Bibliothèque Vaclav Havel

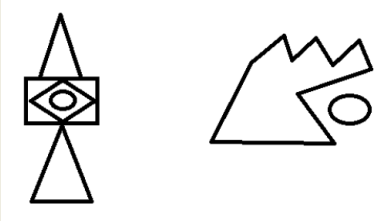
Le 16 et du 19 au 24 juillet 2016

Atelier de programmation 9 à 11 ans

Jour 1 : Hour of code (Minecraft)

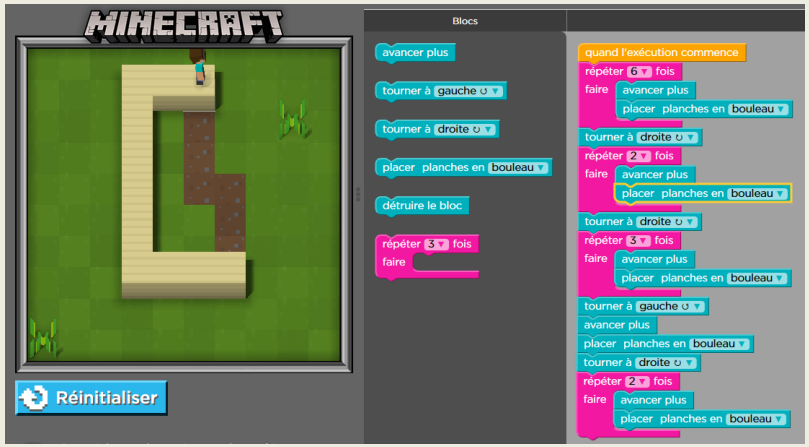
Objectif :

1. Qu'est-ce qu'un ordinateur ?
   Qu'est-ce que programmer ?
2. Introduction à la programmation : hour of code



1° partie : présentation des concepts (ordinateur et programmation)
atelier déconnecté : la programmation orale, sans ordinateur
http://voyageursducode.fr/ressources/fiches-activites/programmation-orale.html

2° partie : hour of code
https://studio.code.org/s/mc/stage/1/puzzle/1



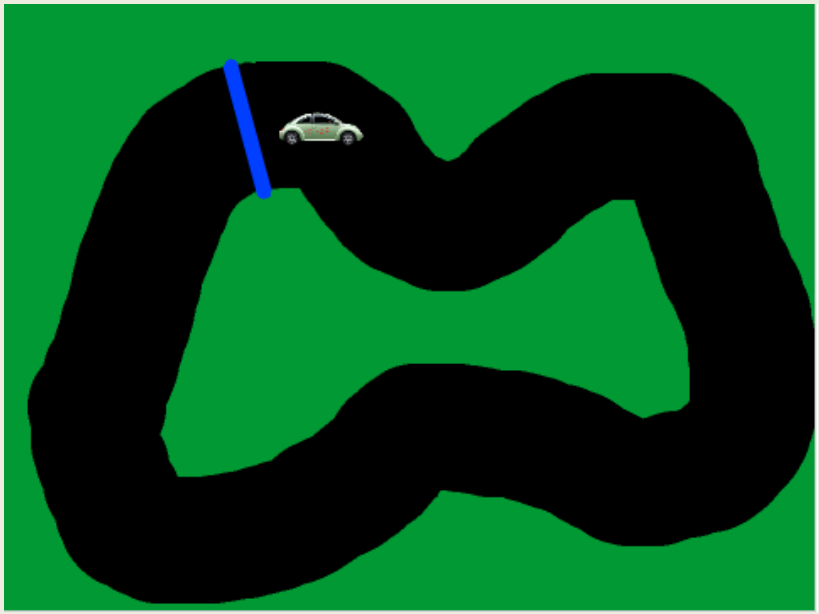
Jour 2 : l'algorithme pour « faire des pates »

Objectif :

1. Décomposer une recette courante en opérations élémentaires qui permettent d'arriver au résultat attendu,
2. Mettre au point l'algorithme en le déroulant pas à pas,
3. Discuter :

- des différents types de blocs utilisés
- des tests : booléens
  oui ou non, vrai ou faux
- des variables,
- des évènements, correspondant à des programmes multiples et simultanés
  (avec la possibilité de conflits entre programmes, ex : si on n'a qu'une plaque et plusieurs plats …)

Jour 2 : Créer un jeu de voiture – la base



Objectif :

1. Créer le circuit
2. Initialiser avec le drapeau vert
3. Faire avancer la voiture
4. Tester si elle sort ou si elle gagne

Ressource : https://youtu.be/bgswFXqYsNE

1° dessiner

- dessines le circuit et l'arrivée sur un fond vert

2° programmer

- Mets la voiture à la bonne taille et au bon endroit au démarrage (drapeau vert),
- Fais avancer la voiture en direction du pointeur de la souris lorsque la touche espace est enfoncée,
- Si la voiture touche le vert, renvoies au départ,
- Si la voiture touche le bleu, dis "*j'ai gagné*" et arrêtes le programme.

https://scratch.mit.edu/projects/117130566/

Jours 3 et 4 : Exercices

https://scratch.mit.edu/projects/117130566/

1. Contrôler la vitesse avec une variable 

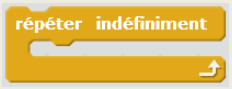
   https://scratch.mit.edu/projects/117130460/

2. Gérer les vies 

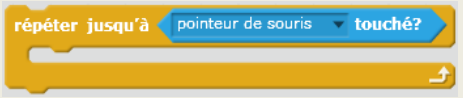
et la fin du jeu 

avec une variable et des tests

   https://scratch.mit.edu/projects/117130405/

3. Faire avancer la voiture en continu avec une boucle infinie 

et

   l'arrêter si elle touche le pointeur avec une boucle conditionnelle 

   https://scratch.mit.edu/projects/117130330/

4. Faire la même chose avec un test 

   https://scratch.mit.edu/projects/117130048/

5. Créer un obstacle intermittent sur la piste et ajouter des sons 

   https://scratch.mit.edu/projects/117081941/

6. Utiliser les procédures pour simplifier et améliorer le programme 

   https://scratch.mit.edu/projects/117082690/

7. Compter les tours et gagner au bout de 3 https://scratch.mit.edu/projects/117097449/
8. Généraliser les procédures https://scratch.mit.edu/projects/117102737/
9. Créer un 2° niveau : messages et configuration https://scratch.mit.edu/projects/117118979/

Exercice 1 sur les variables (vitesse)

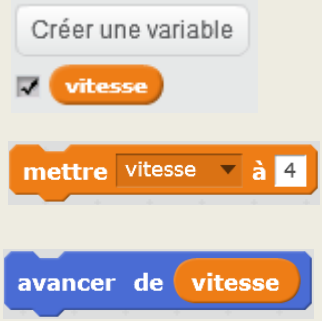
Objectif :

1. contrôler la vitesse avec une variable,
2. augmenter la vitesse pendant le jeu.

