

Jeu du pendu



Présentation

Le joueur doit retrouver un mot pris au hasard, en indiquant successivement les lettres qu'il pense contenir afin de le reconstituer. Il n'a droit qu'à 6 erreurs, sans quoi un malheureux cow-boy sera pendu !

Objectifs pédagogiques

Ce projet aura pour but d'écrire un programme en travaillant :

- les boucles (répétitions)
- les structures conditionnelles (si... alors...sinon...)
- les variables
- les listes
- les chaînes de caractères (*)

Organisation et évaluation

Vous travaillerez seul(e) ou en binôme.

Chaque groupe disposera :

- du présent **guide** explicatif,
- d'un **programme Scratch** de démarrage,
- d'un ensemble de fichiers présents dans le dossier « **Compléments élèves** », et qui pourront être utiles pour la réalisation des objectifs complémentaires.

L'évaluation de votre travail tiendra compte :

- du bon fonctionnement du programme
- des éléments de développement personnalisés que vous aurez ajoutés
- de l'implication de chacun (= évaluation individuelle, pas en binôme)

Les meilleurs projets pourront être publiés sur le site du collège.

A chaque fin de séance, enregistrez votre travail sur :

- la zone de partage de données de la classe
- la zone personnelle de chacun (important en cas de problème sur la zone de partage)
- votre clé USB


(*) Chaîne de caractère = mot formé par une succession de lettres, chiffres, ou autres caractères

Pour chaque début de séance, il vous appartient d'apporter votre travail sur clé USB et/ou par mail et/ou en l'ayant déposée préalablement sur votre zone personnelle au collège.

Durée

Ce projet s'étendra sur 6 séances et le travail devra être rendu à l'issue de la dernière séance (compter 1 séance pour chacun des 4 objectifs principaux, et 2 séances pour les objectifs secondaires).

Scénario

Au clic sur , un mot est choisi aléatoirement par le programme (la liste de mots est incluse dans le programme Scratch de démarrage).

Abby annonce qu'un mot est à découvrir en moins de 7 erreurs.

Ce mot s'affiche, constitué de « _ » représentant les lettres à trouver.

On demande au joueur une lettre.

Si la lettre est correcte, elle s'affiche à la place du « _ » qui la représente, sinon la **Potence** est dressée progressivement.

Puis on redemande une lettre jusqu'à ce que le joueur ait dévoilé toutes les lettres du mot à deviner (partie gagnante) ou que la **Potence** soit complétée (partie perdante).

Si la partie est perdante, **Abby** annonce le mot qu'il fallait trouver.

Réalisation

Vous réaliserez ce programme en menant à bien 4 objectifs principaux successifs présentés dans la suite de ce document.

Pour compléter ce travail une fois les 4 objectifs principaux réalisés, vous choisirez un ou plusieurs objectifs secondaires parmi ceux proposés dans le sous-dossier « Objectifs secondaires » : le niveau de chaque objectif est indiqué (bronze, argent ou or).

Données

Vous aurez besoin de 5 variables et de 4 listes :


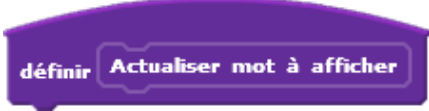


- **Erreurs** *Variable chaîne de caractères*
Contient toutes les lettres proposées par le joueur et qui ne sont pas présentes dans le mot à deviner. Pour une meilleure lisibilité, les lettres sont séparées par des espaces. Cette variable s'affiche dès la première erreur.
- **Fin du jeu** *Variable « oui » ou « non » ([booléen](#))*
Indique l'état de la partie. Elle doit être initialisée à « non » au début de la partie. On cesse de demander une lettre au joueur quand sa valeur passe à « oui ».
- **Mot affiché** *Variable chaîne de caractères*
Comme son nom l'indique, représente le mot à trouver tel qu'il est affiché au cours de la partie. Initialement uniquement composé de « _ _ _ _ » (alternance de tirets du bas et d'espaces), il se complète avec les lettres du mot à deviner progressivement. Cette variable est visible tout au long de la partie. Pour une meilleure lisibilité, son affichage a été placé en mode « Grande lecture » (mode accessible dans la scène par clic droit sur la variable une fois visible).
- **Mot à deviner** *Variable chaîne de caractères*
C'est le mot choisi par le programme dans le Dictionnaire.
- **Numéro de caractère** *Variable numérique*
C'est un nombre qui indique le numéro du caractère dans une chaîne qu'on veut analyser caractère après caractère.
- **Alphabet** *Liste*
Cette liste contient tous les caractères que le joueur est autorisé à saisir durant le jeu. Le coup n'est pas considéré comme perdant si le joueur saisit autre chose que l'un de ces caractères. Cette liste n'est normalement pas modifiée durant le jeu.
- **Dictionnaire** *Liste*
Cette liste contient tous les mots que le programme pourra faire deviner, en choisissant l'un d'eux au hasard au début du jeu. Elle n'est normalement pas modifiée durant le jeu, mais peut être remplacée, en important par exemple un autre dictionnaire (dans la scène, clic droit sur **Dictionnaire** une fois visible).
- **Lettres déjà trouvées** *Liste*
Cette liste contient toutes les lettres gagnantes trouvées par le joueur. Elle doit être vidée en début de partie, puis complétée au fil des bonnes réponses. Elle permet de reconstituer le **Mot affiché** durant la partie en analysant chaque caractère de **Mot à deviner** et en regardant s'il se trouve dans cette liste auquel cas on affiche la lettre et non plus « _ ».
- **Lettres à découvrir** *Liste*
Cette liste contient toutes les lettres de mot restant à découvrir.
En début de partie, elle contient donc toutes les lettres du mot, et se vide au fur et à mesure que le joueur devine les lettres du **Mot à deviner**.
Ainsi, **Fin du jeu** vaut « oui » quand **longueur de** **Lettres à découvrir** vaut 0.

