Un exemple projet présentant le moteur pacman

Le projet est développé sur 3 séances ou plus. Lors de chaque séance, les élèves travaillent sur des modules particuliers, et les adaptent suivant le type de jeu qu'il souhaite développer.

Il y a un **pré-requis** : connaître les bases de Scratch comme le système de coordonnées et le déplacement d'un lutin par des instructions de type logo (utilisation des instructions : Avancer de, S'orienter à , aller à , ajouter 10 à x).

Certaines compétences comme "créer un nouveau lutin", "comprendre les codes attachés aux différents lutins" , "boucle répéter indéfiniment et condition de type capteur", même si elles ne sont pas indispensables, constituent des "plus" évidents.

Voici le jeu qui nous sert de support pour expliquer le moteur pacman.

Ecran de départ

Ecran final

Le jeu

Le papillon de gauche, le lutin "**mobile**", doit libérer sa copine, le lutin "**cible**" et doit également récupérer tous les ballons.

A chaque fois qu'il récupère un ballon, sa vitesse augmente.

Il y a trois types d'obstacles et 2 sanctions différentes :

- la chauve-souris qui poursuit le papillon. Perte de points de points en cas de collision.
- les chauve-souris marrons qui se déplacent aléatoirement. Perte de points si contact.
- les fusées qui se déplacent verticalement selon le circuit bas-haut-bas. Retour à la case départ si contact et perte de point.

Le score commence à 120 (secondes) et baisse avec le temps et les collisions.

Le score est basé sur le temps de jeu (la variable chronomètre de Scratch) et est affecté par les malus dûs aux éventuelles collisions.

Pour avoir essayer le jeu, lancer le fichier **https://scratch.mit.edu/projects/108611042/**

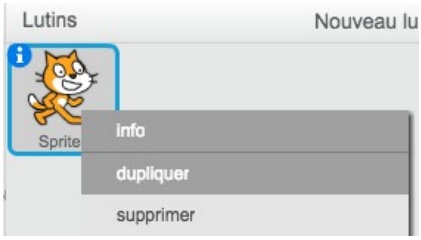
Dans cette dernière version, j'ai ajouté la technique de clonage dans le déplacement du lutin "mobile". Elle donne un effet lors des déplacements : les dernières positions du lutin mobile persistent un cours instant et sont un peu transparentes. Mais cette fonctionnalité me semble trop délicate pour le collège. Elle est d'ailleurs complètement facultative.

MODULE 1

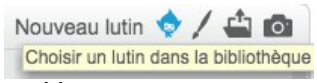
Création du
- lutin dynamique
- du lutin cible (optionnel)

➔ *Supprime le lutin chat par défaut et crée deux lutins : un lutin "**mobile**" et un lutin "**cible**" qui sera immobile sur la scène et que le lutin "mobile" devra libérer.*

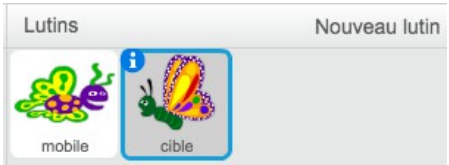
Pour supprimer le lutin chat , bouton droit sur le lutin chat et "supprimer".



Pour créer un lutin :



Une fois les deux lutins créés, on peut voir les deux lutins :



(choisis les lutins que tu veux mais nomme-les "mobile" et "cible")

➔ Pour changer le nom d'un lutin, double clique dessus puis :

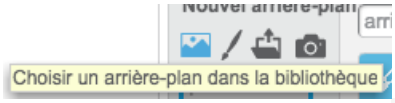
➔ Modifie également le style de rotation :

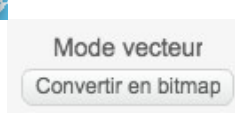
MODULE 2

Création de
- l'arrière-plan
- des éventuels obstacles fixes

➔ *Clique sur l'arrière-plan* 

et sur l'onglet "Arrières-plans" : 

➔ *Commence par choisir une image d'arrière-plan* 

➔ *Clique sur le bouton Vectoriser (en bas à droite) afin d'être en "Mode vecteur"* 

Ce bouton permet de créer des formes géométriques.