Attention : au mois de juillet 2016, la version française de Scratch comprend une erreur sur le libellé de la commande « remplacer … par … » qui doit se lire « ajouter à … la quantité … »
cf. aide de cette commande

1° partie :

1. définis la variable **vitesse**
2. Mets sa valeur à 4 au démarrage
3. Dans le bloc **avancer**, remplaces le chiffre par la variable
4. Essaies : la vitesse est affichée ?



2° partie : modifies la valeur de la variable

1. Ajoutes un évènement **sur flèche haut**
2. Ajoutes 1 à la variable **vitesse** avec un **opérateur** + et remplace la valeur précédente de la **vitesse**
3. Ajoutes un test pour limiter la vitesse à 10
4. Essaies et fais pareil pour réduire la vitesse avec la flèche bas

Exercice 1 sur les variables (vitesse)

Objectif :

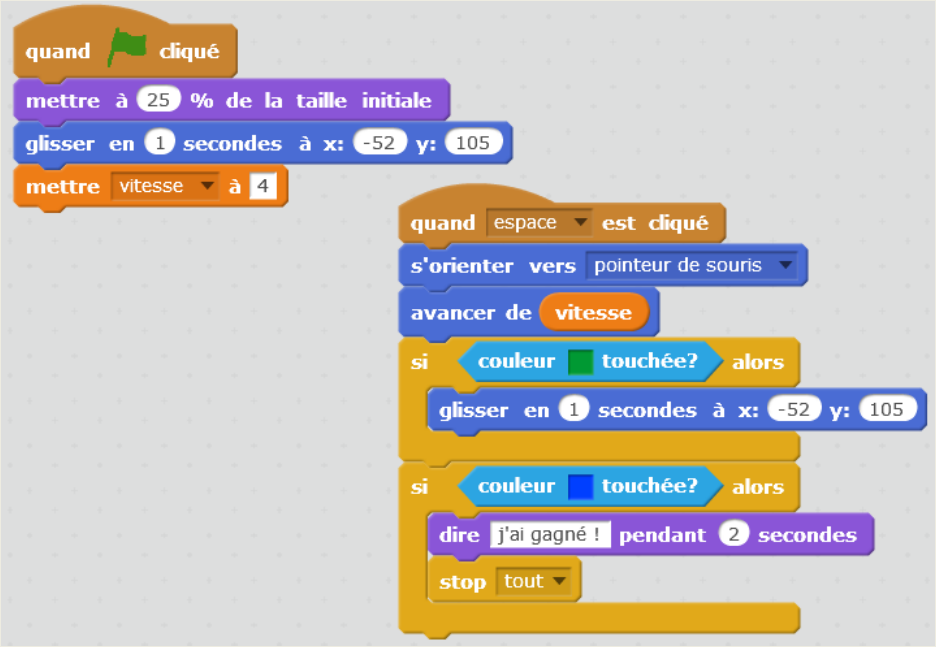
1. contrôler la vitesse avec une variable,
2. augmenter la vitesse pendant le jeu.

1° partie :

1. définis la variable **vitesse**
2. Mets sa valeur à 4 au démarrage
3. Dans le bloc **avancer**, remplaces le chiffre par la variable
4. Essaies : la vitesse est affichée ?

2° partie : modifies la valeur de la variable

1. Ajoutes un évènement :
   clique sur flèche haut
2. Ajoutes 1 à la variable **vitesse**
   avec un **opérateur** +
   et remplace la valeur précédente de la **vitesse**
3. Essaies et fais pareil pour réduire la vitesse avec la flèche bas



Exercice 1 sur les variables (vitesse)

Objectif :

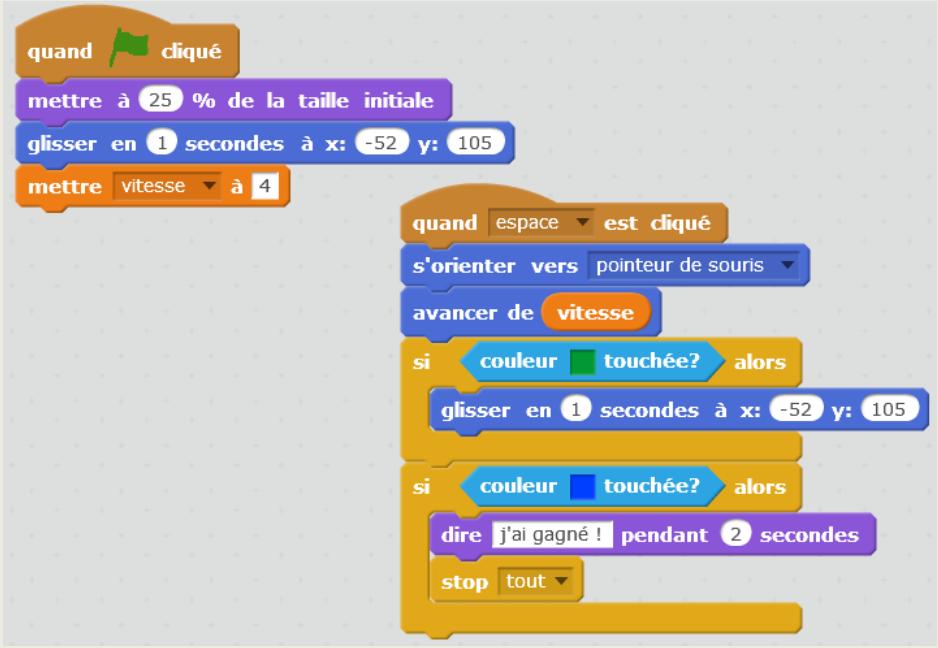
1. contrôler la vitesse avec une variable,
2. augmenter la vitesse pendant le jeu.

1° partie :

1. définis la variable **vitesse**
2. Mets sa valeur à 4 au démarrage
3. Dans le bloc **avancer**, remplaces le chiffre par la variable
4. Essaies : la vitesse est affichée ?

