Jeu sur les espèces envahissantes dans Scratch	6e année – Comprendre les systèmes de la vie

Guide de programmation

Avant d'expliquer en détail comment programmer notre jeu sur les espèces envahissantes, établissons d'abord les comportements qu'il devrait avoir. Nous voulons illustrer l'effet de l'agrile du frêne sur les écosystèmes établis. Pour ce faire, nous permettrons au joueur de contrôler une petite créature poilue des bois. L'objectif du joueur sera de « manger » (en touchant) les frênes apparaissant de façon aléatoire. Cela fera disparaître l'arbre et augmentera notre pointage d'un point. En compétition avec le joueur, un agrile du frêne (et, au fur et à mesure que le pointage augmente, d'autres agriles du frêne). Nous programmerons le jeu de manière à ce que si les agriles du frêne touchent un arbre ou le joueur, la partie est terminée.

La façon dont Scratch est structuré est que des scripts sont liés aux sprites, des acteurs individuels dans le jeu. Nous aurons donc besoin d'un sprite de joueur, d'un sprite d'arbre et d'un certain nombre de sprites d'agrile du frêne (heureusement, nous n'avons qu'à en programmer un seul). Cependant, avant de commencer à programmer ces sprites, nous devons d'abord (comme le font tous les programmeurs) définir nos variables.

Nous avons besoin d'une seule vraie variable pour ce script : le pointage. Chaque script Scratch commence avec un sprite (un joyeux chat) et une variable (appelée « my variable » [ma variable]). Renommons-les.

Renommer une variable
Clique sur le cercle orange sur le côté gauche de la fenêtre Scratch qui indique « variables ». Clique avec le bouton droit sur le bloc intitulé *my variable* et sélectionne *rename variable (renommer la variable).* Nomme-la *score (pointage).*

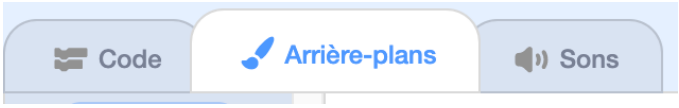
Clique la case à cocher à côté du pointage pour rendre le pointage visible au joueur. Maintenant, dans le coin supérieur droit de la fenêtre de jeu, tu peux voir le pointage représenté par un 0. Cela restera visible au joueur.

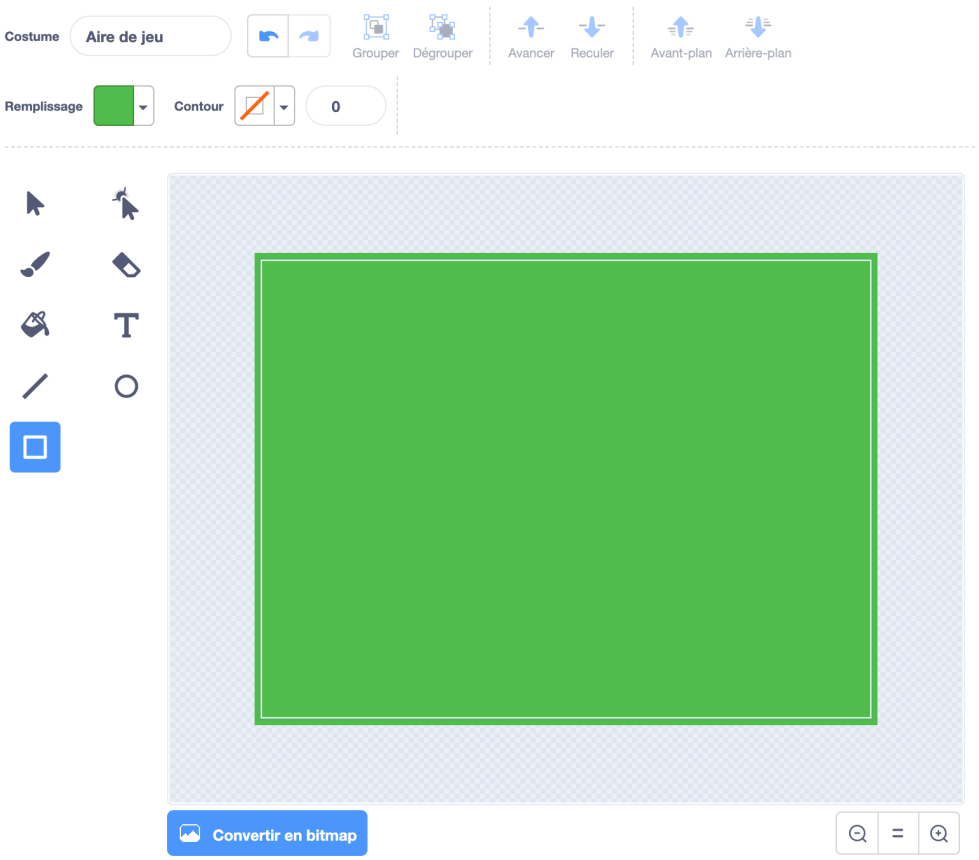
Avec les variables définies, il est temps de préparer la scène pour notre jeu.

