Projet Scratch

Réalisation d'une fiction interactive en « point and click »

26/02/2015

Table des matières

I. Qu'est ce qu'une fiction interactive en point and click ?

Ce sont des histoires avec lesquelles le joueur interagit grâce à la souris. Ces jeux très populaires dans les années 1980-1990 avec des jeux très connus tels que Monkey Island ou Day of the Tentacle, connait depuis les années 2000 un certain renouveau (Runaway, Syberia, Machinarium)

Mais en quoi consistent ces jeux ? Ce sont d'abord des histoires et par conséquent un scénario. Celui-ci tourne souvent autour d'une mission à accomplir pour le héros (ou l'héroïne) ou d'une enquête. Au cours de cette histoire, le personnage principal est entouré de personnages secondaires, qui peuvent l'aider, lui donner des indices ou au contraire tenter de l'empêcher de réussir sa mission. Il peut aussi exister des personnages qui participent uniquement au « décor ».

Dans ce type d'histoire, le personnage principal se déplace de lieu en lieu, fouille, récolte des objets et des indices et utilise ces objets pour avancer dans sa mission. Les déplacements dans ces différents lieu peuvent être linéaires, c'est-à-dire qu'on ne peut passer au lieu suivant qu'en ayant résolu l'énigme ou réaliser les actions liées à cette pièce. Mais il est également possible que le héros puisse explorer un ensemble de pièces ou de lieux, afin de trouver un ensemble d'indices ou d'objets utiles pour avancer.

Ces jeux sont également caractérisés par leur interface.

L'interface est souvent décomposée en plusieurs parties :

- La scène, dans laquelle se trouvent des personnages avec lesquels le héros peut dialoguer, ou auxquels il peut donner des objets, et /ou des objets qu'il peut observer ou prendre ou utiliser.
- Une fenêtre inventaire, dans laquelle on peut voir tous les objets que le héros a récupérés. Cette fenêtre peut être visible en permanence ou apparaître et disparaître sur commande du joueur (avec la souris et/ou raccourci clavier).
- Une fenêtre permettant de sélectionner les actions du héros (marcher, prendre, parler, regarder…).

Généralement le héros est visible dans la scène (jeu à la troisième personne) bien qu'il existe des jeux dits « à la première personne » dans lesquels on voit à travers les yeux du héros. Nous nous intéresserons seulement au premier type.

Au niveau des dialogues, dans les jeux les plus évolués, les répliques du héros sont à choisir parmi un ensemble de possibilités.

II. Le projet

Le projet que nous vous proposons est de réaliser un jeu d'aventure en « point and click » à l'aide du logiciel Scratch.

Pour cela vous serez en charge de :

- L'écriture du scénario,
- La création des personnages, des décors et des sons et musiques,
- Le développement du jeu avec toutes les interactions nécessaires entre les divers éléments du jeu entre eux et avec le joueur.

Pour la réalisation de ce jeu vous aurez quelques contraintes, et beaucoup de liberté.

Commençons par les contraintes :

- Le héros doit être visible dans la scène (pas de jeu à la première personne),

- Le héros doit avoir des interactions avec le joueur, les autres personnages et les objets,

- Le héros doit progresser en résolvant des énigmes (à l'aide d'objets ou non), les professeurs pourront décider du thème des énigmes,
- Le nom des objets doit apparaitre lorsque la souris passe dessus,
- Les dialogues sont fixés (pas de choix possibles).

En revanche vous êtes libres de décider

- Du thème du jeu (Aventure, enquête….),
- Du scénario,
- De la manière dont vous réalisez les personnages, décors, sons (voir chapitre ressources),
- Si l'histoire est linéaire ou non (attention le second choix est plus complexe),
- Si le joueur peut choisir les actions ou si celles-ci sont imposées par le jeu en fonction de l'objet cliqué (attention le premier choix est plus complexe),
- Si le jeu possède un inventaire ou non (attention le premier choix est plus complexe).

Vous travaillerez par groupe de 4, ou plus exactement par groupe de 2 binômes. L'ensemble des tâches hors ligne seront réalisées à 4. Ensuite chaque binôme travaillera sur des tâches différentes de développement. Enfin, les différents développements seront fusionné dans un seul projet.

A) Le travail hors ligne

La première ou les deux premières séances seront consacrées aux tâches sans ordinateur. Il s'agit de spécifier votre projet, c'est-à-dire écrire les éléments importants du projet, éléments dont chaque membre du groupe aura connaissance lors du développement.

1. Synopsis

Il s'agit de rédiger en quelques lignes le sujet de votre jeu. Par exemple : un début de synopsis pourrait consister en : « Gandalf demande à Frodon de détruire l'Anneau Unique pour sauver la Terre du Milieu Il est aidé par des hommes, des elfes, des nains et bien sûr ses amis Hobbits et doit. affronter des orcs, des gobelins et bien sûr Sauron . »

2. Personnages

Pour chaque personnage important du jeu, (ne le faites pas pour les personnages décoratifs), il vous faudra réaliser une fiche permettant de préciser :

- Son nom,
- Son âge,
- Son caractère (pas obligatoire),
- Une description physique, avec éventuellement un dessin,
- Son métier, ou son occupation,
- Ses relations avec le héros. (avant le jeu et pendant le jeu).

3. Lieux

De même que les personnages, chaque lieu devra posséder une fiche contenant :

- La description des lieux,
- Les liens et accès vers les autres lieux,
- Les spécificités de ce lieu (par exemple : fermé à clé, ou accessible sous certaines conditions).

4. *Scénario*

Phases d'un scénario

Dans les jeux comme dans les films, le scénario est souvent découpé en trois parties.

Etape 1 : l'exposition

C'est la phase durant laquelle on plante le décor : on présente les personnages principaux et le fil directeur de l'histoire. Elle se termine par l'évènement perturbateur, l'élément déclencheur de l'aventure. Ce n'est généralement pas là qu'est exposé l'objectif final de la quête, celui-ci se précisera dans la phase 2.

Dans les jeux qui nous intéressent, cette phase est généralement une animation sans ou avec très peu d'interaction. Elle permet de poser rapidement l'histoire.

Etape 2 : la confrontation

C'est le cœur du jeu, c'est dans cette phase que le héros comprend vraiment le sens de sa quête. C'est la que va se concentrer l'essentiel de vos développements et des interactions entre le joueur et son héros !

Etape 3 : la résolution

A cette étape, plus rien ne doit être flou, c'est le dénouement de l'histoire. Comme pour la première phase, dans les jeux que nous visons, cette dernière phase est généralement une animation.

Nous vous demandons d'essayer au maximum de coller à ce déroulement.

A vous de jouer !

Pour chaque séquence de l'histoire, le scénario doit contenir (voir fiche scénario):

- Le lieu,
- Les objets présents et les actions possibles sur ces objets,
- Les personnages présents,
- Les actions des personnages,
- Les dialogues,
- Ce que doit faire le héros (pour passer à la suite).

B) Le développement

1. *Organisation*

Pour développer de manière efficace votre projet, il faut en premier lieu planifier votre travail.

Cela se traduit par les actions suivantes :

- Découper en tâches, par exemple :
  - Création des décors (sous tâche = 1 décor) ,

- Création des personnages,
- Création des musiques et sons,
- Gestion des déplacements du héros,
- Développement de la scène N°1 (2, 3..),
- Gestion de l'inventaire.

- Mettre des priorités sur ces tâches. Par exemple la tâche « créer une musique pour la scène de fin » est probablement moins prioritaire que la tâche « création du décor scène N°1 ».
- Le présenter sous forme de tableau, 1 ligne par tâche, et des colonnes Fait, En cours, à Faire.

2. *Quelques conseils*

Soyez ambitieux, mais pas trop. Mieux vaut faire un petit projet qui fonctionne bien, quitte à rajouter des éléments à la fin qu'un gros projet qui ne fonctionne pas du tout.

Nous vous conseillons donc de faire les choses petit à petit. Testez régulièrement vos scripts. Essayez d'avoir un élément qui fonctionne à la fin de chaque session.

Pensez également qu'à la fin il faudra fusionner les éléments développés par chaque binôme. Il faut donc bien identifier les transitions entre les scènes (comment sont elles déclenchées : messages ; évènement..)

Pensez à INITIALISER vos scripts !

3. *Ressources*

Pour développer votre projet, plusieurs possibilités s'offrent à vous.

Pour les lutins, décors, musiques et sons :

- Les ressources de la bibliothèque Scratch (le plus simple).
- Créer vous-mêmes
  - les décors et les lutins à l'aide de l'outil de dessin ou à partir de photos de vos dessins à la main, de décors en Lego, ou même de copie d'écrans (Minecraft ?),
  - les musiques et les sons avec l'option d'enregistrement.

- Réutiliser les créations des autres scratcheurs.
- Utiliser des ressources (photos, sons, musiques) sur internet gratuites mais SURTOUT libres de droit. Par exemple sur les sites suivants vous pourrez trouver votre bonheur :
  - Bruitages :
    - http://www.universal-soundbank.com/
    - http://www.sound-fishing.net/bruitages.html
    - http://lasonotheque.org/
    - http://www.freesound.org/
  - Musiques
    - http://www.auboutdufil.com/
    - https://www.jamendo.com/fr/

- Images
  - http://pixabay.com/fr/
  - http://www.stockvault.net/

Pensez à citer les noms des auteurs si vous utilisez la dernière solution.

En informatique, on a le droit de copier si ce qu'on copie est intelligent, nous fait gagner du temps, et que l'on comprend ce que l'on copie. Donc pour votre développement, vous pouvez utiliser les scripts des copains, ou ceux d'autres scratcheurs mais comprenez bien auparavant à quoi ils servent et comment les adapter à votre jeu.