Deux lutins sont également fournis :

- **Abby** : c'est essentiellement le script d'**Abby** que vous allez programmer. Elle a trois costumes (neutre, souriante, triste) qui pourront être utilisés selon les circonstances du déroulement du jeu.
- **Potence** : contient les images (costumes) qui pourront être affichées successivement au fil des erreurs. Le premier costume est totalement vide et le dernier représente l'ultime étape, quand le joueur a perdu la partie.

Objectifs principaux

Votre travail va consister essentiellement à compléter successivement les 4 blocs :

<u>Objectif principal 1</u>	 A purple Scratch block with a tab labeled 'définir' and the text 'Initialisation'.	Dans ce bloc, on initialise listes et variables. Le Mot à deviner est pris aléatoirement dans le dictionnaire.
<u>Objectif principal 2</u>	 A purple Scratch block with a tab labeled 'définir' and the text 'Actualiser mot à afficher'.	Ce bloc, appelé au démarrage du programme et à chaque fois qu'une lettre du mot est découverte, permet de recalculer l'affichage de Mot affiché en haut de la scène.
<u>Objectif principal 3</u>	 A purple Scratch block with a tab labeled 'définir' and the text 'Déroulement du jeu'.	Ce bloc consiste à demander au joueur de façon répétitive une lettre et à l'analyser (coup gagnant ou coup perdant) avant de recommencer.
<u>Objectif principal 4</u>	 A purple Scratch block with a tab labeled 'définir' and the text 'Fin du jeu'.	Lorsque la partie est terminée, Abby affiche au joueur qu'il a gagné ou perdu, auquel cas elle lui annonce le Mot à deviner qu'il fallait découvrir.

⇒ Ne pas commencer l'objectif suivant tant que l'objectif en cours n'est pas terminé.



Objectif principal 1

Idee générale :

On initialise les listes et variables avant leur utilisation au cours du jeu.


Application :

On vide les deux listes **Lettres à découvrir** et **Lettres déjà trouvées** à l'aide de l'instruction **supprimer l'élément** **tout** de la liste **Lettres à découvrir** (sur Scratch 3, taper « all » à la place de « tout »).

On initialise également **Erreurs** (chaîne vide), et **Fin du jeu** (à « non ») à l'aide de **mettre** **à** **Fin du jeu**.

On appelle **cacher la variable** de **Mot affiché** ainsi que pour **Erreurs** qui ne seront rendues visibles que lorsque la partie aura commencé.

On prend un mot au hasard dans **Dictionnaire** et on le mémorise dans la variable **Mot à deviner**. Pour cela, on pourra utiliser les instructions : **élément** **de** **Dictionnaire** (possibilité de choisir « au hasard » sur Scratch 2, taper « random » sur Scratch 3) et **mettre** **Mot à deviner** **à** **Mot à deviner**.



⇒ Vérifier, en l'affichant temporairement pour tester, que **Mot à deviner** prend bien des valeurs différentes en cliquant plusieurs fois sur .

Enfin, on va remplir la liste **Lettres à découvrir** en lui ajoutant toutes les lettres de **Mot à deviner** qu'on va parcourir caractère après caractère dans une répétition.



Pour ce faire, on commence par initialiser **Numéro de caractère** à 1 pour signifier qu'on commence au premier caractère.

Puis, on effectue une répétition dont le nombre d'itérations est égal au nombre de lettres de **Mot à deviner** :




⇒ On distinguera bien **longueur de**  qui donne le nombre de caractères d'une chaîne de caractères, et **longueur de**  qui donne le nombre d'éléments d'une liste.

Dans la répétition :

- On demande d'ajouter dans **Lettres à découvrir** le caractère **Numéro de caractère** de **Mot à deviner** : on aura besoin de **lettre** **de**  et **ajouter** **à** **Lettres à découvrir**.
- On passe au caractère suivant en augmentant **Numéro de caractère** de 1 (utilisation de **ajouter à** .