Arrière-plans

La « scène » sur laquelle notre jeu se déroulera s'appelle un « arrière-plan » (backdrop). Tu peux considérer les arrière-plans comme une sorte de variable visuelle : l'arrière-plan à utiliser est certainement traité par le code comme une variable, donc nous devons les définir en premier lieu.

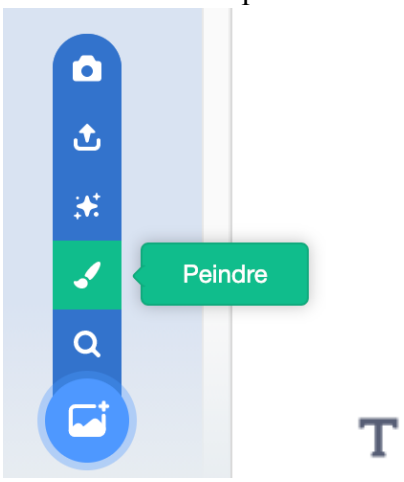
Nous voulons deux arrière-plans : un pour jouer et un qui nous servira d'écran de fin de partie. Tu peux choisir un arrière-plan, le téléverser de ton ordinateur ou utiliser l'outil « paint » (peinturer) pour créer un arrière-plan simple. Fais cela en cliquant sur l'onglet « stage » (scène) en bas à gauche, puis sélectionne l'onglet « backdrops » dans le coin supérieur droit (ou utilise le bouton dans le coin inférieur gauche pour choisir un arrière-plan).
Au moyen de l'outil « rectangle », nous ferons un simple arrière-plan vert pour cet exemple. Sélectionne simplement la couleur verte sous la couleur de remplissage, clique sur l'outil de rectangle et glisse le rectangle sur l'ensemble de l'aire en damier.





Remarque que nous avons également renommé le costume de « backdrop1 » à « Play » (Aire de jeu). Nous créerons maintenant un deuxième arrière-plan (s'il n'a pas été automatiquement créé). Placez le curseur de votre souris sur l'icône d'image circulaire dans le coin inférieur GAUCHE. Choisissez le pinceau et un nouvel arrière-plan sera créé en mode « paint ».