2° partie : modifies la valeur de la variable

1. Ajoutes un évènement : **clique sur flèche haut**
2. Ajoutes 1 à la variable **vitesse** avec un **opérateur** + et remplace la valeur précédente de la **vitesse**
3. Essaies et fais pareil pour réduire la vitesse avec la flèche bas



Exercice 2 sur les variables et tests (vies)

Objectif :

1. Définir un nombre de vies au départ,
2. Retirer une vie à chaque fois que la voiture redémarre (sortie de route),
3. Afficher « Game over » quand il n'y a plus de vies

Réalisation :

1. définis la variable **vies** et mets sa valeur à 3 au démarrage
2. Ajoutes un costume « Game Over » à la scène
3. quand la voiture touche le vert, retires une vie,
   et si le nombre de vies est égal à 0, affiches « Game Over » et stop tout

Exercice 3 : sur les boucles et boucles conditionnelles

Objectif :

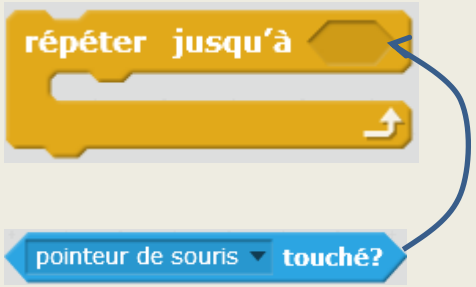
1. La voiture doit continuer à avancer même si on n'appuie plus sur la touche espace.
2. Elle doit s'arrêter si elle touche le pointeur de la souris

1° partie :

Pour que la voiture avance en permanence, mets la commande avancer dans une boucle infinie.
Penses à ajouter une commande « stop ce script » pour que la voiture s'arrête si elle retourne au point de départ.

2° partie :

Pour que la voiture s'arrête, lorsque la voiture touche le pointeur de la souris, utilises une condition pour sortir de la boucle lorsque le capteur détecte que la voiture touche le pointeur souris.



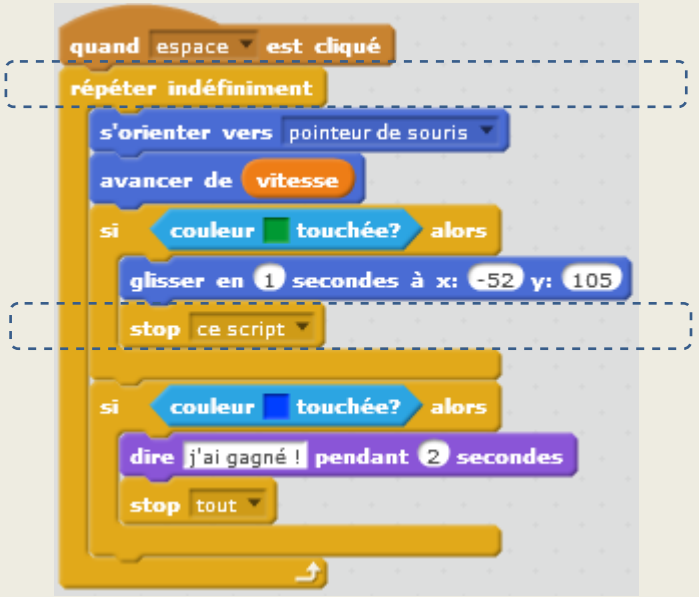
Exercice 3 : sur les boucles et boucles conditionnelles

Objectif :

1. La voiture doit continuer à avancer même si on n'appuie plus sur la touche espace.
2. Elle doit s'arrêter si elle touche le pointeur de la souris

1° partie :

Pour que la voiture avance en permanence, mets la commande avancer dans une boucle infinie.
Penses à ajouter une commande « stop ce script » pour que la voiture s'arrête si elle retourne au point de départ.



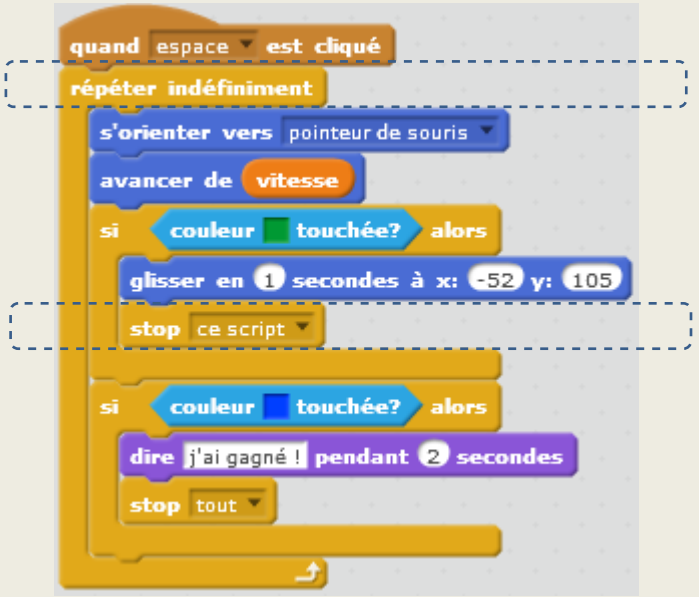
Exercice 3 sur les boucles et boucles conditionnelles

Objectif :

1. La voiture doit continuer à avancer même si on n'appuie plus sur la touche espace.
2. Elle doit s'arrêter si elle touche le pointeur de la souris

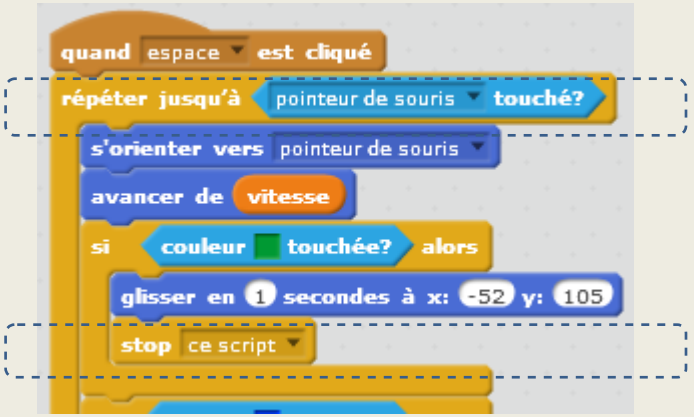
1° partie :

Pour que la voiture avance en permanence, mets la commande avancer dans une boucle infinie.
Penses à ajouter une commande « stop ce script » pour que la voiture s'arrête si elle retourne au point de départ.



2° partie :

Pour que la voiture s'arrête, lorsque la voiture touche le pointeur de la souris, utilises une condition pour sortir de la boucle lorsque le capteur détecte que la voiture touche le pointeur souris.



Exercice 4 sur branchement si/sinon et opérateurs logiques

Objectif :

1. Pour arrêter la voiture lorsqu'elle touche le pointeur de la souris, remplacer la solution précédente par un test avant d'avancer

1° version :

Au lieu de mettre la condition dans la boucle, testes avec un bloc si/sinon :

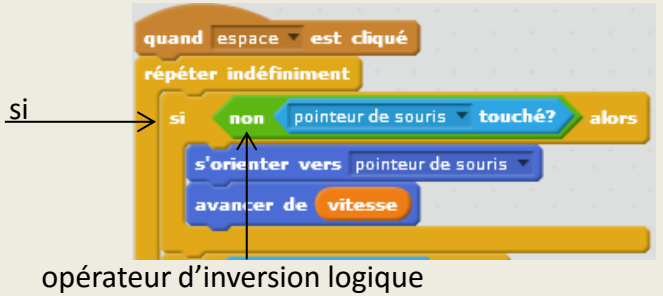
- Si le pointeur est touché : on ne fait rien
- Sinon : orientation vers le curseur et on avance

2° version :

- Comme la branche "Si" ne sert à rien on peut l'éviter, mais il faut inverser la valeur du test avec l'opérateur 

, ça inverse le résultat

Nota : l'exercice permet d'illustrer qu'il y a toujours deux branches possibles dans un test, même si une seule est utilisée.