Nous allons construire des rectangles représentant les murs de la scène.

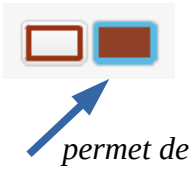
➔ *Crée un mur vertical et un mur horizontal à l'aide de l'outil rectangle.*
Ensuite il suffira de copier-coller ces murs pour créer tous les autres murs.

Attention, ne choisis pas des murs de couleur noire.

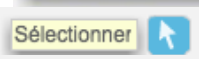
Pour créer un rectangle, utilise le bouton : 

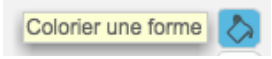
Voici quelques copies d'écran pour t'aider :

deux murs... 

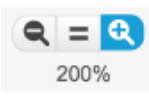

permet de construire un rectangle plein

Pour recopier un rectangle, utilise le bouton "dupliquer" : 

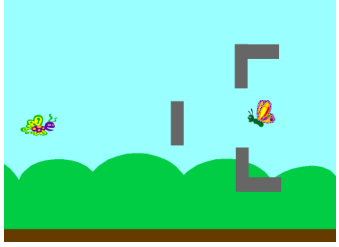
Pour modifier un rectangle, clique d'abord sur le bouton : 

Pour remplir un rectangle, clique sur le bouton : 

Pense à annuler si nécessaire : 

Les boutons zoom peuvent être utiles : 

Voici un **exemple possible** avec les deux lutins sur la scène :



<u>➔ Crée ton propre tableau !</u>

Si nécessaire, **change la taille des lutins** à l'aide de : 

MODULE 3

Initialisation du jeu
- On remet tous les lutins à leur place initiale
- Réinitialisation des variables

Pendant l'exécution du jeu, des objets se déplaceront, en particulier le lutin "mobile".

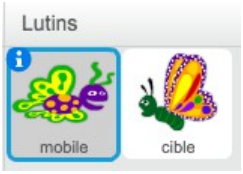
➔ A chaque lancement du jeu, **il faut donc positionner le lutin "mobile" à sa place initiale**.

Utilise les briques suivantes pour cela :

Attention, pour le remettre à sa place initiale, il faut placer le code sur le lutin "mobile".

Il faut toujours veiller à ce que ce soit le bon lutin qui est sélectionné dans la fenêtre des lutins quand on ajoute du code.

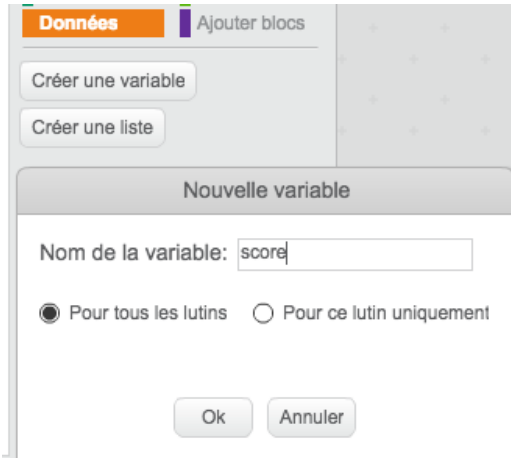
Ici par exemple , c'est le lutin "mobile" qui est sélectionné :



➔ Le score sera stocké dans une variable "***score***".

Crée une variable score et initialise cette variable à 0 pour l'instant.

Voici deux copies d'écran pour t'aider :



*Cette **instruction** doit être exécutée une fois à chaque lancement du programme.*
Insère là donc au début du programme.

Tu remarques le score est affiché à l'écran. Cela nous convient mais on peut améliorer la présentation.

➔ Clique sur score (celui qui est sur la scène) avec le bouton droit et sélectionne "grande lecture" :

➔ Puis modifie l'arrière-plan ! 