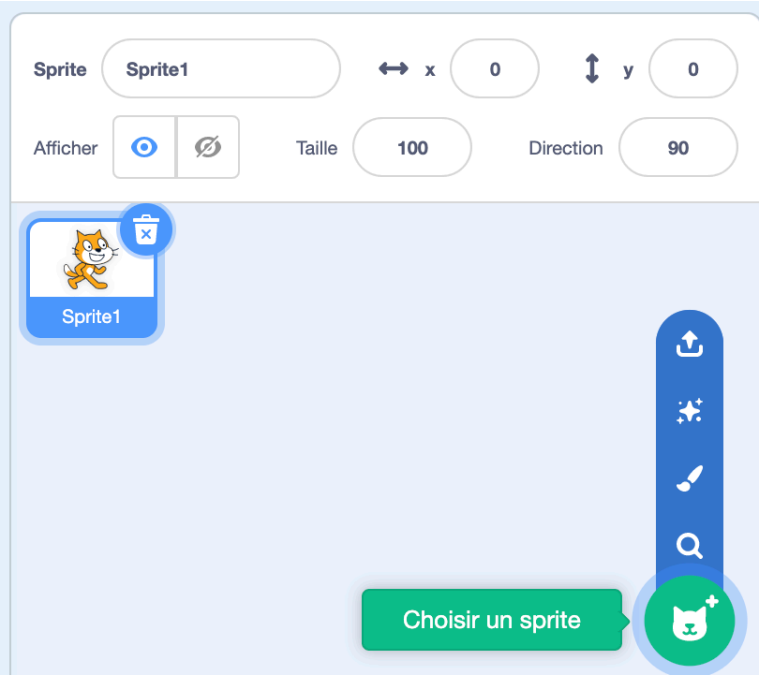
Nous utiliserons le mode « paint » pour créer un écran de fin de partie. De nouveau, remplis l'aire avec une boîte verte. Maintenant, utilise l'outil de texte pour écrire une explication appropriée. Par exemple : Partie terminée! L'agrile du frêne détruit les habitats d'autres animaux de forêt, rendant l'adaptation et la survie difficiles pour eux.



Sprites

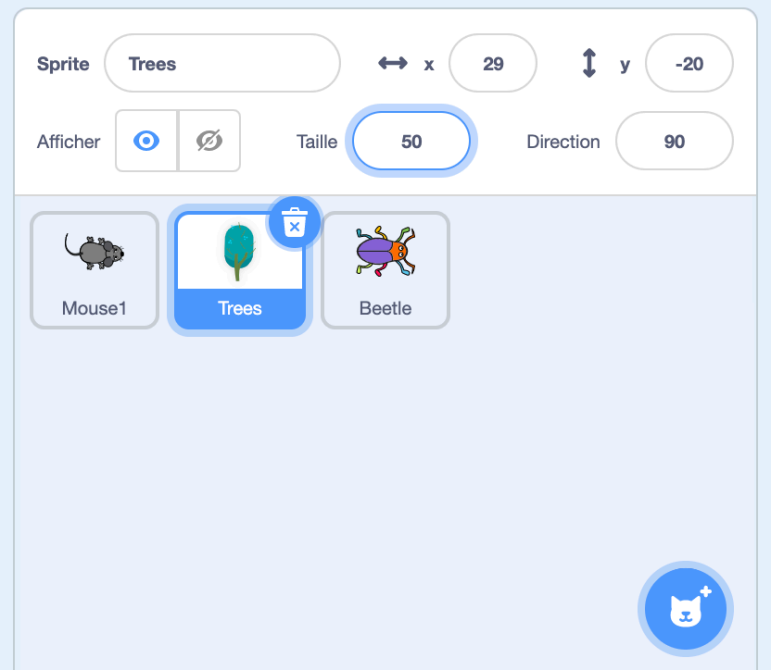
Comme les arrière-plans, c'est une bonne pratique de créer les sprites à l'avance. Nous en créerons trois : le personnage du joueur, l'arbre et le coléoptère.

Nous avons déjà un sprite de créé, donc il suffit de changer son costume et de le renommer (dans Scratch, l'image associée à un sprite s'appelle un « costume »; cela peut porter à confusion pour certains, puisque dans d'autres environnements de programmation ce que Scratch appelle un costume est appelé un sprite).