Exercice 5 : Obstacle et son (exercice défini en séance)

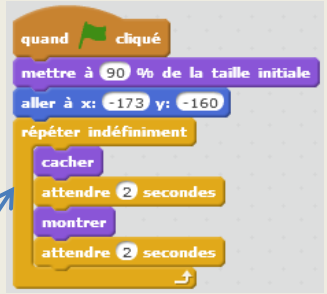
Objectif :

- Mettre un obstacle au milieu de la route (il doit apparaitre de façon intermittente, si la voiture le touche elle perd une vie et retourne au départ).
- Jouer un son quand la voiture sort de la route et un autre quand elle touche l'obstacle

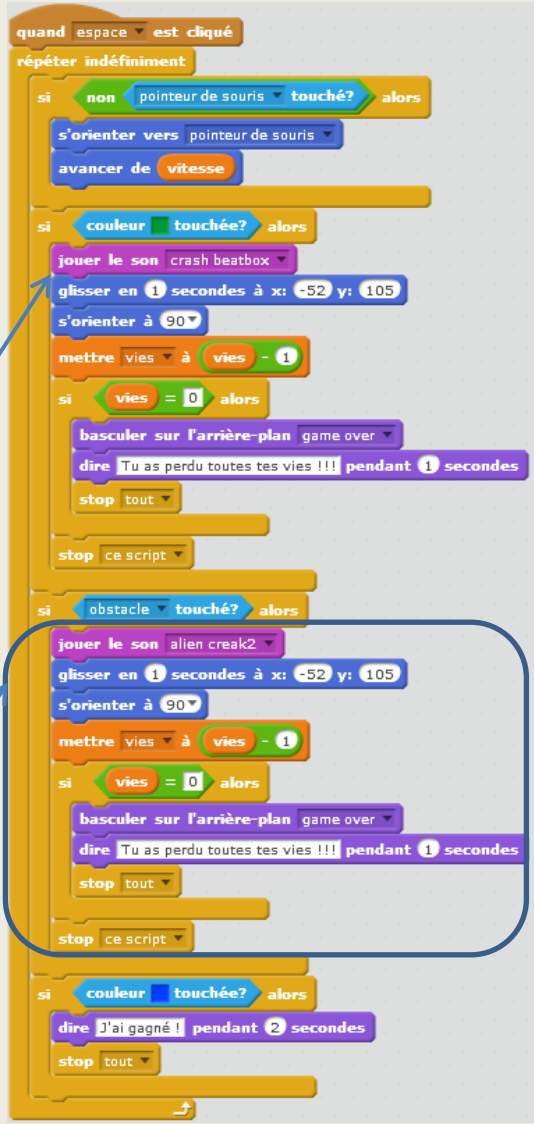
Obstacle au milieu de la route :

1. créer un lutin « obstacle »
   - le mettre à la taille et au bon endroit au départ
   - le faire apparaitre et disparaitre dans une boucle infinie, avec un temps d'attente
2. Dans le programme de la voiture,
   - ajouter un son si la voiture sort de la route
   - si la voiture touche l'obstacle, jouer un son et faire pareil que pour la sortie de route : retirer une vie et renvoyer la voiture au départ.

Script de l'obstacle



Script de la voiture



Exercice 6 : procédures et abstraction

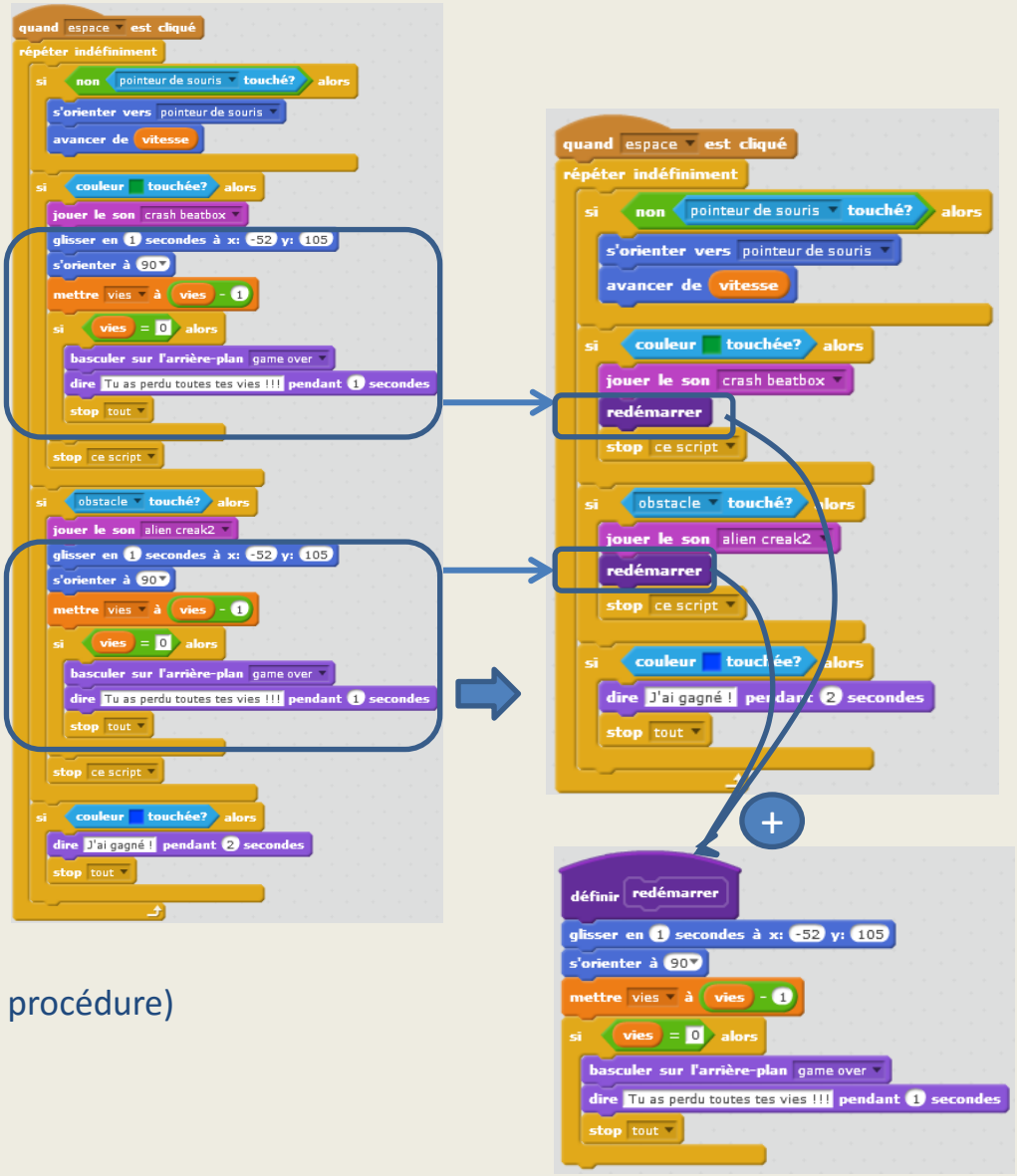
Objectif :