Pour créer un texte, utilise le bouton 

Quand un texte est créé, pour le modifier à nouveau, tu dois utiliser le bouton 

MODULE 4

- Gestionnaire clavier pour les déplacements du lutin dynamique
- Gestion optionnelle des costumes
- Gestion des murs

Pour comprendre comment déplacer ton lutin "***mobile***" à l'aide des touches fléchées, examine le code suivant :

➔ ***Programme toi-même les autres déplacements.***

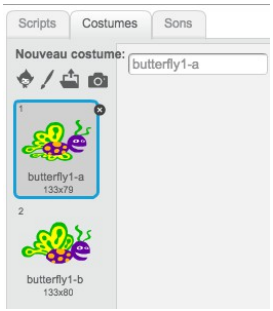
N'oublie pas de tester ton programme !

Que se passe-t-il si on change le style de la rotation :

Remarque sur les costumes

Le pacman que j'ai choisi possède deux costumes :

Quand le lutin change de costume rapidement,
on a l'illusion que ses ailes bougent.



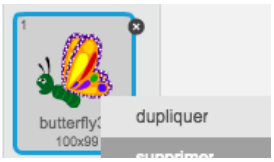
Dans la catégorie Apparence 

il y a une instruction permettant de changer le costume d'un lutin :

 Où placer cette instruction pour donner l'illusion du vol ?

Pour aller plus loin

Mon second lutin ne possède qu'un seul costume.

J'ai donc créé un second costume en dupliquant le premier 

puis j'ai modifié ce second costume en diminuant légèrement sa taille.

Ce second lutin est immobile, il faut donc lui demander de costume régulièrement et de lui-même.

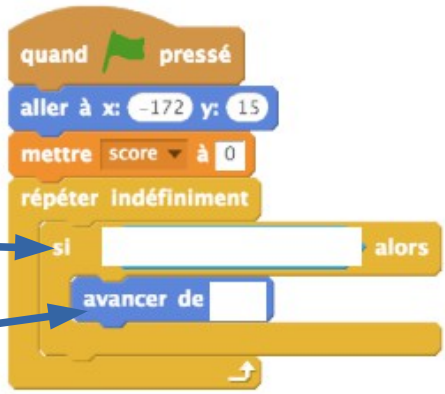
➔ Essaie de le faire en utilisant les briques suivantes :

➔ *A ton avis, que fait l'instruction de contrôle "répéter indéfiniment" ?*

➔ *Que se passe-t-il si on n'utilise pas l'instruction "attendre 0.3 secondes" ?*

Teste ton programme.

Il y un souci, ton lutin "mobile " traverse les murs !



➔ Complète le bloc suivant dans le lutin "mobile" :

Tu dois trouver les deux expressions manquantes.

La première teste si un mur est touché :

Pour cela, il faut tester si la couleur du mur est touchée

(cherche dans la catégorie Capteurs)

La seconde est un nombre.

Que faut-il faire si après un déplacement on a touché un mur ?

Comprends bien que le programme va tester à chaque instant si ton lutin touche la couleur d'un mur.

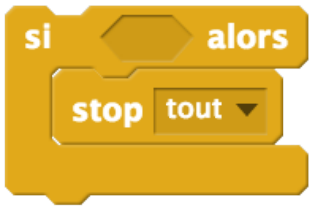
Si jamais cela arrive alors l'instruction avancer de... sera exécutée.

MODULE 5

Gain de la partie :
- cible touchée ?
- cibles optionnelles récupérées ?
  Nécessite une variable supplémentaire.

A notre stade, la partie est gagnée quand le lutin "***cible***" est touchée par le lutin "***mobile***".

Observe attentivement le code
suivant déposé sur le lutin "mobile"
dans la **boucle *répéter indéfiniment :***



➔ Cherche **l'expression** manquante (à insérer dans le SI) dans la catégorie ***Capteurs***

A quoi sert l'instruction :  ?

➔ Ajoute un message comme ***Gagné !*** par exemple.
Pour cela, utilise l'instruction  de la catégorie ***Apparence***.

Teste ton programme !

MODULE 6

Perte de la partie
- temps écoulé
- nombre maximal de collisions atteint (les vies)

MODULE 7

La variable score
- peut dépendre du chronomètre
- peut dépendre des collisions
- peut nécessiter une variable malus augmentant avec les obstacles touchées.

