Session 8 : Niveau 1-Etape2

Dans cette seconde étape, nous allons apprendre à gérer plusieurs arrière-plans, à créer un compteur de temps, et à communiquer par messages.

Préparation du projet

- Ouvrir le projet Session6.sb2 et le sauvegarder sous le nom « Session8.sb2 »
- Sélectionner la scène et cliquer sur « Arrière-plans ».
- Dupliquer le premier arrière-plan deux fois.
- Sur sol_martien2, ajouter le texte « Gagné » et sur sol_martien3 ajouter le texte « Perdu »

Créer un chronomètre

Un chronomètre est une variable dont la valeur change toutes les secondes. Si l'on veut programmer un chronomètre qui compte une minute on peut utiliser l'algorithme :

Initialiser la variable chronomètre à 0

Répéter 60 fois

| **attendre une seconde**

| **ajouter à la variable chronomètre la valeur 1**

Fin répéter

Lorsque la boucle s'arrête, la minute est écoulée.

Où placer le programme correspondant à cet algorithme ?

On pourrait le placer dans la zone de programmation de n'importe quel lutin. Cependant la gestion du chronomètre n'est pas propre à un lutin donné.

Il n'est pas plus logique de le placer dans le « rover », que dans un des lutins « eau » ou « nourriture ».

Nous placerons donc ce programme dans la zone programme de l'arrière-plan.

Lorsque le chronomètre arrive à 60, tout doit s'arrêter.

Nous disposons dans la catégorie « Contrôle » de la commande « Stop tout »

Cette commande arrête tous les programmes (scripts) qui tournent dans une boucle comme ceux des lutins « eau » et « nourriture ». Par contre les scripts du rover qui teste l'appui ou non sur une flèche du clavier, eux ne s'arrêtent pas, car ce sont des scripts indépendants qui se déclenchent lorsqu'un évènement extérieur arrive.

Comment faire pour bloquer le rover, une fois que le temps est écoulé ?

Il suffit d'inclure tout le code gérant les flèches dans une boucle infinie :

Répéter indéfiniment
| **Si flèche haut pressée alors**
| **|** **….**
| **Fin si**
| **Si flèche bas pressée alors**
| **|** **….**
| **Fin si**
| **Si flèche gauche pressée alors**
| **|** **….**
| **Fin si**
| **Si flèche droite pressée alors**
| **|** **….**
| **Fin si**
Fin répéter

Quand avons-nous gagné ?

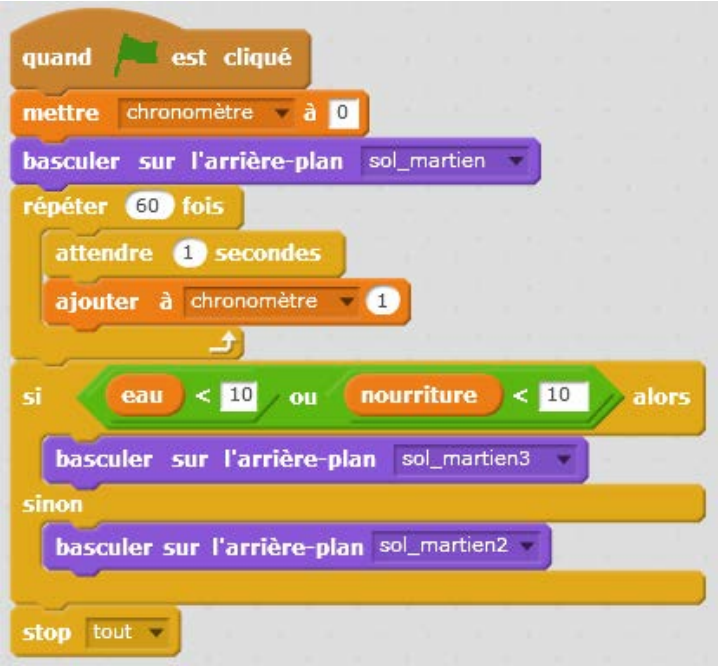
Nous gagnons lorsque nous avons récolté la quantité d'eau et de nourriture nécessaire à la survie de la mission, avant la fin du temps de la minute.

Comment fixer les valeurs minimum de la quantité d'eau et de la quantité de nourriture à récolter ?

On fixe cette valeur dans le programme (10 par exemple)

Lorsque la minute est écoulée, **si** la variable eau a une valeur inférieure à 10 **ou** si la variable nourriture a une valeur inférieure à 10, **alors** nous avons perdu, **sinon** nous avons gagné.

Si nous avons perdu, l'arrière-plan « sol_martien3 » doit s'afficher avant que tout s'arrête, sinon c'est l'arrière-plan « sol_martien2 » qui doit s'afficher.



Comment cacher les lutins lorsque la partie est terminée ?

Lorsque « Gagné » ou « Perdu » s'affiche, nous aimerions que tous les lutins disparaissent de la scène.

Scratch propose un système de messagerie, permettant de synchroniser les lutins entre eux.

1. un lutin ou l'arrière-plan crée un message, qu'il envoie à tous les autres lutins ou arrière-plan, ou bien qu'il envoie à un lutin particulier

2. les lutins ou l'arrière-plan autorisent l'ordinateur à prendre en compte un message particulier et à le leur faire parvenir.

3. lorsque le lutin qui a autorisé l'arrivée d'un message particulier, le reçoit, il exécute les instructions qui conviennent.

Dans notre jeu, lorsque la partie est terminée, l'arrière-plan crée un message « fin » et l'envoie à tous (catégorie « **Évènements** ») :

que nous plaçons juste avant le bloc « Stop tout »

Dans tous les lutins nous plaçons le script :

et nous rajoutons 

au début du script de chaque lutin, car si après une partie, nous en démarrons une autre, les lutins restent cachés.