Sélectionne le sprite, puis clique sur le bouton « Choose a Sprite » (Choisir un sprite). Sélectionne « animals » (animaux), puis un animal que tu peux trouver dans un habitat forestier. Pour cet exemple, nous utiliserons une souris (si tu crées un nouveau sprite par accident, supprime simplement le sprite1).

Pour créer un nouveau sprite, fais la même chose sans avoir sélectionné préalablement un sprite existant. Souviens-toi, nous avons besoin d'un arbre et d'un coléoptère. Tu devrais avoir quelque chose de semblable à ceci dans le panneau des sprites du coin inférieur droit :

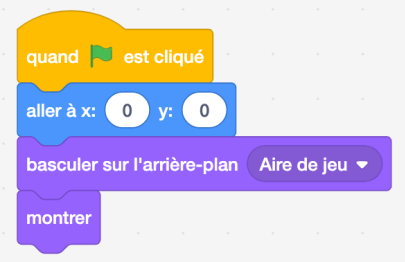


Puisque nous ne voulons pas que nos sprites apparaissent tous au centre de l'écran, configure-les à « hide » (masquer) en cliquant sur l'icône  de chaque sprite. Les sprites sont maintenant invisibles jusqu'à ce que le code les demande.

Nous sommes prêts à commencer à programmer.

Programmation : Le sprite du joueur

Le personnage du joueur doit faire deux choses : suivre le curseur de la souris (ou le doigt) et pointer le costume de souris vers l'arbre, soit la cible de la souris.

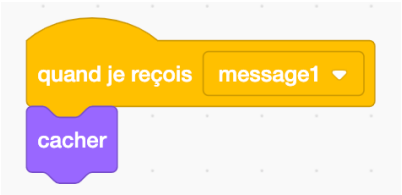


Nous voulons initialiser la souris lorsque l'on clique sur le drapeau vert (un bloc d'événement jaune), la faire bouger (un bloc de mouvement bleu) et s'assurer que l'arrière-plan est notre arrière-plan d'aire de jeu (un bloc d'apparence mauve), puis l'afficher (un autre bloc d'apparence mauve). Souviens-toi que les blocs doivent se connecter, comme le montre l'image à gauche. Placer les blocs les uns près des autres n'est pas suffisant.

Nous devons maintenant placer la souris dans une boucle où elle suit toujours (du moins, jusqu'à la fin de la partie) le curseur. Nous utiliserons un bloc de contrôle orange *forever (infini)*. À l'intérieur de cette boucle, nous dirons à la souris d'aller (bloc de mouvement bleu *go to*) vers le curseur de la souris (clique sur le menu déroulant sur le bloc et sélectionne *mouse-pointer*). Afin que la souris ne fasse pas que maladroitement glisser dans une direction, nous la ferons se tourner vers les arbres au moyen du bloc de mouvement bleu *point towards (pointer vers)*. Nous placerons également un bloc *wait 1 seconds (attendre 1 seconde)* avant la boucle afin que le jeu ne démarre pas immédiatement. Cela est

habituellement un peu déroutant pour les joueurs. Nous avons presque terminé tout le script dont nous avons besoin pour la souris. Nous laisserons le pointage aux arbres et les conditions de fin de partie seront vérifiées par le coléoptère. Cependant, nous voulons que la souris disparaisse lorsque la condition de fin de partie est fixée par le coléoptère.

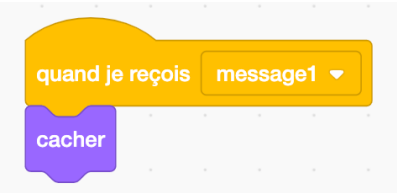
Pour cela, nous créerons un deuxième script, en commençant par le bloc d'événement jaune *when I receive message1 (lorsque je reçois le message1)*. Le message1 sera le signal de fin de partie diffusé par l'autre sprite (ce message peut également être renommé « fin » ou « fin de partie » si tu le veux, c'est en fait une meilleure pratique; pour ce faire, crée un nouveau message à partir du menu déroulant dans le bloc *when I receive*). Peu importe comment nous appelons le signal de fin de partie, lorsque nous le recevons, nous voulons simplement cacher la souris au moyen de bloc d'apparence mauve *hide.*