- Montrer que l'on peut simplifier
  écrire une seule fois les mêmes opérations.
- Expliquer comme abstraction

Réalisation :

- On remplace les deux gros paquets de blocs identiques, de redémarrage par un nouveau bloc `redémarrer` qui fait la même chose.
  C'est un bloc « **procédure** »

- Attention, il ne faut pas mettre `stop ce script` dans la procédure
  (car une procédure peut être appelée depuis plusieurs scripts.
  `stop ce script` ne marche pas dans une procédure)



Exercice 7 tests logiques dynamiques (compter les tours)

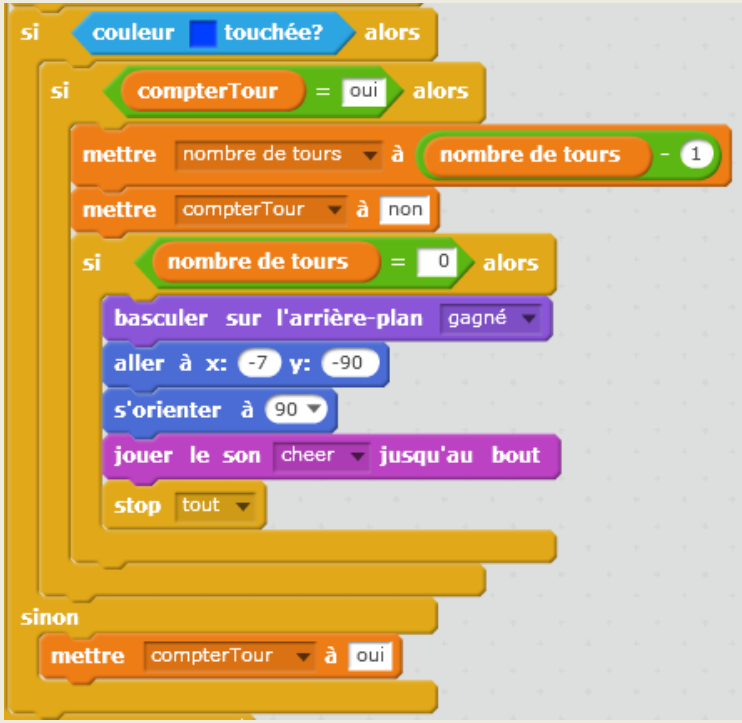
Objectif :

- Compter les tours 3,2,1, et dire « on a gagné » quand le nombre vaut 0

Nature de la difficulté :

- La voiture touche la ligne d'arrivée en continu (plusieurs fois), il ne faut compter qu'un seul tour,
- C'est un problème de logique dynamique, l'action à prendre dépend de l'action précédente

➢ Sujet à éviter de traiter en semaine 1



Réalisation :

- Créer la variable `nombre de tours` initialisée à 3 et la variable `compterTour` initialisée à **oui**
- Dans la boucle infinie
  - Si la voiture touche la ligne d'arrivée, alors
    - Si `compterTour` = oui,
      mettre `compterTour` à non
      retirer 1 à `nombre de tours`
      et si `nombre de tours` = 0 passer à l'écran gagné et applaudir
  - Sinon mettre `compterTour` à oui (on a dépassé la ligne, la prochaine fois on pourra décompter un tour)

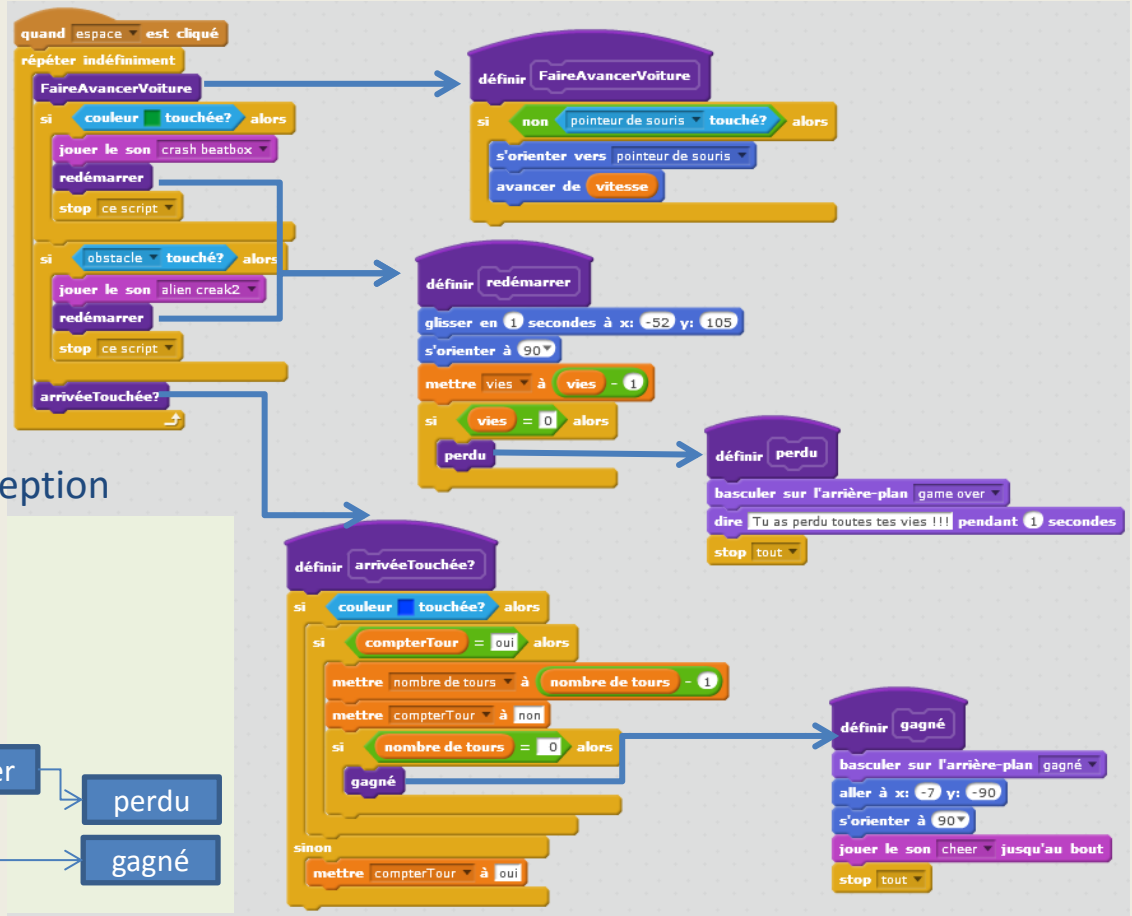
Exercice 8 : abstraction / généralisation des procédures

