sur l'arrière-plan Niveau
- si Niveau = 1 alors
  - mettre x départ à -180
  - mettre y départ à -130
- si Niveau = 2 alors
  - mettre x départ à -180
  - mettre y départ à -127
- si Niveau = 3 alors
  - mettre x départ à -180
  - mettre y départ à -77

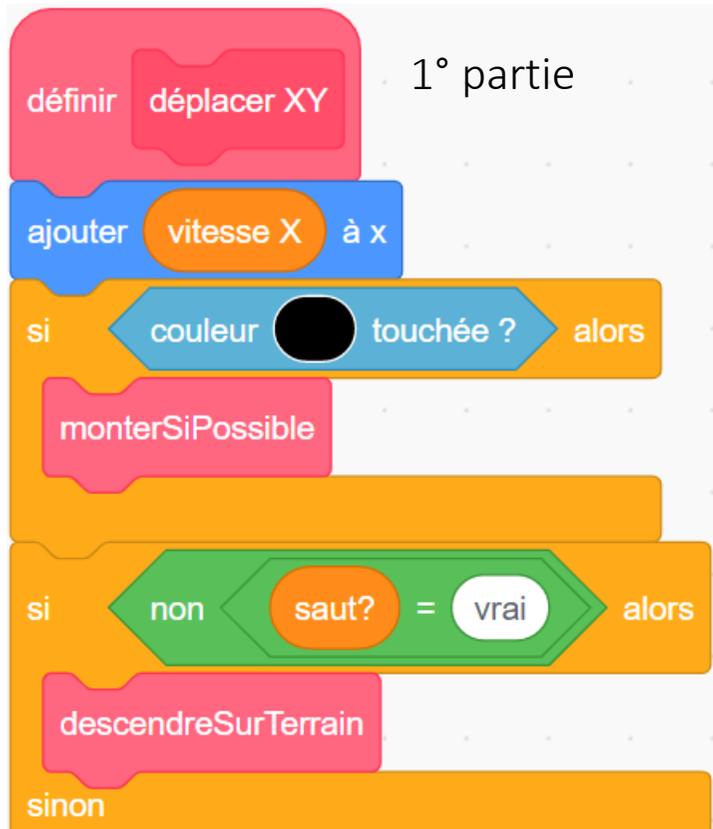
### Procédure arrivée?

The code for 'Procédure arrivée?' consists of the following blocks:

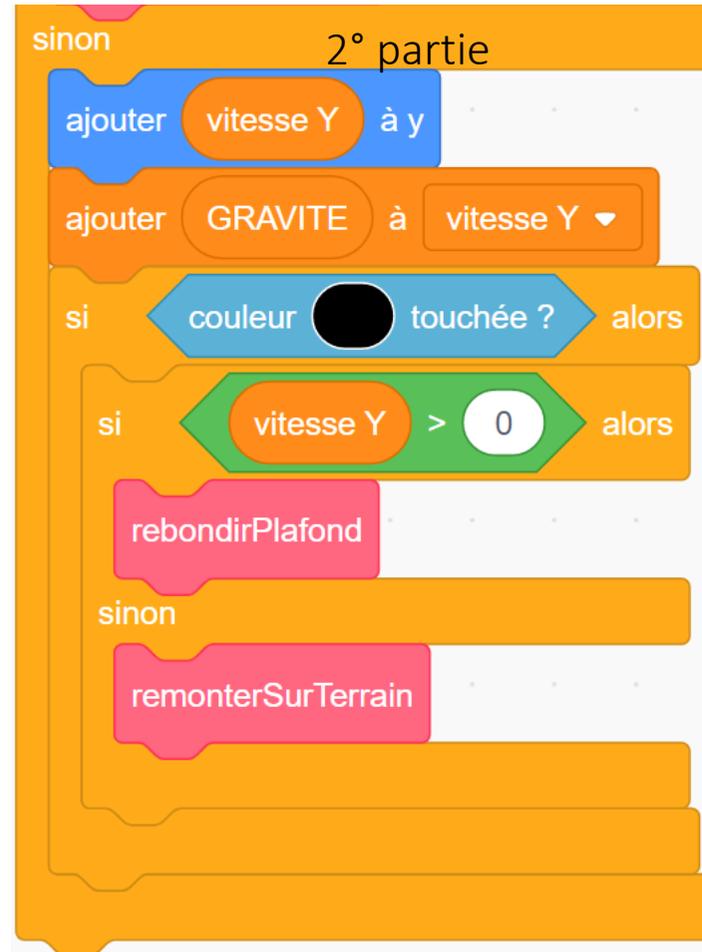
- définir arrivée ?
- si couleur touchée ? alors
  - ajouter 1 à Niveau
  - si Niveau > 3 alors
    - dire vous avez gagné pendant 2 secondes
    - stop autres scripts dans sprite
  - sinon
    - initialiser