III. Rappels et aide

A) Rappels sur Scratch

1. *Scratch ? C'est quoi ça ?*

Scratch est un environnement de développement graphique qui permet de réaliser des animations interactives très facilement, grâce notamment à un langage de programmation (ou script) visuel.

Il a été développé par le MIT (Massachussetts Institute of Technology) et est disponible gratuitement en ligne (http://scratch.mit.edu) ou via un logiciel à installer sur l'ordinateur.

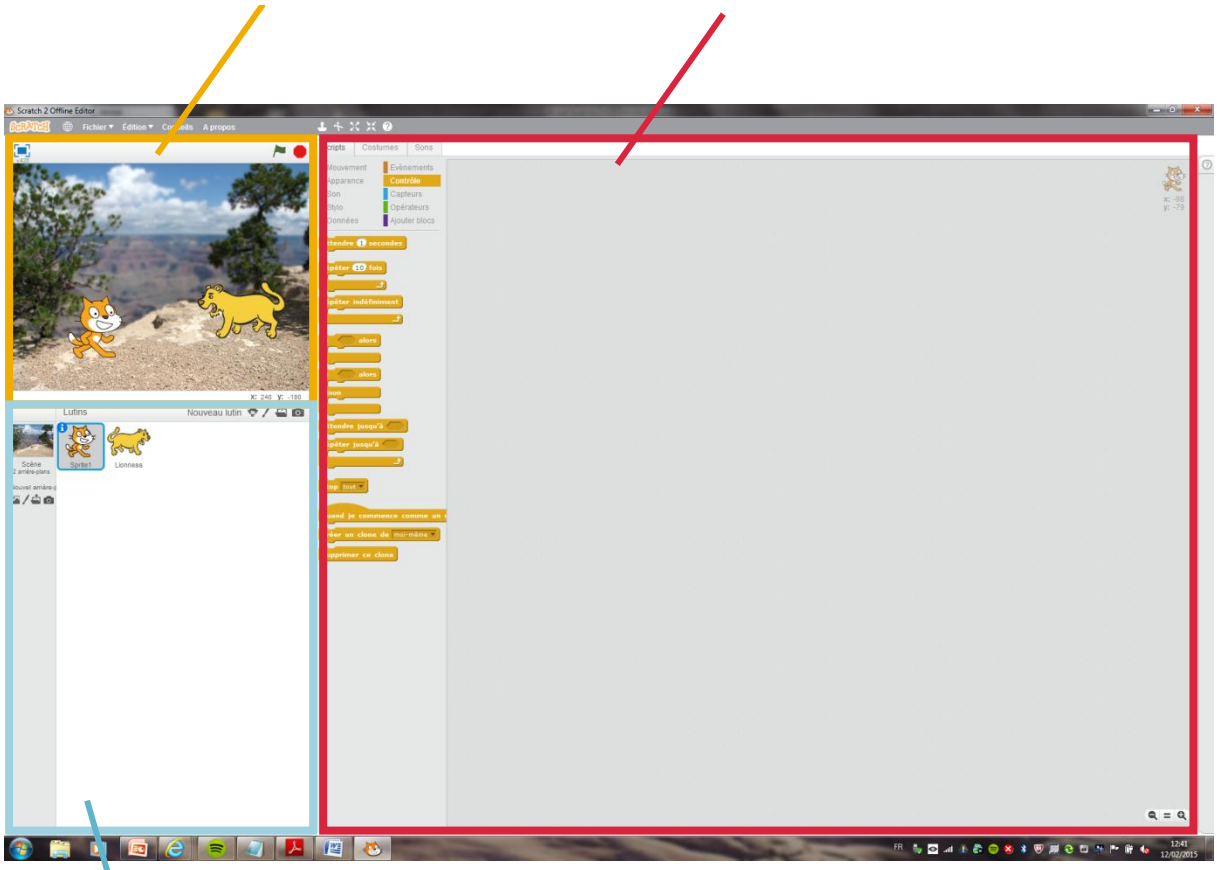
2. *Interface*

L'interface de scratch est divisée en trois zones :

- La zone de rendu : ou l'animation se joue,
- L'inventaire : où sont listés les personnages et les objets (« les lutins »), et les arrière-plans,
- La zone de programmation : où l'on réalise notre programme.

Zone de rendu

Zone de programmation



Inventaire des lutins

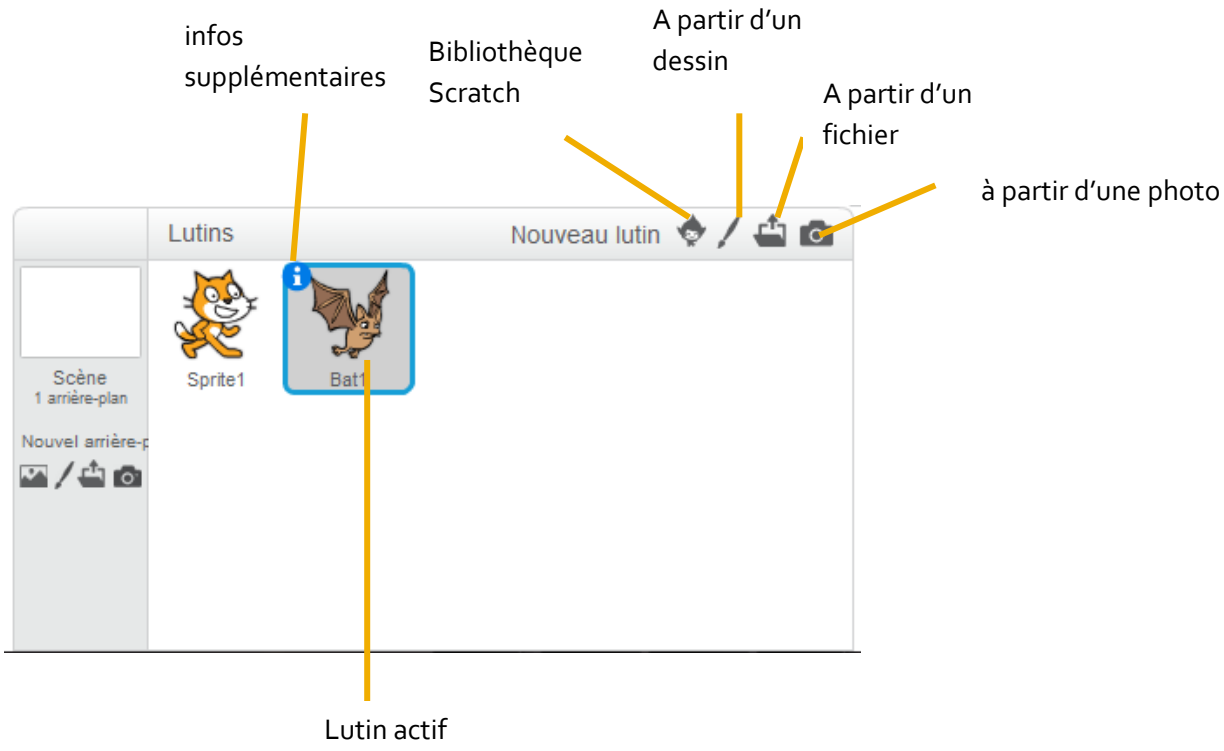
Pour commencer, pensez à mettre votre interface en français en cliquant sur la sphère en haut à gauche.

3. *Lutins*

Les lutins sont les personnages et les objets que vous pouvez programmer. Un lutin ne voit et n'exécute que les programmes qui ont été écrit sur SA page de script (quand l'image en haut à droite de la zone de script représente le lutin en question).

En revanche quand un évènement arrive (Appui sur le drapeau vert, appui sur espace..) tous les lutins le voient et tous les lutins peuvent potentiellement réagir à cet évènement.

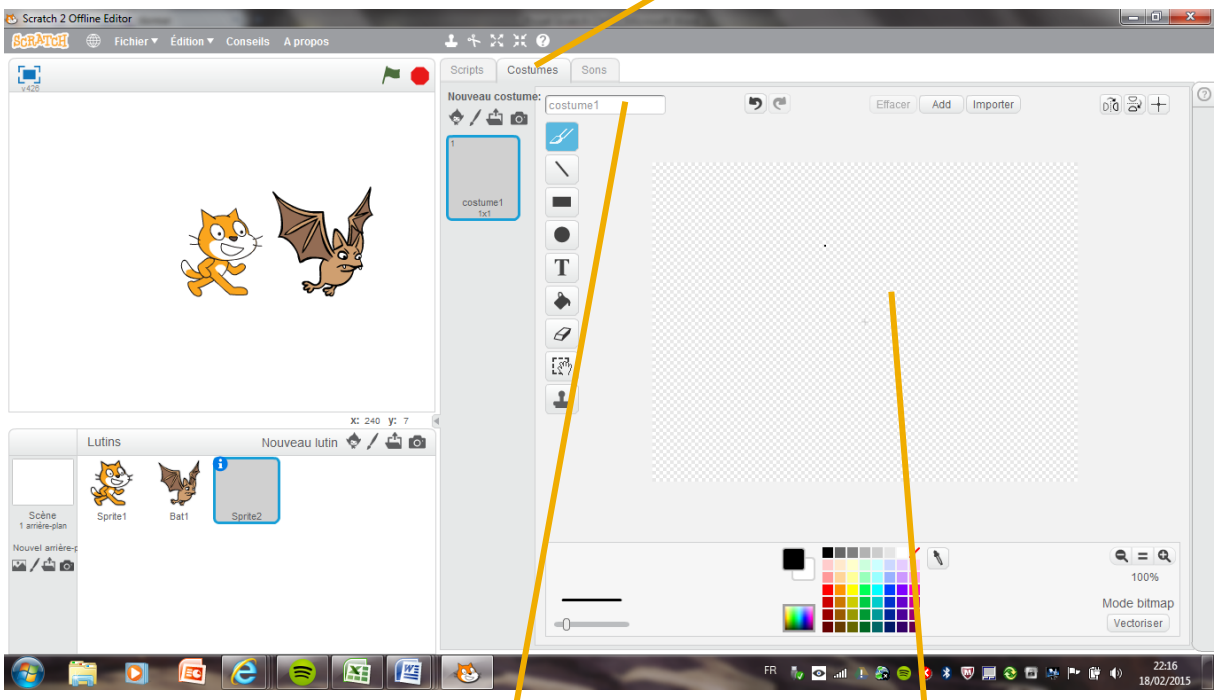
Les lutins sont gérés dans la fenêtre « inventaire ». C'est dans cette fenêtre que l'on peut ajouter/supprimer des lutins.