Dans notre projet, nous traiterons ces deux modules en même temps car le score dépend du temps écoulé.

Mais cela peut être différent dans ton jeu !

➔ Créons une variable ***score*** initialisée à 120 (secondes) et diminuant avec le temps.

De cette manière, le score sera d'autant plus élevé que nous libérerons vite notre compagnon.

Si le temps écoulé dépasse 120 secondes, alors on décide que la partie est perdue.

➔ La variable score a déjà été créée. On **l'initialise** à 120.

Dans Scratch, dès qu'un programme est lancé, une variable ***chronomètre*** augmente de 1 toutes les secondes.

➔ Dans ton programme, ajoute le code suivant dans la boucle répéter indéfiniment :

➔ *Teste le programme.*

➔ *Quel est le problème ?*

Essaie de le régler en utilisant l'expression : 

Cette boucle 

accueille déjà de nombreuses instructions.

➔ Quand la partie est-elle perdue ?

➔ Que faut-il ajouter dans la boucle, à l'aide d'un **SI...ALORS....**, pour stopper le programme quand la partie est perdue ?

Utilise l'expression 

de la catégorie ***Opérateurs*** et 

Tu peux ajouter bien-sûr un message ***Perdu !***

MODULE 8

Clefs à récupérer

Dans ton jeu, tu voudras peut-être que le lutin "mobile" récupère un certain nombre de clefs.

Dans mon jeu, ces clefs sont des lutins "ballons".

Pour gagner la partie, le lutin mobile doit libérer son compagnon **ET** attraper tous les ballons.

Quand mon lutin "mobile" touche un ballon (une clef), le ballon disparaît.

➔ Commence par créer un lutin représentant une de ces clefs.

➔ Dans le code de ce lutin, dépose le code suivant :

➔ Que faut-il insérer dans le SI ?

Cherche l'instruction dans la catégorie ***Capteurs***.

➔ Le lutin ballon a-t-il vraiment disparu ?

Supposons qu'il y ait 3 ballons sur la scène. Si le lutin touche les trois ballons, alors il a récupéré tous les ballons. Puis il libère son compagnon. **Mais comment savoir que tous les ballons sont récupérés ?**

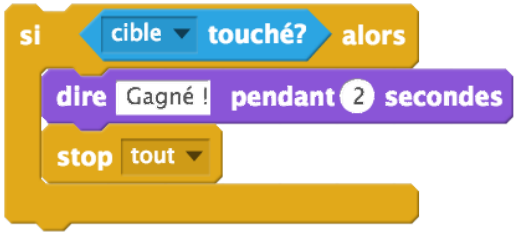
On a besoin de le savoir pour décider que la partie est gagnée.

➔ Une méthode possible consiste à créer une variable ***clefsrecuperees*** qui est initialisée à 0 au lancement du programme. Ensuite à chaque fois que le lutin "mobile" touchera un ballon, on augmente la variable clefsrecuperee de 1 :

Où faut-il déposer cette instruction dans le script de la clef ?

➔ Une fois le code de la clef complet, tu peux **dupliquer** le lutin clef.

Pour l'instant dans le script du lutin "mobile" on a le bloc suivant :



Pour gagner maintenant il faut que le lutin "cible" soit touché

ET que *clefsrecuperee* soit égal à 3 (s'il y en a trois).

C'est pour cela qu'il faut déposer dans le SI une expression comme

Cherche le ***et*** dans la catégorie ***Opérateurs***

➔ Essaie de compléter correctement le code et teste ton programme !

MODULE 9

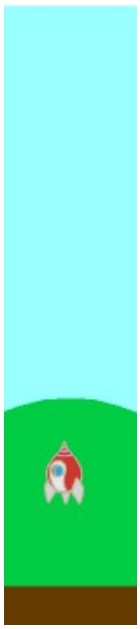
Création des lutins obstacles mobiles
- déplacement aléatoire
- déplacement glissant
- poursuite

Dans les obstacles mobiles, on peut avoir des lutins qui se déplacent aléatoirement, des lutins qui poursuivent le lutin mobile et des lutins qui se déplacent sur un circuit.