# V3 : BLOCS DE LA VERSION 3

Procédure déplacer XY

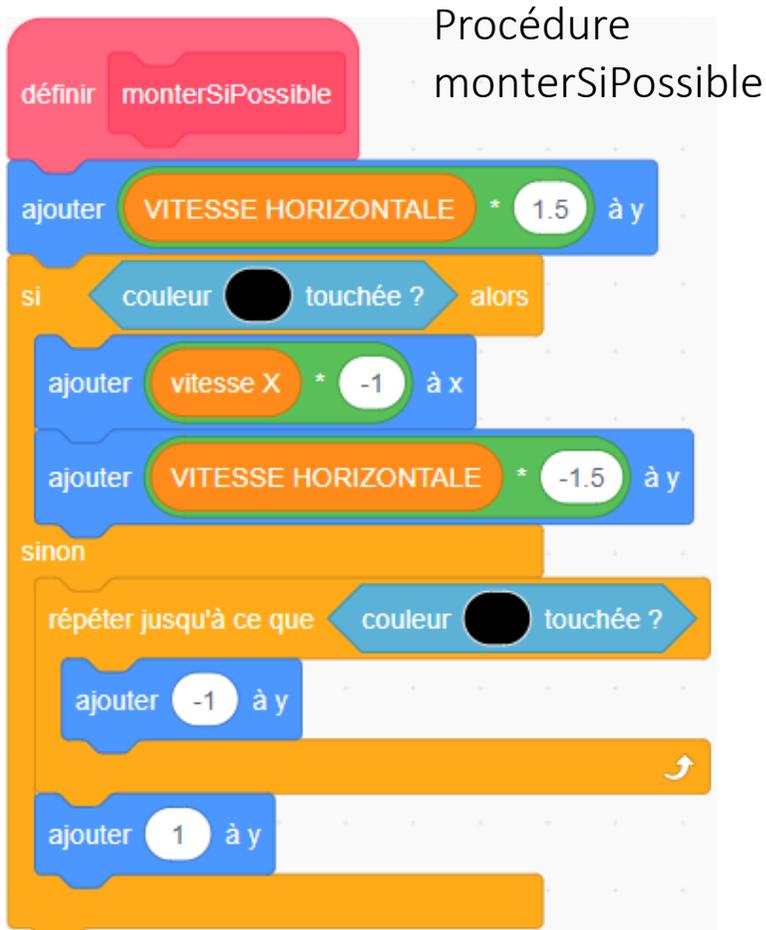


2° partie



# V3 : BLOCS DE LA VERSION 3

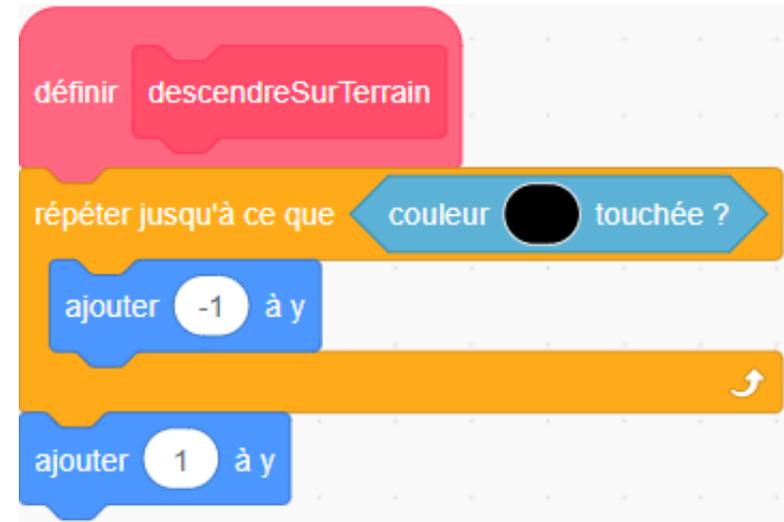
Procédure monterSiPossible



```
definer monterSiPossible
  ajouter VITESSE HORIZONTALE * 1.5 à y
  si couleur [noir] touchée ? alors
    ajouter vitesse X * -1 à x
    ajouter VITESSE HORIZONTALE * -1.5 à y
  sinon
    répéter jusqu'à ce que couleur [noir] touchée ?
      ajouter -1 à y
    ajouter 1 à y
```

The code defines a procedure named 'monterSiPossible'. It starts with a blue 'ajouter' block that multiplies 'VITESSE HORIZONTALE' by 1.5 and adds it to 'y'. This is followed by an orange 'si' block with a black circle in the 'couleur' field and 'touchée ?' in the 'alors' field. Inside the 'si' block, there are two blue 'ajouter' blocks: one that multiplies 'vitesse X' by -1 and adds it to 'x', and another that multiplies 'VITESSE HORIZONTALE' by -1.5 and adds it to 