Lorsque l'on clique sur le bouton « i » des informations supplémentaires sur le lutin apparaissent.

Lorsqu'on clique sur « dessiner un lutin », la page de script est remplacée par une page de dessin, et l'onglet devient « costumes »

Onglet
« Costumes »



Nom du nouveau lutin

Zone de dessin

4. *Arrière plans*

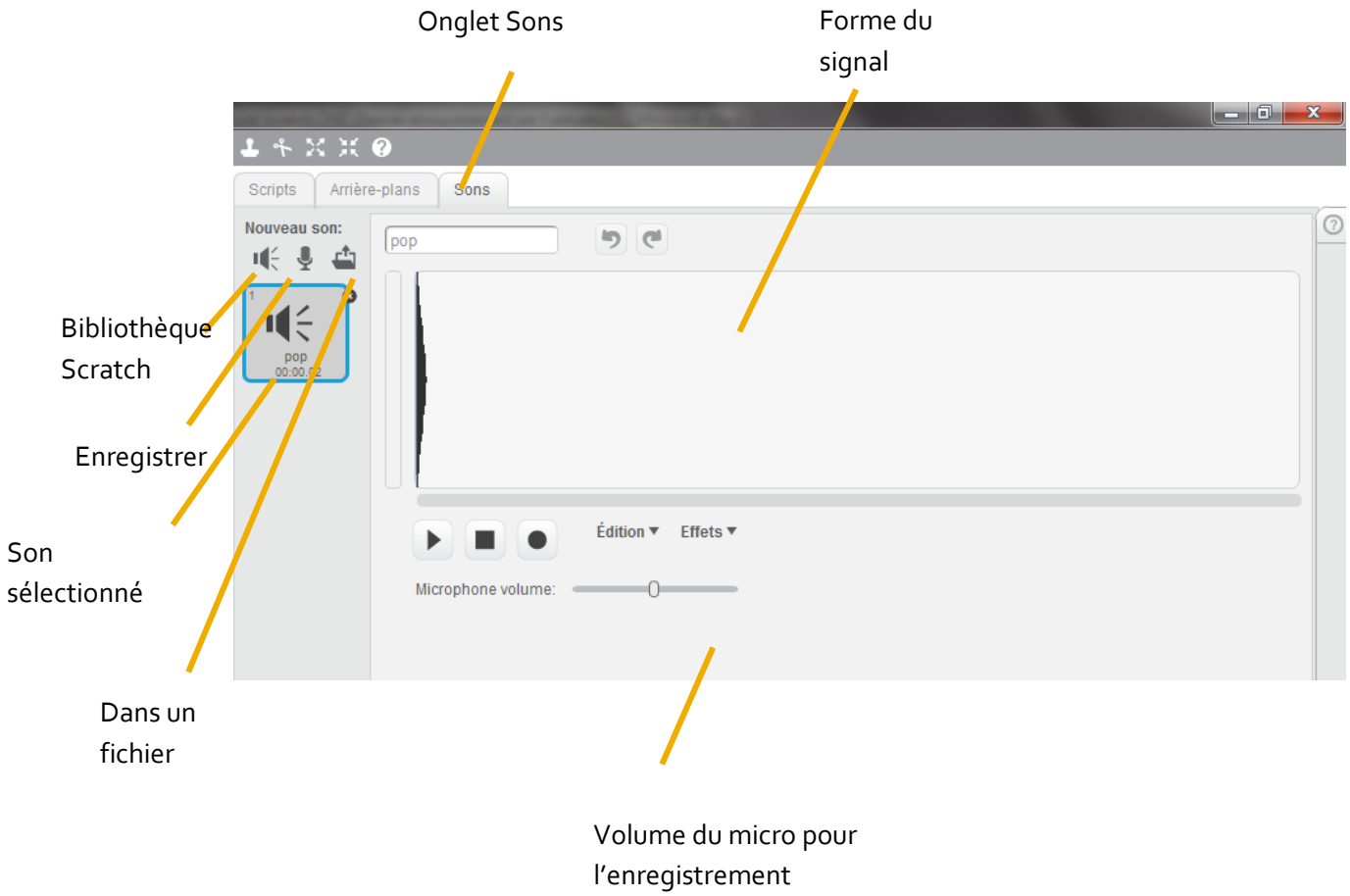
Les arrière-plans sont les décors des scènes. Comme les lutins ils sont programmables, et reçoivent également les évènements déclenchant les scripts.

Des icônes identiques à celles des lutins permettent d'ajouter des nouveaux arrières plans.

5. *Sons et musiques*

Il est possible d'associer aux lutins et aux arrière-plans des sons et de musiques. Ces sons, comme les programmes, sont associés à un lutin ou à un arrière-plan. Les autres lutins ou arrière-plans ne les voient pas.

Cette fonctionnalité se trouve dans l'onglet « sons », le troisième onglet situé à droite de « costumes » ou « arrière plans ».



On peut y voir la forme du son (de son signal plus exactement), les boutons pour le lire, des menus pour modifier le son (le couper, lui rajouter des effets).

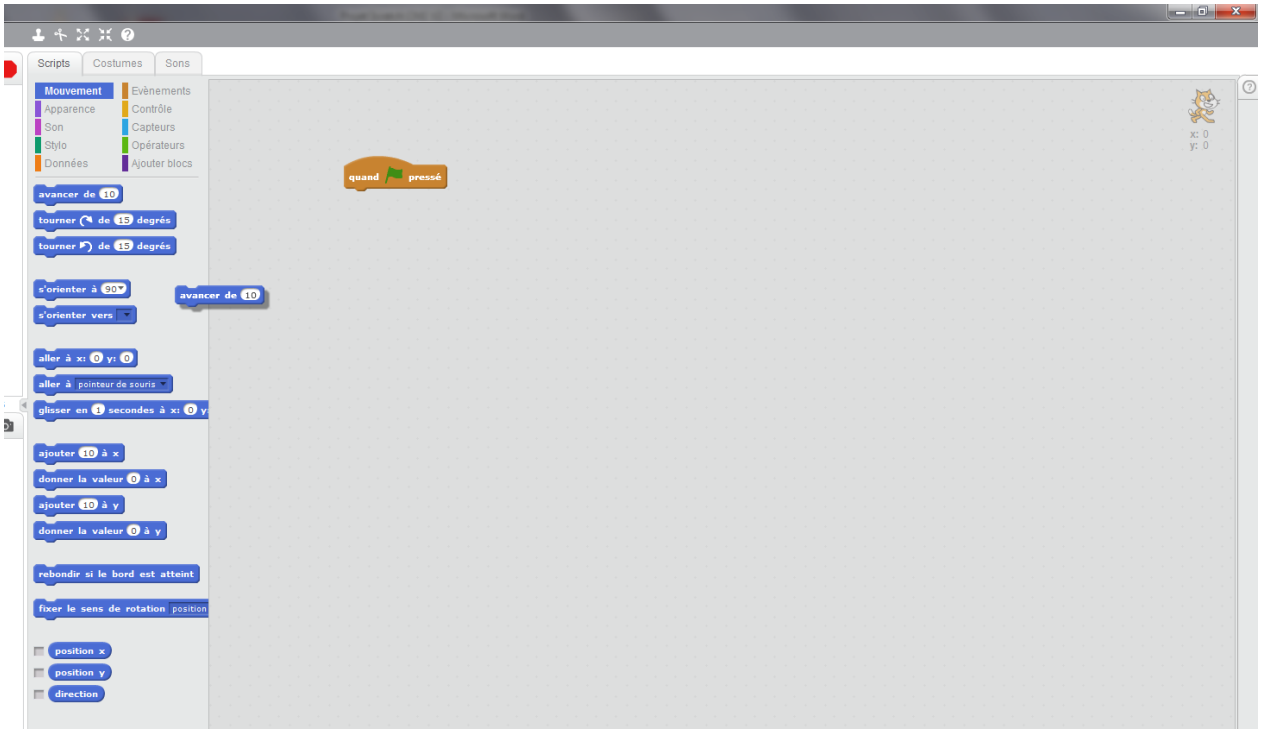
De la même manière que pour les lutins, on peut trouver des sons dans la bibliothèque scratch ou choisir un son dans un fichier sur l'ordinateur.

On peut également enregistrer un son directement avec le bouton « micro ».

6. *Programmation et fonctionnement des blocs*

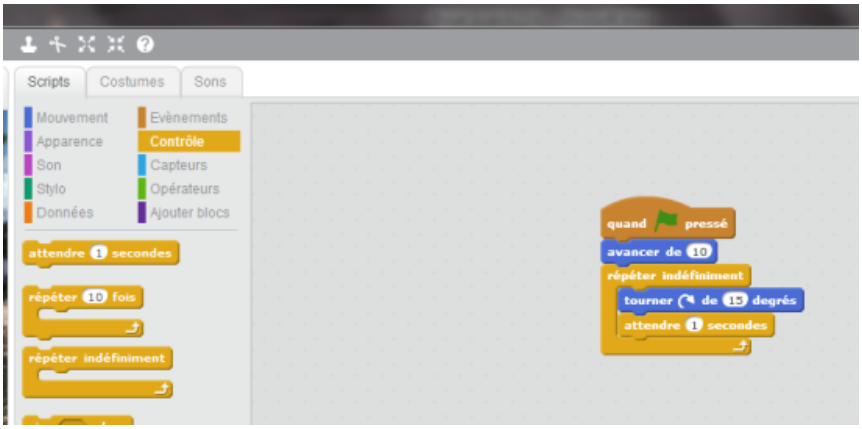
Les blocs de programmation sont disponibles dans l'onglet « scripts » pour chaque lutin. On peut identifier le lutin que l'on programme en regardant le dessin en haut à droite de la zone de programmation.