Tu peux choisir les trois, ou seulement un des trois. Cela dépend du jeu que tu souhaites programmer.

Un lutin qui décrit un circuit

On souhaite par exemple créer un lutin faisant des **allers-retours** verticalement.



➔ Choisis un nouveau lutin dans la bibliothèque.

Sur ce lutin, tu dois programmer ces déplacements.

Pour t'aider, voici les instructions qu'il faut utiliser :

➔ Dans quelle instruction de contrôle faut-il placer le bloc d'instructions ?

Tu souhaites plusieurs obstacles de ce type ?

Il suffit de **dupliquer** le lutin et de **changer éventuellement le code du lutin dupliqué.**

Un lutin déplaçant aléatoirement

➔ Choisis un nouveau lutin dans la bibliothèque.

➔ Sur ce lutin, crée le code suivant :

➔ Comment l'orienter aléatoirement ?

Cherche l'expression manquante dans la catégorie ***Opérateurs***

➔ Que se passe-t-il si on ne met pas l'instruction "attendre 0.01 secondes" ?

Le lutin change de direction tous les combien de pixels ? Il aura attendu combien de temps durant ce déplacement ?

S'il touche le bord, il faut qu'il rebondisse, insère pour cela l'instruction 

➔ Duplique ce lutin et adaptes les paramètres des différents lutins pour que les déplacements soient comme tu le désires.

Un lutin poursuivant le lutin "mobile"

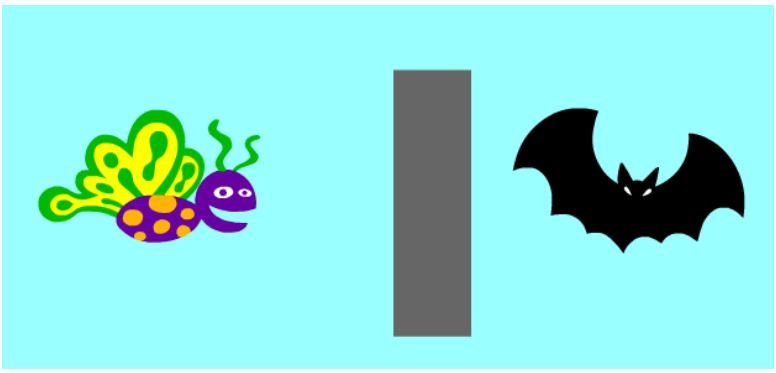
➔ Choisis un nouveau lutin dans la bibliothèque.

➔ Ce lutin doit se déplacer vers le lutin mobile.
Crée ce déplacement en utilisant l'instruction s'orienter vers

On souhaite que ce lutin ne traverse pas les murs. C'est assez délicat à programmer parfaitement.

Comment avions nous procédé pour le lutin "mobile " ?
Quand ce lutin touche la couleur du mur, alors il recule d'un certain nombre de points.

Que va-t-il se passer si on applique la même technique et si on est dans cette configuration :



La bat (chauve-souris) se rapproche du lutin "mobile", rencontre le mur et recule par exemple de 10 points. On se retrouve donc dans la même configuration et la bat est coincée !

On peut améliorer un peu en insérant ce code :

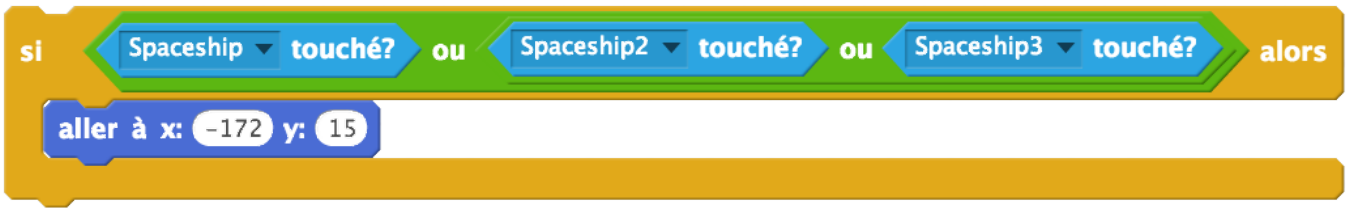
L'idée est que dès qu'elle touche un mur, la bat avance vers la droite....