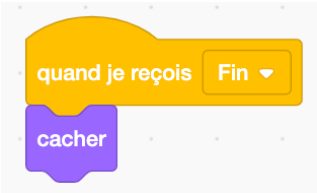


Programmation : Le sprite de l'arbre

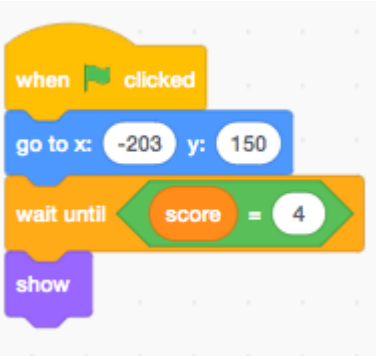
L'arbre doit apparaître à un endroit aléatoire, détecter s'il est entré en contact avec la souris et, si c'est le cas, disparaître et réapparaître à un autre endroit aléatoire. Cela doit être répété encore, et encore, et encore. Commence avec le bloc d'événement jaune *when green flag clicked (lorsque l'on clique sur le drapeau vert)*. Comme pour la souris, nous attendrons une seconde, puis commencerons une boucle « infinie » pour le comportement. Nous devons également préciser que le pointage doit être zéro au début du programme. Utilise le bloc de variable orange *set* (*fixer)* quelque part avant le bloc *wait (attendre)*. À l'intérieur de la boucle, nous utiliserons le bloc de mouvement bleu *go to,* mais plutôt que d'indiquer un endroit exact, nous glisserons dans le bloc l'opérateur vert *pick random (choisir au hasard)*. Nous voulons que l'arbre apparaisse sur l'aire de jeu, donc la valeur aléatoire pour X devrait se limiter à -200 à 200. Pour Y, elle doit être de -150 à 150 (l'aire de jeu est plus large que haute). Après s'être rendu à son emplacement l'arbre peut se montrer au moyen du bloc mauve *show (montrer)*. Nous voulons ensuite attendre (au moyen du bloc de contrôle orange *wait until*) jusqu'à ce que l'arbre touche la souris (mouse1) (au moyen du bloc de perception bleu pâle *touching*). Ensuite nous changeons le pointage de 1 (*change score by 1)* et masquons (*hide*) l'arbre avant d'attendre 0,5 seconde pour recommencer la boucle. Il est important d'attendre avant de recommencer la boucle, sinon le programme pourrait compter des points en trop.

Comme pour la souris, nous voulons que l'arbre disparaisse à la condition de fin, donc nous utilisons les mêmes blocs pour un deuxième script. Selon la façon dont tu as configuré les choses, cela ressemblera à l'une des images suivantes :