Les blocs se positionnent sur la zone de script par simple glisser-déposer. Dans l'exemple ci-dessous, nous programmons le chat (regardez, c'est bien lui en haut à droite), et nous sommes en train de faire glisser le bloc « avancer de 10 ».



Les blocs s'enclenchent les uns à la suite des autres comme un puzzle.

Vous pouvez mettre autant de scripts que vous le souhaitez sur la page de script d'un lutin.



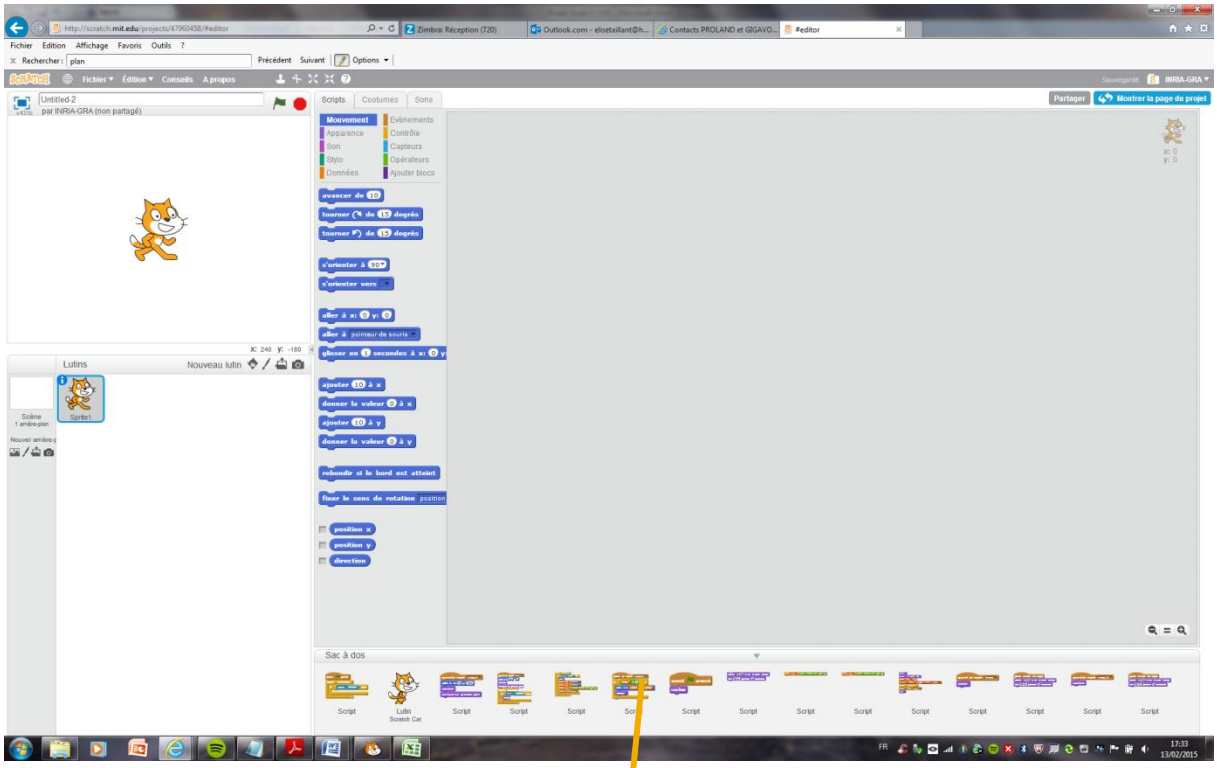
Un script doit toujours commencer par un évènement. C'est ce qui permettra de lancer le script. Les instructions seront ensuite exécutées dans l'ordre dans lesquelles vous les avez mises.

Point important, tous les scripts, de tous les lutins s'exécutent en même temps (ou plutôt parallèlement), donc si tous les lutins ont un script qui démarre avec le même évènement, alors tous ces scripts s'exécuteront en même temps.

On peut copier un script d'un lutin à l'autre en le glissant sur l'image du lutin destinataire.

7. *Sac à dos*

Le sac à dos est disponible uniquement sur la version en ligne si l'on possède un compte Scratch. Il permet de conserver des scripts et des lutins (accompagné de ses scripts) créés dans un projet et de les utiliser dans un autre projet.

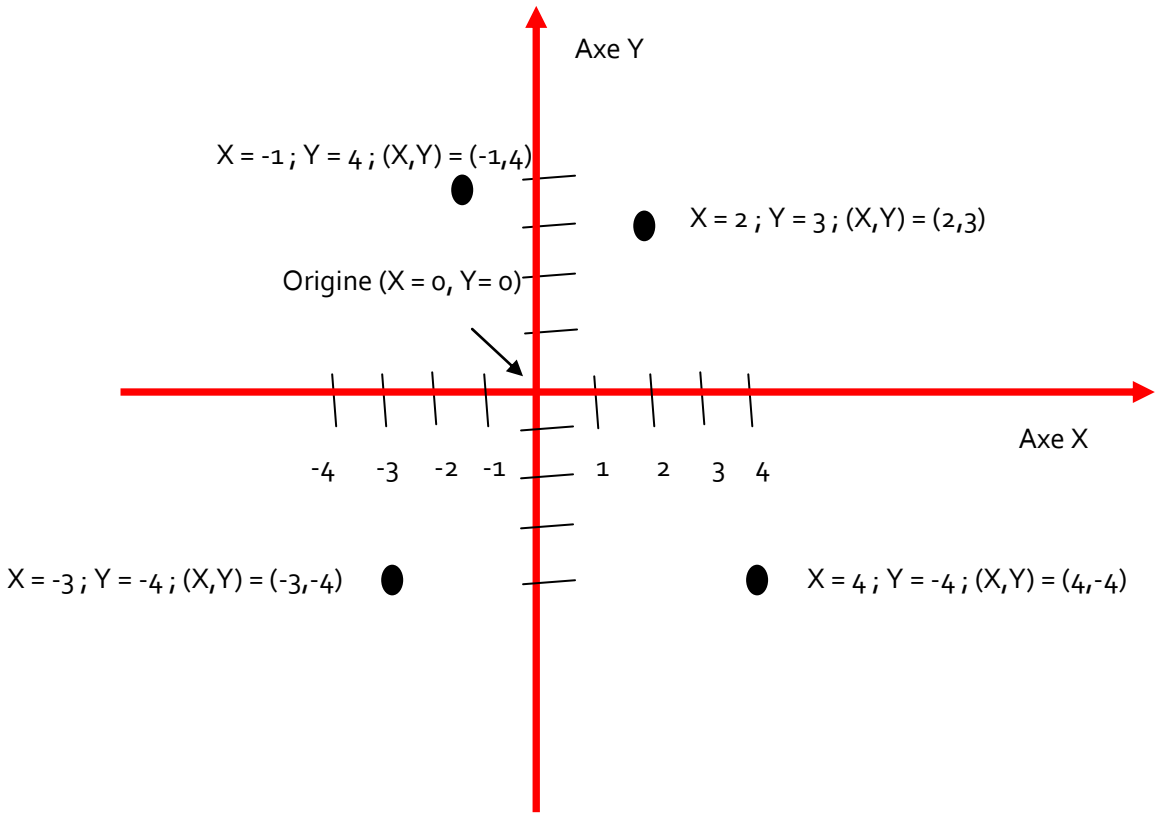


Sac à dos rempli de scripts

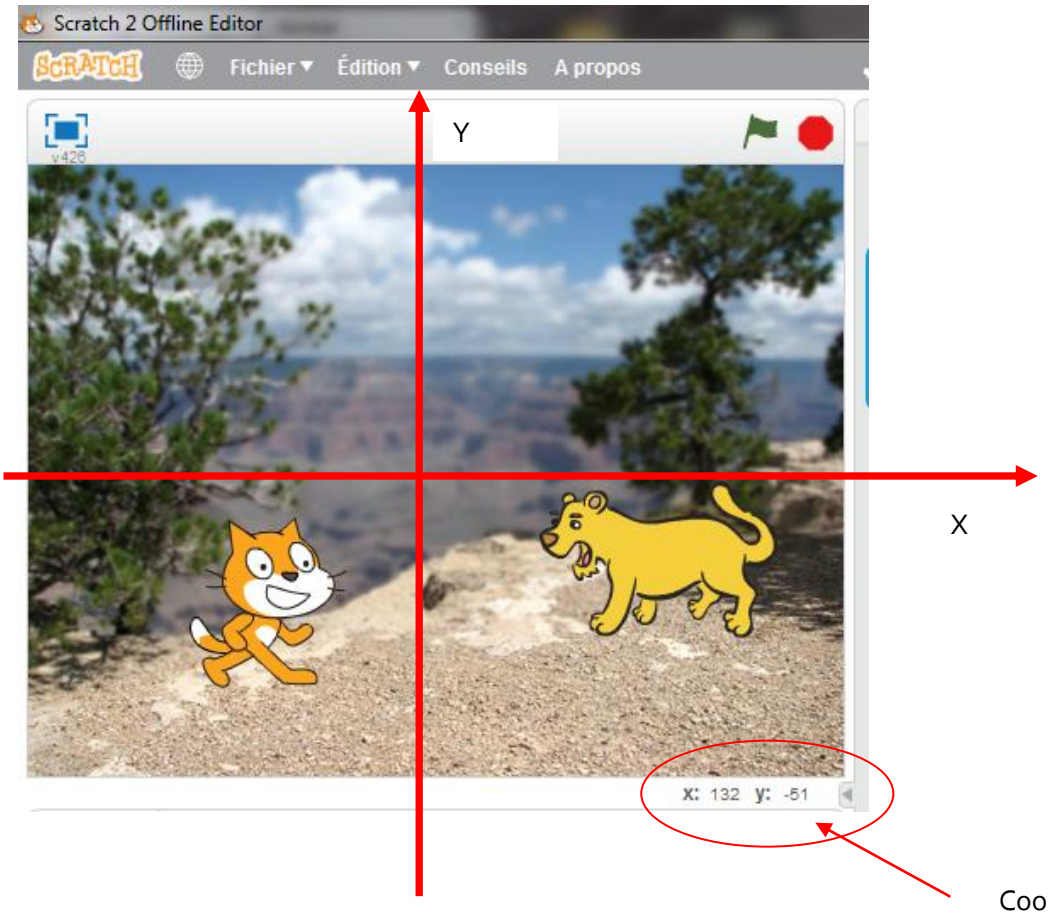
B) Notions importantes dont vous aurez (probablement) besoin