MODULE 10

Gestion des collisions avec les obstacles mobiles

Dans mon jeu, j'ai trois types d'obstacles mobiles. Nous allons expliquer le traitement des collisions avec les obstacles glissants. Il s'agit des fusées, les lutins Spaceships :

Dans la boucle ***Répéter indéfiniment*** du lutin "**mobile**", il faut tester si le lutin "mobile" touche une des fusées. Si c'est le cas, je décide de renvoyer le lutin mobile à sa place de départ :



Je décide en plus que quand cela arrive alors le joueur perd du temps.

Pour cela, je crée une variable ***malus*** qui vaut 0 au départ.

A chaque fois que le lutin "mobile" touchera une fusée, il sera envoyé au départ et aura un point de malus supplémentaire. J'insère donc l'instruction suivante

Je n'oublie pas de modifier le score maintenant car le score est maintenant égal à
120 – chronomètre – malus !

Quand le lutin "**mobile**" touche un obstacle qui se déplace aléatoirement, j'ai décidé d'augmenter le malus de 1 et de ne pas renvoyer le lutin "mobile" à la case départ. Mais attention si on reste en contact le malus augmente très rapidement !

<u>Tu peux gérer en fait les collisions comme tu veux, cela dépend de ton jeu.</u>

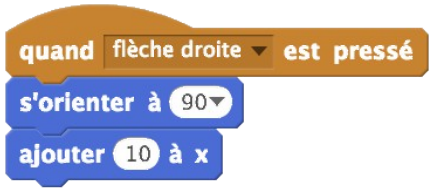
Par exemple, en ce qui concerne le lutin qui me poursuit, j'ai déposé l**e code de la collision sur le lutin qui me poursuit** et ai décidé de le renvoyer à sa position initiale s'il me touche, avec un gros malus.

MODULE 11

Variable vitesse qui peut dépendre :
- des objets collectés
- de certaines touches clavier
- des collisions
- du chronomètre

Dans ton jeu, tu auras peut-être besoin de faire varier la vitesse de déplacement du lutin mobile.
Par exemple, on peut décider que lorsqu'il récupère une clef alors sa vitesse augmente.
On peut également décider d'augmenter ou de diminuer la vitesse à l'aide de touches du clavier.

Où apparaît la vitesse dans le code ? Elle apparaît ici :



➔ Dans tous les cas, la première chose à faire est de créer une variable ***vitesse*** et de l'initialiser à une certaine valeur, 10 dans mon jeu.
Modifie les blocs gérant les déplacements pour utiliser la variable ***vitesse***.

➔ Dans mon jeu encore, j'ai décidé d'augmenter la vitesse de 1 à chaque fois que je récupère une clef.
Il suffit d'augmenter la vitesse de 1 dans la partie du code correspondante à la clef touchée par le lutin mobile.

Tu peux aussi créer 2 autres événements Quand telle touche pressée pour augmenter ou diminuer la vitesse. Tout dépend de ton jeu.

Tu peux aussi créer une touche à bascule :

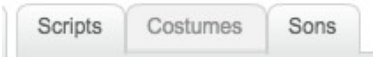
Que fait au juste le code suivant ?

MODULE 12

Les sons pour :
- musique de fond
- deplacement du lutin mobile
- collisions
- etc

Commençons par la musique de fond.

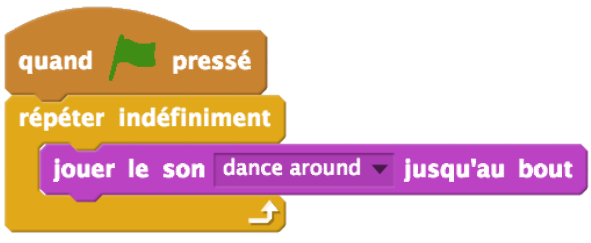
➔ Sélectionne le lutin "mobile" et clique sur l'onglet son



➔ Clique sur

➔ Clique sur la catégorie **Boucles musicales** et choisis une musique (double clique dessus).
J'ai choisi **dance around**.