Objectif :

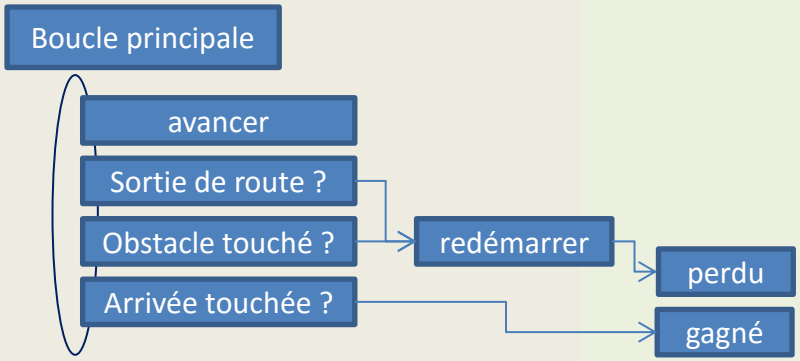
1. Dégager les principales fonctions de la voiture et les mettre en procédures,
2. Faire apparaitre la structure générale du programme sous forme de pseudo-code,
3. Discuter des notions d'abstraction et de conception.

Extraction : des procédures

1. "redémarrer" (existe déjà)
2. "avancer voiture"
3. "arrivée touchée ?"
4. "gagné"
5. "perdu«



Discussion : abstraction et conception



Exercice 9 algorithme multi-niveaux (1/3)

Objectif :

Réaliser un 2° niveau de jeu après avoir gagné le premier

Difficulté :

1. le programme est le même pour tous les niveaux. TOUTEs les actions qui dépendent du niveau doivent être paramétrées (de préférence à l'initialisation),
2. Il faut distinguer ce qui doit être fait au début du jeu et au début de chaque niveau
3. La plupart des scripts, dont ceux en boucle infinie (ex : l'obstacle intermittent) doivent être interrompus et réinitialisés

Solution adoptée :

Pour chaque niveau on va définir :

- les coordonnées et l'orientation de démarrage de la voiture par des variables,
- les coordonnées et la taille de l'obstacle.

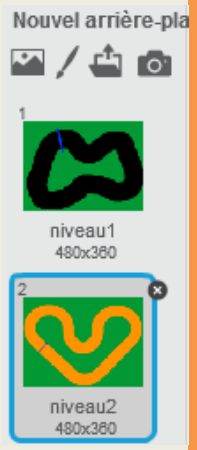
La préparation de chaque niveau se fait à réception du message "préparerNiveau"

Le démarrage se fait à réception du message "démarrerNiveau"

Ces messages sont envoyés

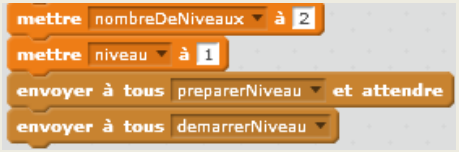
1. au départ (drapeau vert) pour le niveau 1,
2. et dans la procédure appelée quand on gagne un niveau.

Exercice 9 : multi-niveau (2/3)



Réalisation :

1. Crées un arrière-plan "niveau2" (avec exactement la même couleur d'herbe et d'arrivée)
2. Crées la variable **niveau** et une variable **nombreDeNiveaux**
   au drapeau vert initialises ces variables
   et envoies un message de préparation
   puis de démarrage du niveau



3. Crées les variables de démarrage de la voiture **xDépart** , **yDépart** et **orientationDépart**
   (elles vont changer à chaque niveau)
   et dans la procédure redémarrer, reviens aux
   coordonnées et l'orientation définies par ces variables

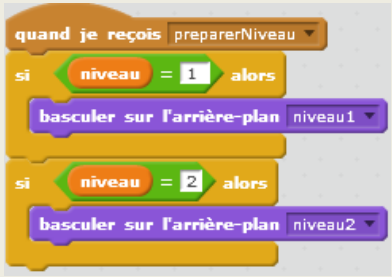
4. Dans la procédure "**gagné**" appelée à la fin de chaque niveau,
   Si le niveau est inférieur au nombre de niveaux,
   ajouter 1 au niveau et envoyer les mêmes messages
   Sinon, gagner comme avant

5. Ensuite, il faut dire ce que doit faire chaque lutin quand
   il reçoit les messages **préparerNiveau**
   et **démarrerNiveau**

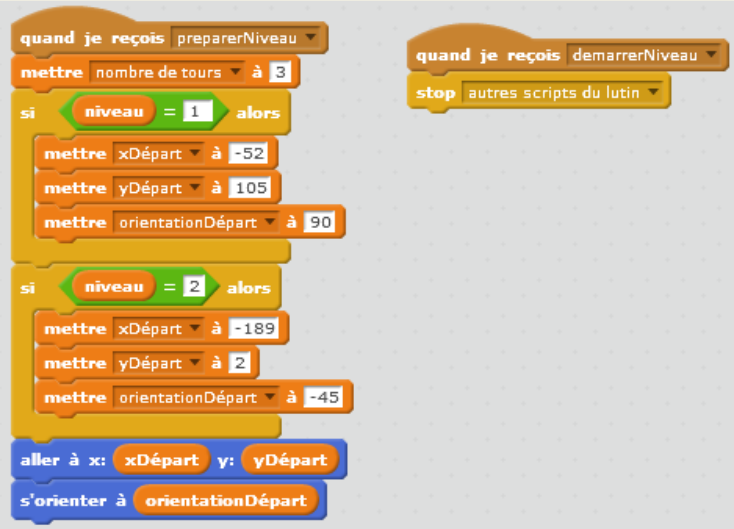
Exercice 9 : multi-niveau (3/3)