1. *Coordonnées dans le plan*

Les lutins et objets dans la scène sont repérés par leurs coordonnées cartésiennes. Les coordonnées cartésiennes permettent de repérer un point dans un plan selon un axe horizontal X et un axe vertical Y.



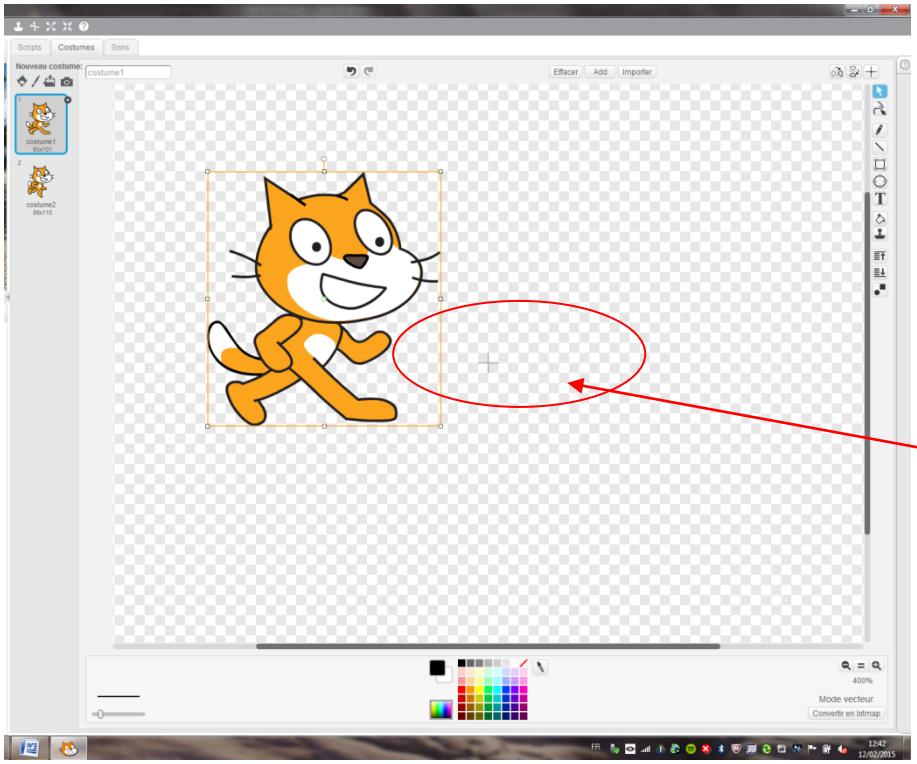
Dans la scène, l'origine (X=o, Y=o) est au centre de l'image. Tout ce qui est au dessus de l'axe horizontal correspond à un Y positif, en dessous à un Y négatif. Tout ce qui est à droite de l'axe vertical correspond à un X positif, à gauche à un X négatif.



Coordonnées de la souris

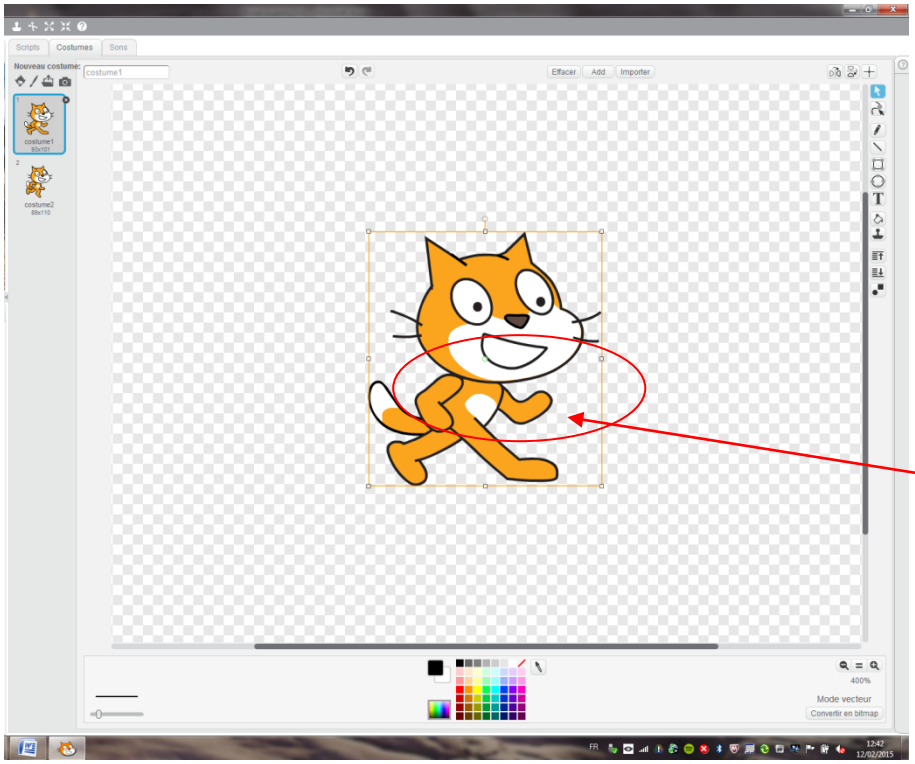
Les coordonnées de la pointe de la souris sont affichées en bas de la scène.

Le lutin, lui, est repéré par rapport à son « centre ». Pour voir ou celui-ci est situé, allez dans l'onglet « Costume » de votre lutin. Le centre du lutin est la croix grise.



Centre du lutin pas très centré

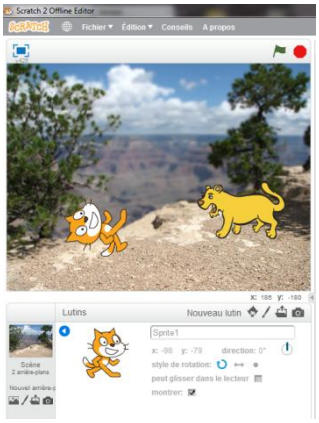
Pour plus de facilité il vaut mieux réellement centrer son lutin sur la croix.



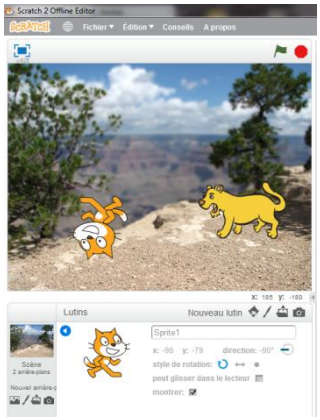
Lutin centré sur son centre

2. *Orientation et rotations*

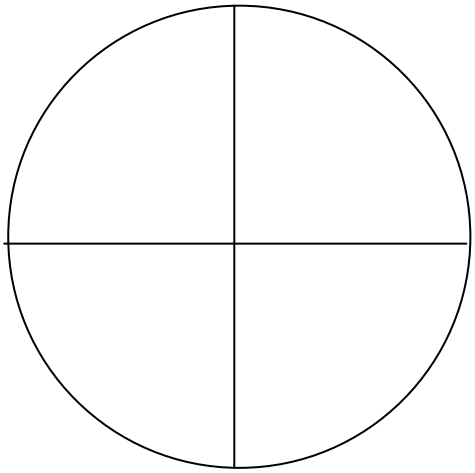
Vous pouvez orienter vos lutins. Les orientations sont repérées par un angle selon le schéma ci-dessous (regardez bien l'orientation du chat dans l'image) :