Enfin, il suffit d'insérer le code suivant dans le lutin "mobile" :

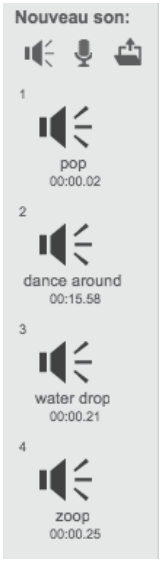


Le son dance around apparaît dans la liste car on est d'abord allé le chercher dans la bibliothèque.

Ajoutons les effets sonores des collisions et des déplacements.

Ton lutin "mobile" étant sélectionné, clique sur l'onglet Sons et sélectionne un son dans la bibliothèque.

Quand tu doubles cliques sur un son, il apparaît dans la liste et devient disponible pour ton lutin :



Ensuite dans la Catégorie ***Sons***, tu as l'instruction

(le son pop est disponible par défaut, inutile d'aller le chercher avant dans la bibliothèque)

Il suffit alors de sélectionner le son désiré et d'insérer l'instructions au bon endroit.

Par exemple, dans le code gérant le déplacement ou dans le code gérant une collision....

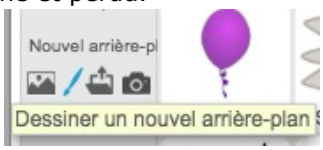
➔ Pour modifier le volume sonore, utilise l'instruction Mettre le volume sonore au niveau …

MODULE 13

Ecrans de démarrage et de finalisation

➔ Commençons par créer les écrans finaux : Gagné et perdu.

Clique sur Dessiner un nouvel arrière-plan



➔ Clique sur le bouton **Vectoriser** puis crée une zone de texte "Gagné !"

➔ Crée un autre arrière-plan Perdu !

Quand la partie est gagnée ou perdue, il suffit de basculer sur l'arrière-plan correspondant.

➔ Quand la partie commence, il faut basculer sur l'arrière-plan du jeu !

Donc il faut insérer cette instruction au début du programme 

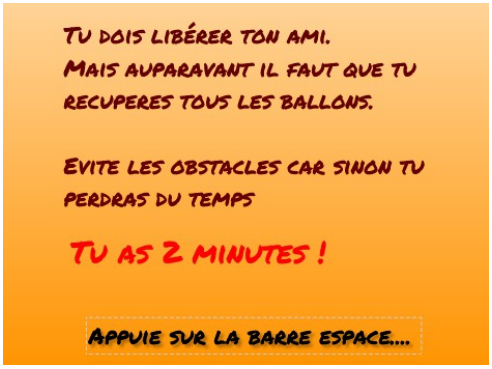
➔ A toi de personnaliser ces arrières-plans

➔ Créons maintenant l'écran de démarrage.

Dans un nouvel arrière-plan, je crée un dégradé à l'aide d'un rectangle recouvrant toute la scène.

Je l'appelle ***debut***

Puis il suffit de créer des zones de textes.



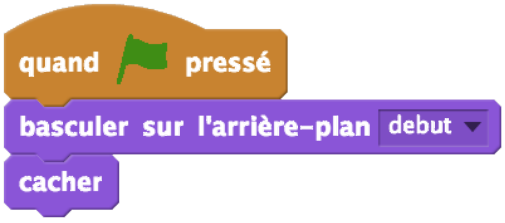
Cet écran s'affiche au lancement du programme et il ne faut lancer le jeu que quand le joueur appuie sur la barre espace.

C'est compliqué car il faut lancer tous les scripts des lutins à ce moment là.

Dans tous les scripts des différents lutins, il faut faire ceci :

On affiche la page de présentation

On cache le lutin



Quand le jeu commence,

On montre le lutin

Et tout le code précédent suit

Par exemple dans le lutin ballon :

Le jeu commencera réellement quand les lutins recevront **go**

go est un événement

Cet événement sera déclenché quand on appuiera sur la barre espace :