'y'. After the 'si' block, there is an orange 'sinon' block. Inside the 'sinon' block, there is an orange 'répéter jusqu'à ce que' block with a black circle in the 'couleur' field and 'touchée ?' in the 'ce que' field. Inside this loop, there is a blue 'ajouter' block that adds -1 to 'y'. After the loop, there is another blue 'ajouter' block that adds 1 to 'y'.

Procédure descendreSurTerrain

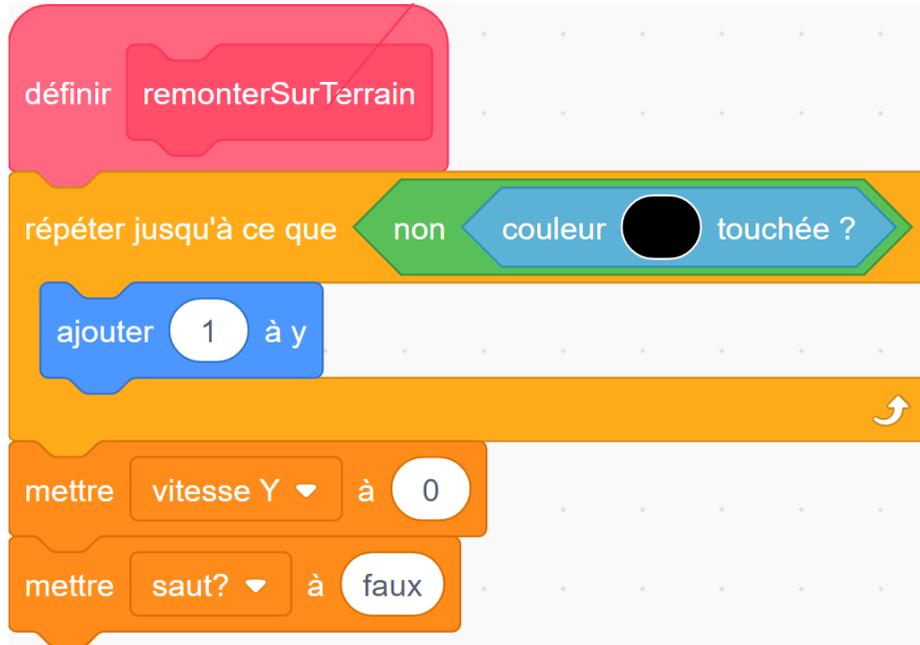


```
definer descendreSurTerrain
  répéter jusqu'à ce que couleur [noir] touchée ?
    ajouter -1 à y
  ajouter 1 à y
```

The code defines a procedure named 'descendreSurTerrain'. It starts with a blue 'ajouter' block that adds -1 to 'y'. This is followed by an orange 'répéter jusqu'à ce que' block with a black circle in the 'couleur' field and 'touchée ?' in the 'ce que' field. Inside this loop, there is a blue 'ajouter' block that adds -1 to 'y'. After the loop, there is another blue 'ajouter' block that adds 1 to 'y'.