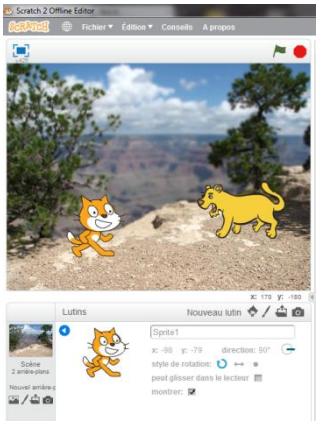
0°



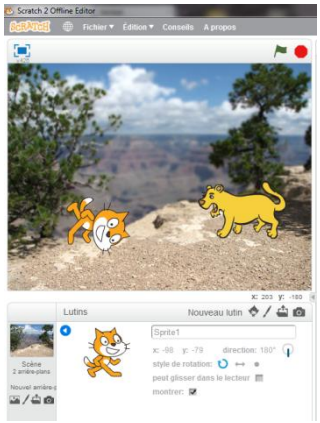
-90°



90°



180°

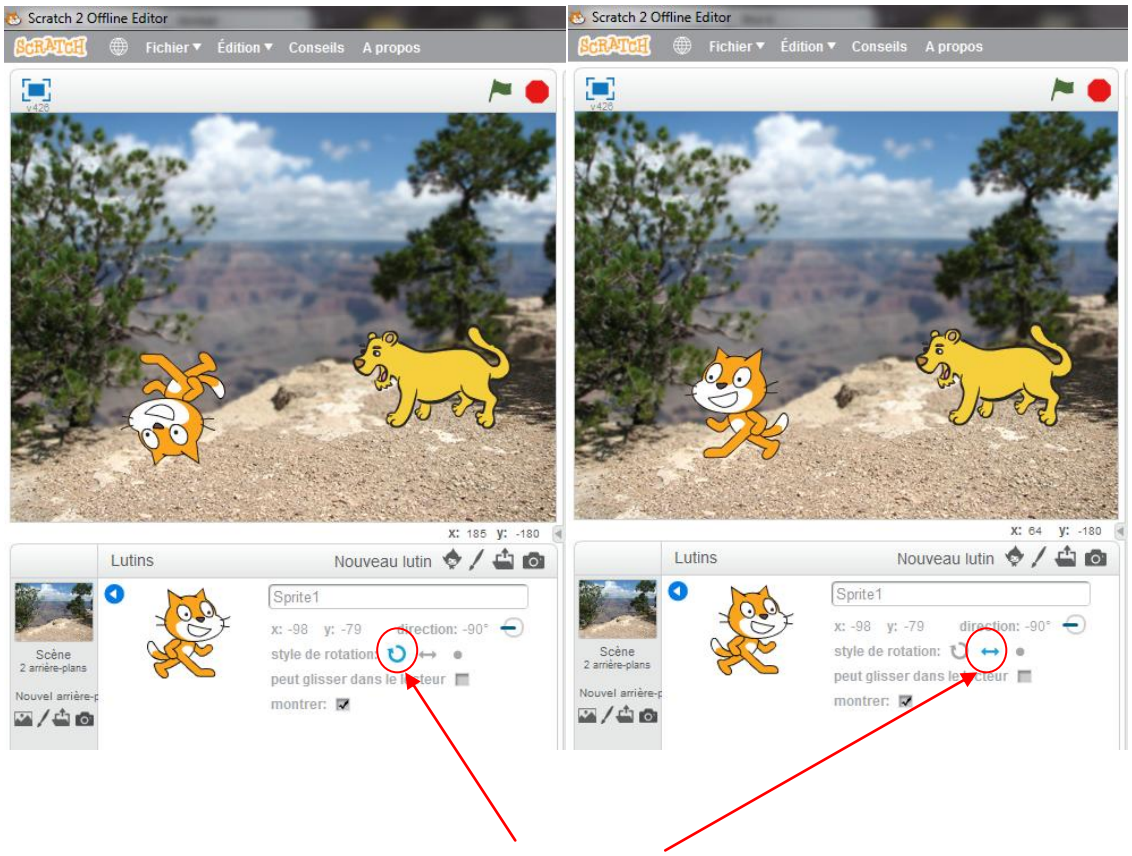


Vous vous poserez peut être dans quelques années la même question que nous, à savoir pourquoi avoir choisi les orientations comme ça, car c'est assez curieux… Nous aurions plutôt mis le 0° à droite, le 90° en haut, le 180° à gauche et le -90° en bas mais c'est une autre histoire (pour les curieux tapez « cercle trigonométrique » sur votre moteur de recherche)

Quand vous demandez à votre chat de tourner de 15°, il va donc tourner dans le sens des aiguilles d'une montre de 15°. Si vous lui dites de tourner de -15° il tournera dans le sens inverse des aiguilles d'une montre (pour les mathématiciens, ça s'appelle le sens trigonométrique).

Vous remarquez que quand on demande au chat de s'orienter vers la gauche (-90°), il se retrouve la tête en bas. C'est ennuyeux. Pour éviter cela, il faut contraindre les mouvements du chat. Cliquez sur le petit i au niveau de votre lutin et dans « style de rotation » choisissez ◄►

Miracle le chat a de nouveau la tête en haut.



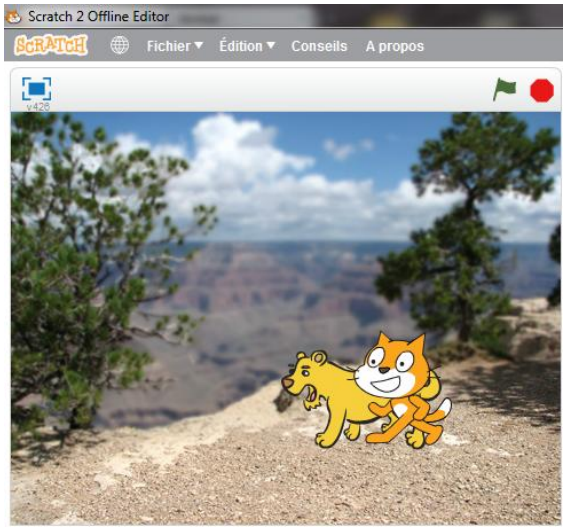
La différence est la !

Vous pouvez aussi contraindre le style de rotation directement dans le programme en utilisant le bloc bleu

3. *Plans*

Les lutins et les objets dans Scratch sont positionnés sur plusieurs plans ou couches qui se superposent.

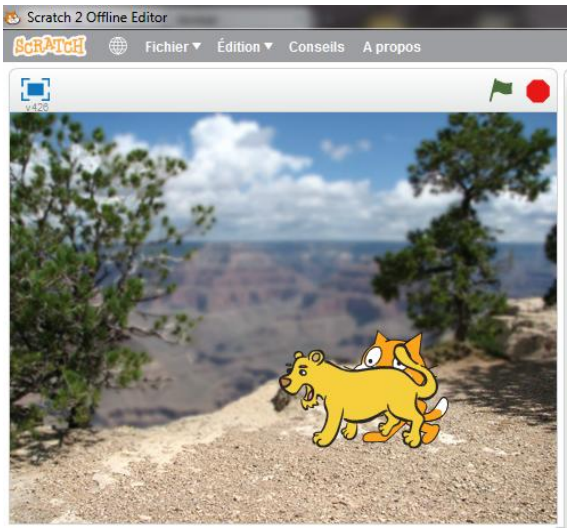
Typiquement, l'arrière plan est derrière, comme son nom l'indique. Sur l'image ci-dessous le chat est au premier plan, la lionne au second, et l'arrière plan…toujours en dernier.



Evidemment ces histoires de plans n'interviennent que quand deux lutins se croisent. Il faut choisir lequel cache l'autre.

Pour modifier le positionnement des lutins il suffit d'utiliser, dans la zone de script, les deux instructions suivantes, que vous trouvez dans les blocs « Apparence » :

Dans l'exemple ci-dessous, on a utilisé « Envoyer au premier plan « sur la lionne, et la voila devant le chat !



4. *Conditions*

Les conditions permettent de tester des variables. Les conditions commencent par un « Si », sont suivies d'un « alors » et quelquefois d'un « sinon ». On utilise tout le temps des conditions dans le langage courant :

« Si il fait beau alors je vais skier»

« Si il fait beau alors je vais skier sinon je reste chez moi »

« Si il fait plus de 25°C alors je mets un short sinon je mets un pantalon »

Dans le premier cas on ne précise pas ce qu'il se passe s'il ne fait pas beau.

On teste ici la variable « météo » ou température, En langage de programmation on pourrait l'écrire :

Si météo = beau alors va au ski sinon reste à la maison.

Si température > 25°C alors mets un short sinon mets un pantalon.

On peut évidemment imbriquer plusieurs niveaux de conditions :

Si température > 30°C alors mets un maillot de bain

sinon si température > 20°C alors mets un short

sinon mets un pantalon.

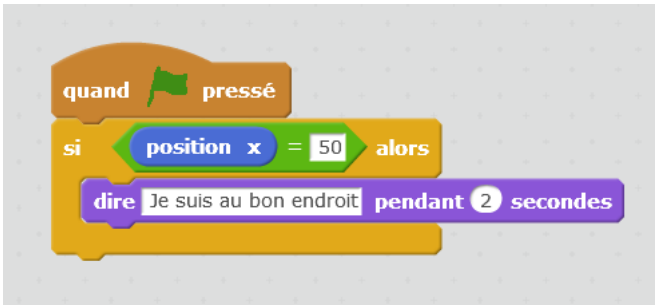
On trouve les blocs de conditions dans l'onglet « Contrôles »

Les cases hexagonales vides sont censées être remplies par un test(Sinon on ne teste rien du tout.)

Pour ceux qui ont un doute : un hexagone est un polygone à 6 côtés.

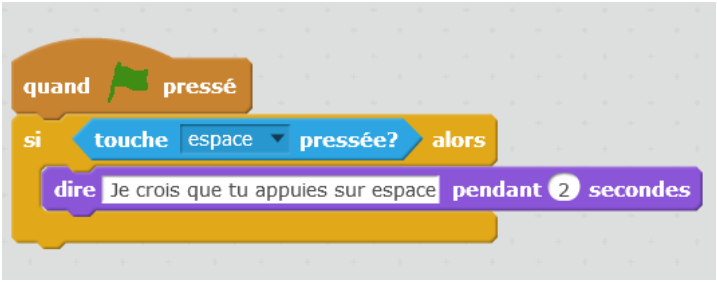
Vous trouverez deux types de tests possibles dans Scratch :

- Des opérateurs (voir le chapitre qui y est consacré plus bas), qui permettent notamment de tester des valeurs :



Ici on test que la position sur l'axe X du lutin est 50, si c'est le cas, le lutin dit « Je suis au bon endroit »

- Des capteurs qui détectent les interactions du joueur avec le jeu ou des lutins entre eux.