Scripts à réception des messages **préparerNiveau** et **démarrerNiveau**

1. Réponses aux messages pour la scène
(on peut supprimer l'évènement drapeau vert)



2. Réponses aux messages pour la voiture

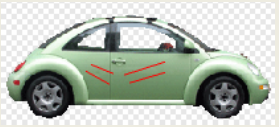


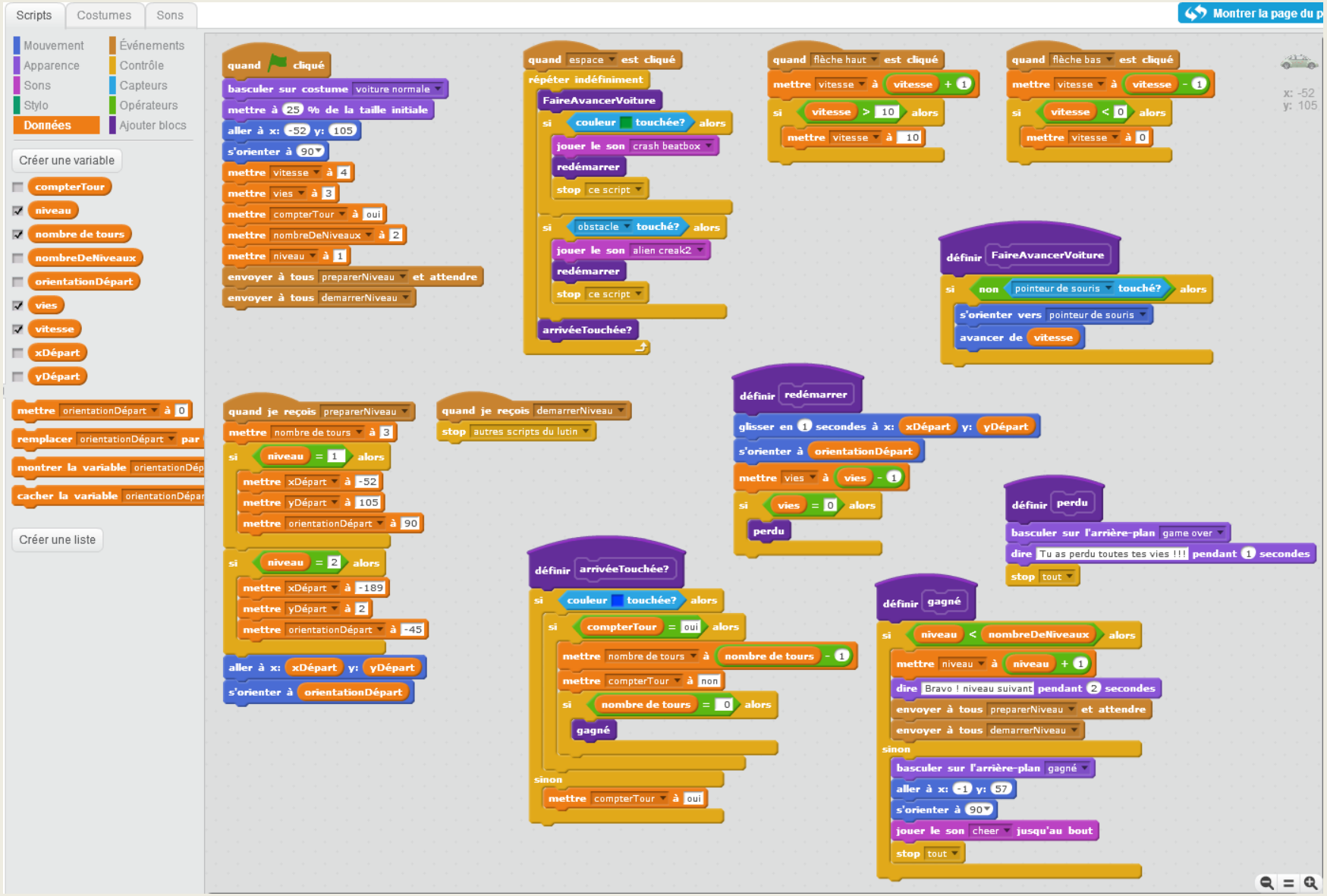
3. Réponses aux messages pour l'obstacle
(on peut supprimer l'évènement drapeau vert)

Nota :

il est indispensable de séparer la boucle infinie de la réception du message de préparation. Sinon, le message n'est jamais reçu, car la boucle est en fonctionnement continu

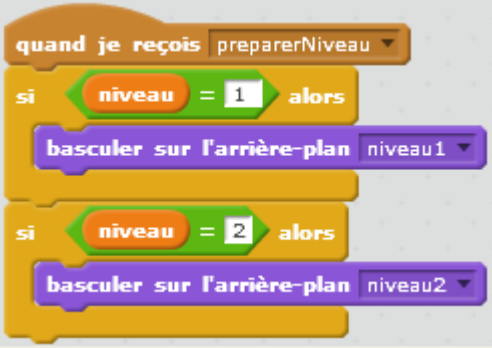
Résultat (scripts de la voiture)





Résultat (scripts de la scène et de l'obstacle)

Scripts de la scène



Arrière-Plans

Scripts de l'obstacle

Résultats intermédiaires des enfants

Les planches précédentes correspondent à la réalisation de toutes les idées émises par tous les enfants au cours de l'atelier. C'est plutôt un guide pour les animateurs.

Pendant l'atelier, les enfants ont commencé par un tronc commun correspondant

- **au dessin du circuit,**
- **à l'initialisation de la voiture et son déplacement,**
- **au redémarrage en cas de sortie de route,**
- **au comptage des vies.**

Puis chaque enfant a développé son jeu dans la direction qu'il souhaitait avec

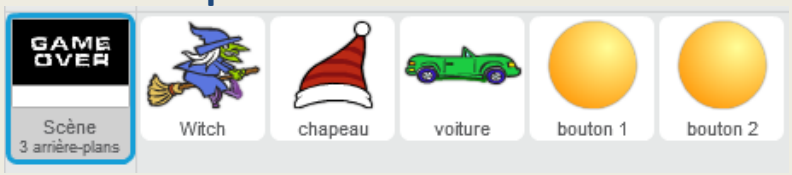
- **des obstacles de différents types,**
- **le décompte des tours,**
- **la gestion des cas où on gagne ou perd,**
- **la création d'un 2° niveau avec des difficultés différentes.**

Des résultats obtenus entre le début et la fin du 4° jour de l'atelier sont illustrés ci-après.

BRAVO ! Ils sont vraiment très forts

Oussem (point le 22/07 à 17:36 fin du 4° jour)

5 lutins et 3 arrière-plans :



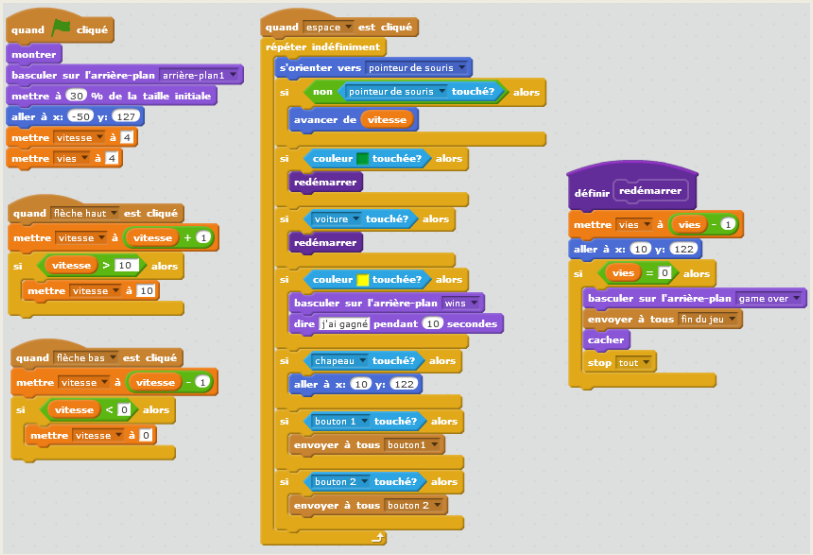
2 obstacles auxquels la sorcière doit échapper