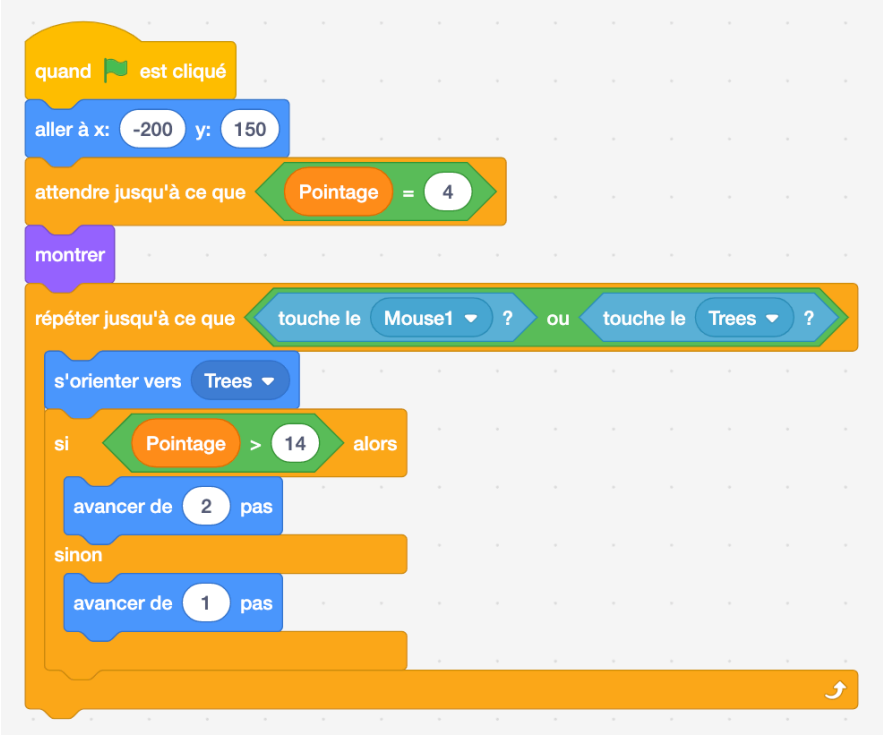


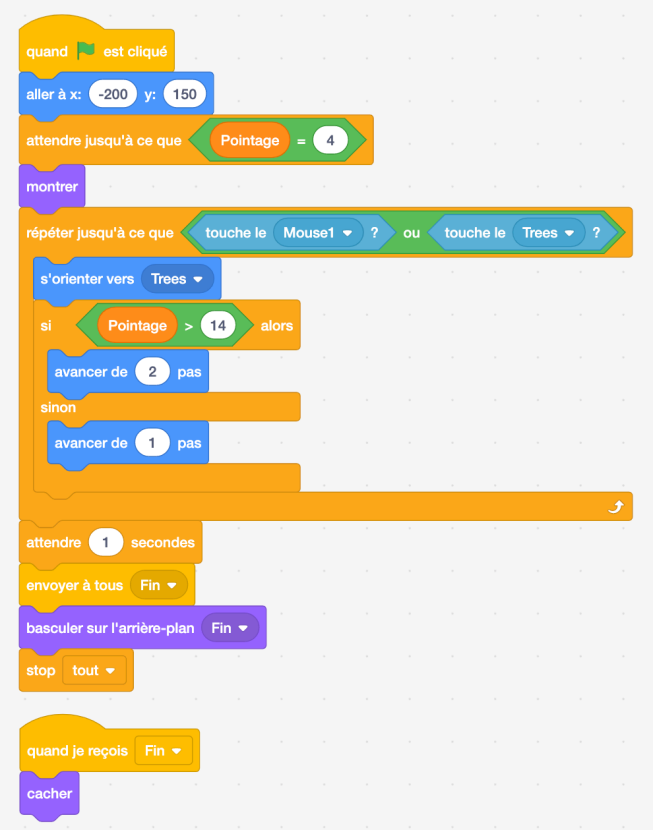
Programmation : L'agrile du frêne



Pour initialiser l'agrile du frêne, nous utiliserons un script semblable à l'un de ceux que nous avons utilisés avant, soit le même bloc d'événement, mais nous utiliserons le bloc *go to* pour placer le coléoptère à l'extérieur de l'écran afin que l'arbre n'apparaisse pas accidentellement dessus. Nous utiliserons le bloc *wait until* avec l'opérateur de bloc = pour attendre que le pointage atteigne 4 avant de dévoiler le coléoptère. Cela permettra au joueur de commencer la partie sans interférence de la part du « méchant ».
Une fois montré, le comportement du coléoptère devrait être dans une

boucle. Plutôt qu'une boucle infinie, nous voulons seulement faire une boucle jusqu'à ce que le coléoptère touche un arbre ou le joueur. Souviens-toi, il s'agit de nos conditions de fin de partie. Nous utiliserons donc le bloc de contrôle *repeat until (répéter jusqu'à)* conjointement avec le bloc d'opérateur *or (ou)* vert et deux copies du bloc de perception *touching* : un configuré pour le sprite de l'arbre (dans ce cas-ci, Trees) et un configuré pour le sprite du joueur (dans ce cas-ci, Mouse1). Le comportement répété qui se produira jusqu'à la fin de la partie ira à l'intérieur de cette boucle; après la boucle, nous aurons les blocs qui programment la fin du jeu.