Quand le jeu démarre, si la touche « espace » est pressée, le lutin nous en fera part.

5. *Boucles*

Les boucles sont extrêmement utiles en programmation, en effet elles permettent de faire une série d'instructions plusieurs fois sans avoir à écrire plusieurs fois cette série d'instruction. C'est pratique car en bon informaticien fainéant, ça fait moins de travail à faire et puis, ce qui ne gâche rien, le programme est plus lisible. Vous trouverez les blocs « boucles » dans l'onglet « contrôles ».

Prenons quelques exemples. Les trois scripts ci-dessous donnent le même résultat :

Le lutin va avancer de 50 pas. (En pratique, au moment de l'exécution, vous noterez une légère différence entre les deux premiers scripts et le troisième).

Dans ce cas, évidemment, le second script est le plus « intelligent » car il fait avancer de 50 pas en une seule instruction. En revanche, si on considère maintenant le script suivant avec deux instructions répétées :

On se rend compte que :

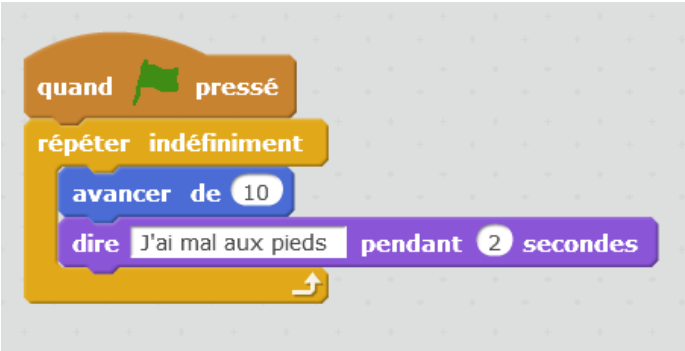
1/ Ce n'est pas très futé

2/ On ne peut plus faire le script N°2.

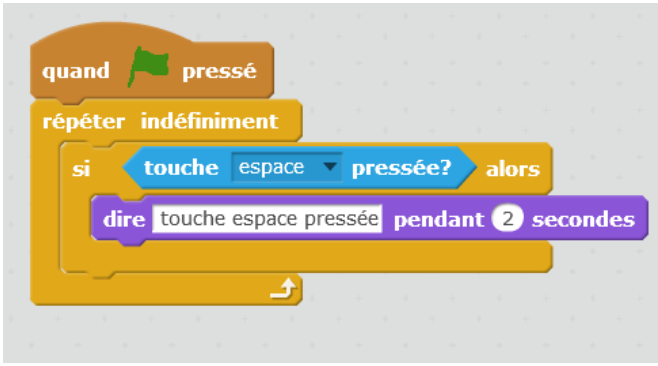
Et là, la boucle devient vraiment intéressante :

Il existe deux autres types de boucle très utiles : la boucle dite « infinie » et la boucle avec condition de fin.

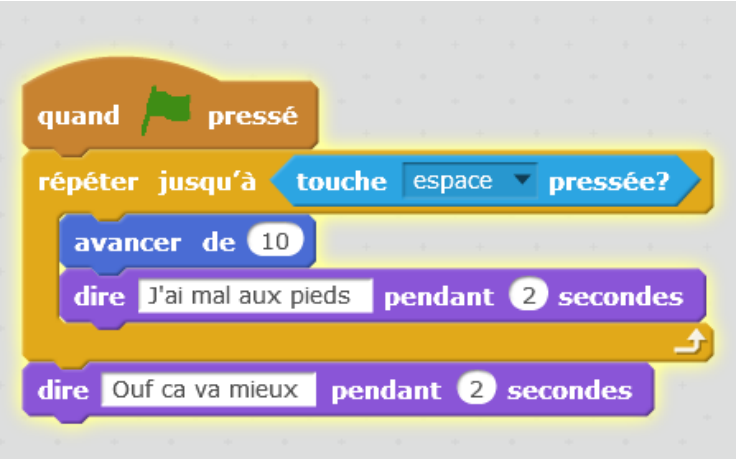
La boucle infinie sert à faire les mêmes instructions en permanence sans indication de fin :



Le lutin va avancer de dix pas et dire « j'ai mal aux pieds » indéfiniment, enfin, jusqu'à temps qu'on arrête l'animation. Ce type de boucle sert par exemple à faire défiler des personnages, ou encore à tester de manière répétée si une action a eu lieu. Dans l'exemple ci-dessous, le lutin dira « touche espace pressée » quand j'appuie sur la touche espace, et ne dira rien quand elle n'est pas pressée (oui, ce programme ne sert à rien) :



Enfin la boucle avec condition de fin permet d'exécuter de manière répétée plusieurs instructions jusqu'à ce qu'un évènement arrive :



Dans cet exemple, le lutin continue sa marche forcée en se plaignant de ses pieds, jusqu'à ce qu'un humain bienveillant le libère (on sort alors de la boucle) en pressant la touche « espace ». Il nous dit alors que ça va mieux.

Notez bien la différence entre les deux derniers exemples : même si la condition (touche espace pressée) est la même, dans la boucle infinie le fait de presser « espace » n'arrête pas la boucle, dans le cas de la boucle avec condition, c'est justement l'appui sur « espace » qui arrête la boucle.

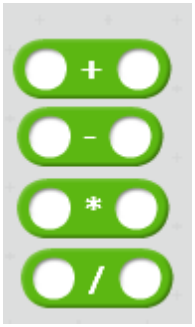
6. *Opérateurs*

Vous trouvez les blocs « opérateurs » dans l'onglet opérateurs (vert clair). Jusque là tout va bien.

Vous remarquerez qu'il y a deux formes de blocs, des hexagonaux et d'autres avec les coins arrondis.

Commençons par les plus évidents, les blocs arrondis sont ceux qui permettent de réaliser des opérations :

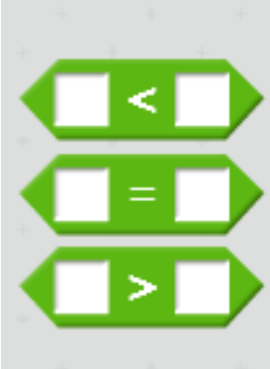
- celles qui s'appliquent à des nombres et que vous connaissez : addition, soustraction, multiplication, division



- celles qui s'appliquent à des mots ou suites de lettres :
  - regrouper ou concaténer deux mots : par exemple regroupe « bonjour » « monsieur » donne « bonjourmonsieur »
  - trouver la lettre numéro n d'un mot
  - compter le nombre de lettres dans un mot

Les blocs hexagonaux sont utilisés dans les conditions (voir chapitre conditions), ils permettent de réaliser des tests :

- comparer des valeurs : égalité, supérieur, inférieur.



- les opérateurs logiques : et, ou, non

Arrêtons-nous quelques instants pour bien comprendre ces opérateurs en reprenant les exemples déjà vu avec les conditions (voir plus haut) :

- L'opérateur ET permet de tester que deux choses sont exactes. Par exemple : » Si il fait beau ET que c'est le weekend alors je vais skier ».
  On ira skier que si les deux conditions sont réunies, il fait beau et c'est le weekend.
- L'opérateur OU permet de tester qu'au moins une des conditions est vérifiée. Par exemple : « Si c'est le weekend OU les vacances alors je peux faire la grasse matinée ». Bien sûr si les deux sont vraies alors on peut dormir aussi.
- L'opérateur NON permet de tester le contraire. Par exemple on veut écrire :

« Si il ne fait pas beau alors je ne vais pas skier ».

C'est possible avec l'opérateur NON :

« Si NON il fait beau alors je ne vais pas skier » ou encore

« Si NON météo= beau alors je ne vais pas skier »

7. Variables

Autre élément extrêmement important en programmation : les variables.

Les variables permettent de stocker une valeur qui pourra être utilisée par le programme. Cette valeur peut changer, d'où le nom « variable ».

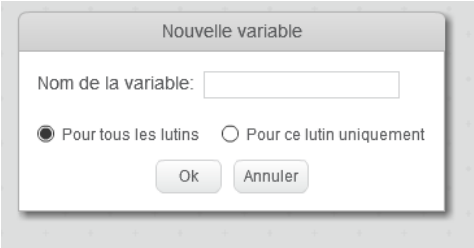
Voici quelques petits exemples pour bien comprendre comment on peut s'en servir :

Imaginons que notre programme soit un jeu de football : il y a donc deux équipes avec chacune un score représentant le nombre de buts marqués par cette équipe. Les scores sont des variables. Appelons les ScoreBleu et ScoreRouge. A chaque instant du match nous sommes capables de connaitre la valeur du score d'une des équipes, et à chaque but nous pouvons changer ce score en ajoutant 1.

On peut aussi se servir des variables pour connaître l'état d'une chose. Reprenons notre exemple sur la décision d'aller au ski ou non : la variable est la météo. On peut lui attribuer la valeur « beau » ou la valeur « moche » ou plus simplement 0 ou 1.

Enfin on peut utiliser une variable pour stocker le résultat d'une opération : par exemple la variable nombre_de_but est l'addition de ScoreBleu et ScoreRouge.

Dans Scratch, vous pouvez créer des variables en allant dans les blocs « données » et en cliquant sur « créer une variable ».



Vous devez lui donnez un nom. Essayez de donner un nom clair, afin de pouvoir vous repérer plus facilement quand vous souhaitez l'utiliser. Evitez par exemple « toto » ou « abcd ».

On vous demande également si c'est une variable pour tous les lutins ou pour ce lutin seulement.