Test :

Vérifier plusieurs fois en cliquant sur , que **Mot à deviner** change à chaque clic, et que **Lettres à découvrir** prend les bonnes valeurs.



Objectif principal 2

Idée générale :

En partant de **Mot affiché** initialisé à une chaîne vide, on parcourt chaque lettre de **Mot à deviner** dans une répétition, et on regarde si elle est contenue dans **Lettres déjà trouvées** ou non. Si c'est le cas, on ajoute la lettre à **Mot affiché**, sinon « _ ».

Application :

On commence par initialiser **Mot affiché** (chaîne vide), et mettre **Numéro de caractère** à 1 avant de commencer une répétition :



Dans cette répétition, on pourra employer **Lettres déjà trouvées** contient ☐ ? pour tester si une lettre est déjà contenue dans la liste, en s'inspirant de ceci :



- Si la lettre est contenue dans **Lettres déjà trouvées**, on la rajoute dans **Mot affiché** suivie d'une espace ^(*) (utilisation de **mettre** **Mot affiché** à ☐ et de plusieurs **regroupe** ☐ ☐).
- Sinon, on rajoute dans **Mot affiché** le tiret du bas (« _ ») suivi d'une espace également.

Avant la fin de la répétition, ne pas oublier de passer au caractère suivant.

Après la répétition, montrer **Mot affiché** (utilisation de **montrer la variable** ☐).

Test :

Pour tester le bon fonctionnement de ce bloc :



1. Double-cliquer sur **Initialisation** pour qu'un mot soit choisi dans le dictionnaire.
2. Montrer **Mot à deviner** (cliquer sur **montrer la variable** ☐).
3. Ajouter manuellement des lettres du mot choisi dans **Lettres déjà trouvées**.



4. Double-cliquer sur **Actualiser mot à afficher** pour vérifier que **Mot affiché** s'affiche correctement en fonction des lettres saisies dans **Lettres déjà trouvées**.

* L'espace (caractère typographique) est un nom féminin.

Objectif principal 3

Idée générale :

Dans une répétition qui aura lieu jusqu'à ce que **Fin du jeu** vaille « oui », on demande au joueur une lettre. Si cette lettre est dans **Mot à deviner** alors le coup est gagnant, sinon la **Potence** se dresse progressivement.

Application :

On crée une répétition jusqu'à ce que **Fin du jeu** soit égale à « oui », à l'intérieur de laquelle :

- On commence par **demande** **et attendre**.
 - ⇒ Le fait de laisser libre le champ après « demande » permet d'éviter qu'une ligne de texte comportant la question n'empiète sur l'image d'arrière-plan, sans gêner la compréhension.
- On teste si la saisie est acceptable. Pour cela, il suffit de vérifier que **Alphabet** contient **réponse** en utilisant :




- Si la saisie est acceptable, on teste si **réponse** est contenue dans **Lettres à découvrir**, en utilisant :



- Si c'est le cas (lettre gagnante) :
 - i. on ajoute **réponse** à **Lettres déjà trouvées**,
 - ii. on appelle **Actualiser mot à afficher** pour redessiner **Mot affiché**,
 - iii. on retire **réponse** de **Lettres à découvrir** : voir comment faire dans [l'encadré de la page suivante](#),
 - iv. si **Lettres à découvrir** n'a plus d'éléments (**longueur de** égale à 0) alors on met **Fin du jeu** à « oui ».
- Sinon (lettre perdante) :
 - i. on rajoute la **réponse** (+ espace) à **Erreurs** à l'aide de **regroupe** (comme dans [l'objectif précédent](#)),
 - ii. on affiche **Erreurs** (utilisation de **montrer la variable**),
 - iii. on appelle **envoyer à tous** **Lettre incorrecte !** **et attendre** pour actualiser le dessin de la **Potence**,
 - iv. dans le lutin **Potence**, après changement de costume, si on atteint le dernier **costume #**, alors on met **Fin du jeu** à « oui ».

Test :

Cliquer sur  et vérifier que le jeu fonctionne bien jusqu'à s'arrêter de demander une lettre quand le mot est découvert ou bien que la partie est perdue.

Pour retirer **réponse** de **Lettres à découvrir**, remettre dans l'ordre les instructions suivantes :





Objectif principal 4

Idée générale :

En fin de partie, **Abby** réagit au fait que le joueur a gagné ou perdu.
S'il a perdu, elle annonce le mot qu'il fallait trouver.

Application :

Abby se place vers la droite de la scène en regardant vers la gauche (changement de costume).

Si le joueur a gagné (cas où **longueur de** **Lettres à découvrir** ▼ vaut 0) alors elle sourit et réagit en conséquence.

Sinon, **Abby** se montre triste et annonce le mot qu'il fallait trouver.

Puis, après un court délai, une nouvelle partie peut commencer (utilisation de **envoyer à tous** **Démarrage du programme** ▼).