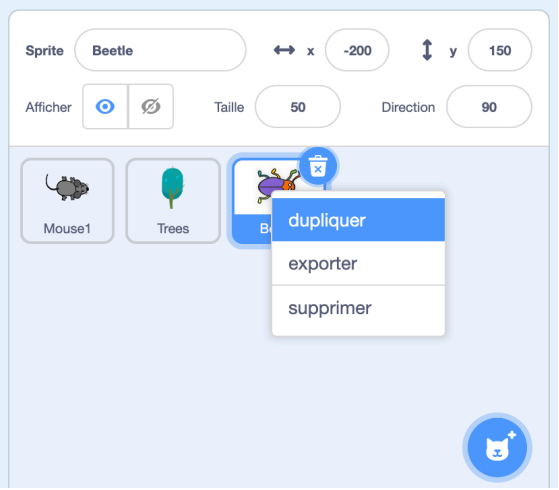
À l'intérieur de la boucle, nous utiliserons le bloc de mouvement bleu *point towards* pour tourner le coléoptère vers l'arbre les blocs *move _ steps (déplace _ pas)* pour le faire avance. Pour faire augmenter la difficulté du jeu au fur et à mesure qu'il progresse et que le pointage du joueur augmente, nous utiliserons un bloc de contrôle *if/else (si/sinon)* pour que l'agrile se déplace plus rapidement si le pointage est supérieur à 14 (le pointage conditionnel peut être changé si voulu et plus de pas peuvent être ajoutés; sinon, une formule mathématique liant le pointage à la vitesse pourrait être adoptée à la place).

Enfin, nous voulons programmer ce qui arrive une fois que le coléoptère touche la souris ou l'arbre. Nous voulons certainement diffuser (*broadcast)* notre message de fin de partie (ici, nous l'appellerons « end ») au moyen du bloc d'événement jaune. Souviens-toi, cela masquera tous les autres objets. Nous pouvons ensuite changer l'arrière-plan (au moyen du bloc mauve

switch backdrop [changer d'arrière-plan]) pour notre écran de fin de partie. Nous voulons aussi interrompre tous les autres scripts afin de ne pas recevoir accidentellement des points en touchant des arbres invisibles avec notre souris invisible. Un bloc *wait* est inclus simplement pour permettre au joueur d'avoir la chance de voir le coléoptère qui a déclenché la fin de la partie. Autrement, il pourrait avoir une surprise insatisfaisante de voir tout disparaître trop rapidement. Remarque que nous incluons aussi le deuxième script *when I receive end, hide* pour le coléoptère. Cela nous permet d'omettre le boc *hide* dans le script principal et aussi d'inclure d'autres ennemies dans la prochaine section.

Ajouter d'autres coléoptères

Clique avec le bouton droit sur le sprite du coléoptère et clique sur « duplicate » (reproduire).



Tout le code associé à ce sprite est récré dans le nouveau sprite Beetle2 et peut être modifié sans toucher au sprite Bettle1. Maintenant, pour rendre le jeu plus équilibré, il est logique de changer les conditions qui régissent l'apparition et l'accélération du sprite Beetle2. Disons que le sprite Beetle2 apparaît lorsque le pointage atteint 6 et change de vitesse à un pointage de 16. Un nombre arbitraire de coléoptères peut être créé de cette façon.

Tu peux réduire la taille de tous les sprites à 50 (les rapetissant de moitié) afin que tous ces coléoptères puissent avoir assez de place à l'écran!