Pour savoir quelle option choisir, il faut savoir si plusieurs lutins du jeu doivent voir cette valeur ou si seulement le lutin concerné en a besoin. Dans le cas du jeu de football, par exemple, il peut être utile de comparer les deux scores, les deux équipes et l'arbitre doivent pouvoir connaitre les deux scores, la variable est donc partagée.

En pratique, si on ne sait pas quoi choisir il vaut mieux que tous les lutins puissent voir cette variable.

Une fois la variable créée vous voyez que de nouveaux blocs sont apparus:

Le premier cloc est le bloc « score », c'est la variable. On utilise ce bloc pour réaliser des opérations.

Le second « Mettre score à » permet de lui attribuer une valeur : un chiffre, une suite de lettre, le résultat d'une opération.

Le suivant « Ajouter à score » permet d'additionner une valeur. Attention au résultat si votre variable contient du texte.

Les deux derniers permettant d'afficher cette variable dans la scène ou de la cacher (utile dans le cas du match de foot !).

Attention à l'initialisation de votre variable, dans Scratch elle contient par défaut la valeur 0. Dans d'autres langages de programmation on ne sait pas ce qu'elle contient quand elle est créée…

Pensez bien à initialiser votre variable si vous ne voulez pas que sa valeur initiale soit 0.

8. *Listes*

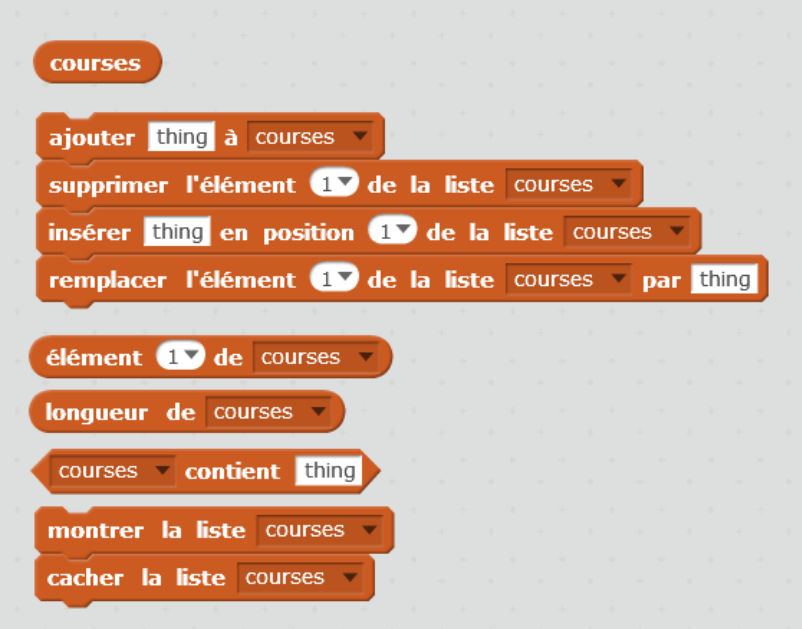
Les listes, également très utiles, sont des variables qui regroupent des éléments ayant un lien entre eux. Ces éléments peuvent être des valeurs, des variables, ou elles-mêmes des listes.

Par exemple :

- la liste de courses est une liste comportant les valeurs : chocolat, tomates, poulet, jus d'orange…
- la liste menu comporte les variables entrée, plat, dessert. Ces variables pouvant être modifiées.
- la liste élèves de l'école comporte les listes CP, CE1, CE2, CM1, CM2, elles mêmes des listes de nom d'élèves.

Les éléments de la liste sont repérés par leur position dans la liste (en commençant à 1)

Comme pour les variables ci-dessus vous trouverez les listes dans les blocs « données ». Comme pour les variables des nouveaux blocs apparaissent quand vous créez pour la première fois une liste, ici la liste courses.



Le premier bloc est le bloc « courses », c'est la variable. On utilise ce bloc pour réaliser des opérations.

Le second « ajouter thing à courses » permet de lui ajouter un élément (en remplaçant « thing ») en dernière position : un chiffre, une suite de lettre, le résultat d'une opération, une variable, une autre liste.

Le suivant « Supprimer l'élément N de la liste » permet de retirer l'élément à la position N dans la liste, vous comprendrez facilement le suivant qui est d'insérer un élément en position N

Le 6ème bloc permet d'obtenir l'élément en position N

Le 7ème bloc permet de connaitre la longueur de la liste, si celle-ci est vide sa longueur est o.

Le 8ème bloc permet de savoir si la liste contient un élément.

9. *Messages et synchronisation*

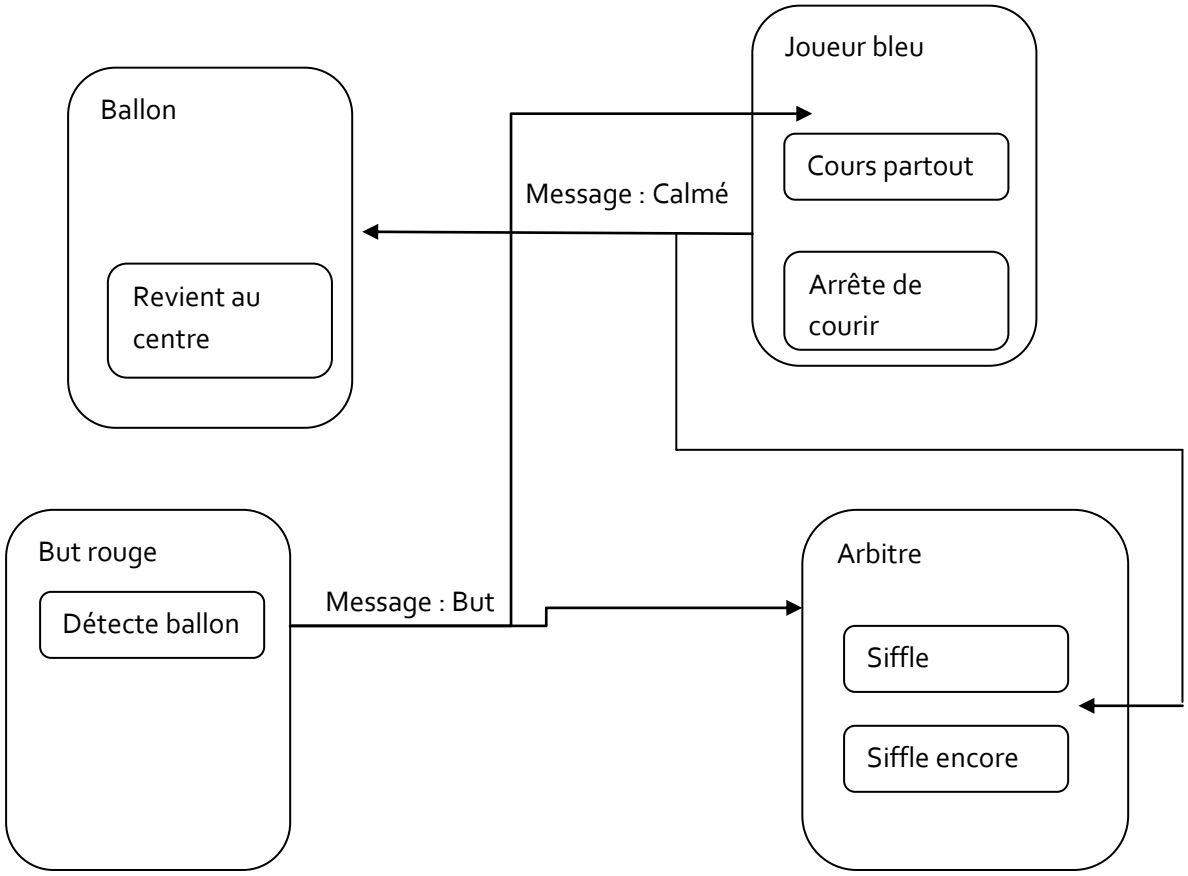
Comme nous l'avons vu, les lutins ont leur monde bien à eux. Ils ne savent pas ce que font les autres lutins et n'ont pas d'interaction sauf éventuellement lorsqu'ils se touchent.

Afin de pouvoir communiquer entre eux (et également avec l'arrière plan, qui est un lutin bien particulier), on utilise des messages. Ces messages nous permettent de synchroniser et déclencher des actions.

Reprenons notre exemple de jeu de football : nous avons 4 lutins pour cet exemple :

- 1 joueur bleu
- 1 arbitre
- 1 ballon
- 1 but rouge.

On veut que lorsque le ballon entre dans le but, l'arbitre siffle, et que le joueur bleu se mette à courir dans tous les sens (c'est ce qu'ils font en vrai non ?).Puis quand il est calmé, l'arbitre doit siffler de nouveau et le ballon revenir au centre du terrain.



Vous trouverez les messages dans la rubrique « Evènements ».

Vous pouvez envoyer des messages et recevoir des messages. Quand un lutin envoie un message, tout le monde peut le voir, mais tout le monde n'est pas obligé de réagir. Vous pouvez créer autant de message que vous voulez, encore une fois, essayez d'être explicite quand vous créez le message, pour autant n'écrivez pas un roman. Pour créer un nouveau message, dans le bloc « envoyer un message » cliquez sur la liste déroulante puis sur » nouveau message ».

C) Fiches

Exemple de présentation du scénario

SITUATION	ACTIONS
◦ Le lieu ◦ Les objets présents et les actions possibles sur ces objets ◦ Les personnages présents	◦ Les actions des personnages ◦ Les dialogues ◦ Ce que doit faire le héros (pour passer à la suite)

Organisation du développement

Exemple de tableau de suivi des tâches

	Tâches	A faire	En cours	Fait
1 - Création personnages				
1.1	Création héros			
1.2	Création ami du héros			
2 - Création scène 1				
2.1	Dessin arrière plan			
2.2	Script dialogues			
2.3	Interactions avec le joueur			