Le chapeau intermittent
La voiture qui va vers les boutons
(déclenchement quand la sorcière passe sur ces boutons)

Programme

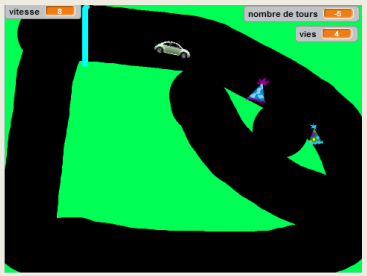
41 blocs, 4 scripts, 1procédure pour la sorcière
11 blocs, 2 scripts pour l'obstacle chapeau
11 blocs, 4 scripts pour l'obstacle voiture

1 procédure, 2 boucles, 10 tests, 2 variables
7 tests capteurs, 3 tests mathématiques (>,<)
1 opérateur logique (non)

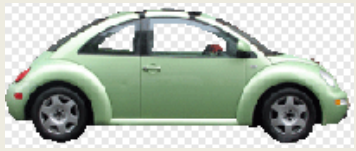


Mode de développement

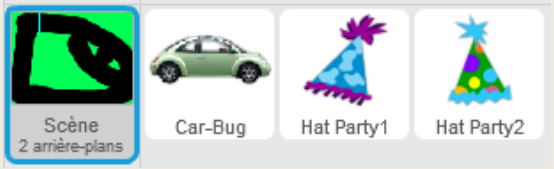
Guidé pour la sorcière, sa vitesse et ses vies
Créatif de groupe pour le chapeau
Créatif individuel pour la voiture et les 2 boutons



Nilan (point le 22/07 à 16:14 – début 4° jour)



3 lutins et 2 arrière-plans :



2 obstacles auxquels la voiture doit échapper

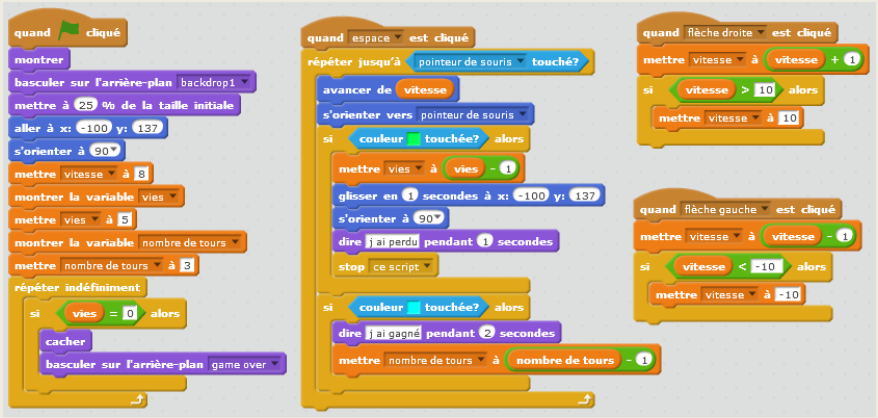
2 chapeaux intermittents (codage en cours de séance)
Codage du nombre de tours : début

Programme

36 blocs, 4 scripts pour la voiture
6 blocs, 1 script pour chaque obstacle
(codage en cours de séance)

4 boucles, 5 tests, 3 variables
2 tests capteurs, 3 tests mathématiques (>,<)

Scripts de la voiture



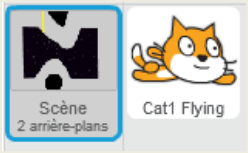
Mode de développement

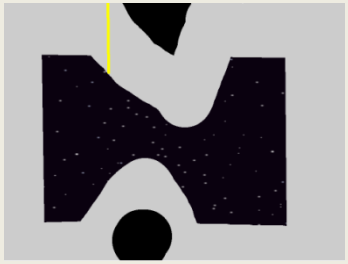
Guidé pour la voiture, sa vitesse et ses vies
Créatif de groupe pour le chapeau
Créatif individuel pour le nombre de tours

Scripts des obstacles
(en cours de codage)

O'Neil (point le 22 ou 23/07)

1 lutins et 2 arrière-plans (pour 2 niveaux) :





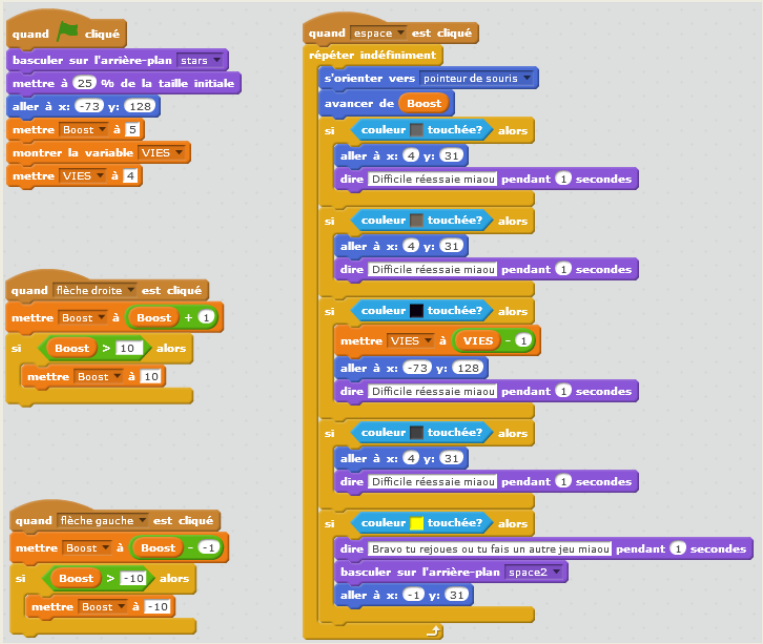
Les obstacles auxquels le chat volant doit échapper

Ces obstacles sont définis par couleur
(plutôt que comme des lutins)
2 niveaux de jeu avec 2 arrière-plans et
2 conditions initiales

Programme

36 blocs, 4 scripts pour le chat volant

1 boucle, 7 tests, 2 variables
5 tests capteurs, 2 tests mathématiques (>,<)



Mode de développement

Guidé pour le chat, sa vitesse et ses vies
Créatif individuel pour le design et le 2° niveau

(la difficulté réside ici dans le multi-niveau)