# V3 : BLOCS DE LA VERSION 3

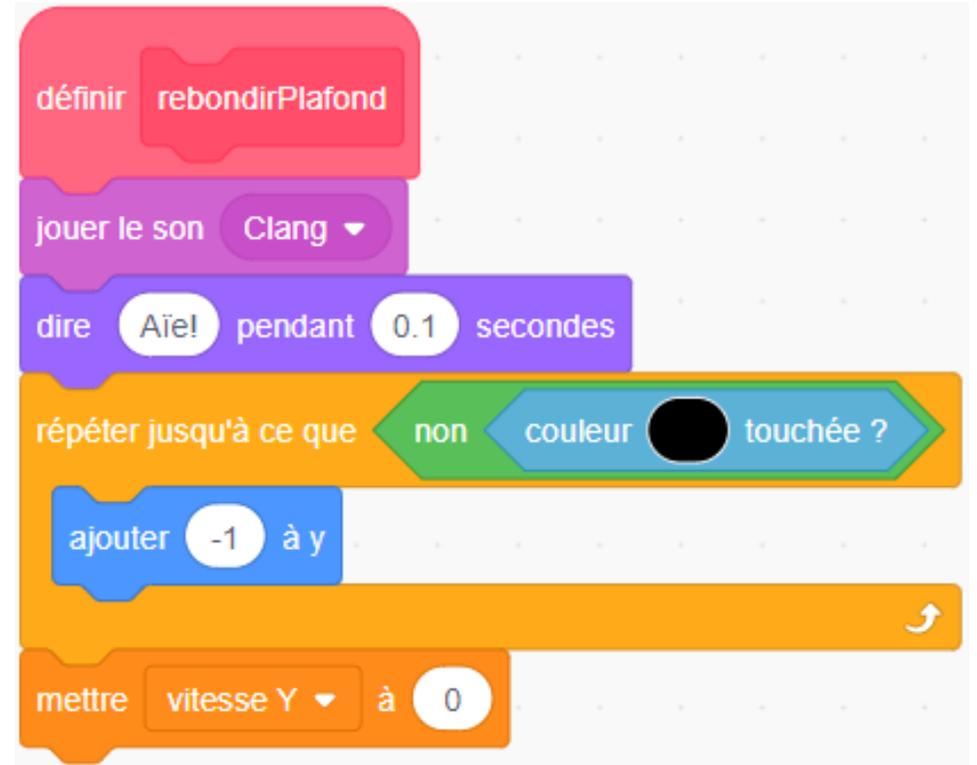
Procédure remonter



```
definer remonterSurTerrain
  répéter jusqu'à ce que non couleur noir touchée ?
    ajouter 1 à y
  mettre vitesse Y à 0
  mettre saut? à faux
```

The code defines a procedure named 'remonterSurTerrain'. It starts with a 'repeat until' loop where the condition is 'not black color touched?'. Inside the loop, the 'y' coordinate is increased by 1. After the loop, the 'Y speed' is set to 0 and the 'saut?' variable is set to 'faux'.

Procédure rebondir sur plafond



```
definer rebondirPlafond
  jouer le son Clang
  dire Aïe! pendant 0.1 secondes
  répéter jusqu'à ce que non couleur noir touchée ?
    ajouter -1 à y
  mettre vitesse Y à 0
```

The code defines a procedure named 'rebondirPlafond'. It starts by playing the 'Clang' sound and saying 'Aïe!' for 0.1 seconds. It then enters a 'repeat until' loop with the condition 'not black color touched?'. Inside the loop, the 'y' coordinate is decreased by 1. After the loop, the 'Y speed' is set to 0.