Titouan BRIERRE

Pierre DAVID

TD2, TP3

**Jeu des multiples**

**Description du jeu**

Le but du jeu est de trouver un mot caché en devinant les lettres qui le composent. Dans cette version du jeu :

* le programme propose un mot à deviner au joueur et le joueur essaie de deviner ce mot en proposant des lettres.
* le joueur peut faire au maximum 7 mauvaises propositions de lettres avant de trouver le mot mystère.

Le déroulement d’une partie est le suivant :

* Au début du jeu, le programme propose trois thèmes possibles pour jouer. Par exemple : informatique, cuisine et animaux.
* Le joueur choisit un thème (par exemple informatique) puis le programme sélectionne au hasard un mot faisant partie de ce thème (par exemple algorithme).
* Le programme affiche ensuite ce mot en représentant chaque lettre par un tiret. Bien que ce premier affichage ne présente aucune lettre du mot à trouver, le joueur peut compter le nombre de tirets et donc déduite le nombre de lettres du mot à trouver.

Le joueur propose ensuite une lettre :

* Si la lettre proposée par le joueur fait partie du mot à trouver, le programme réaffiche le mot en affichant où se trouvent toutes les occurrences de la lettre découverte. Etant donné que la lettre proposée fait partie du mot, le nombre de mauvaises propositions que le joueur peut encore faire reste inchangé.
* Si la lettre proposée par le joueur ne fait par partie du mot à trouver, le programme réaffiche le mot tel quel et comptabilise une mauvaise réponse pour le joueur.

Suite à chaque proposition du joueur (bonne ou mauvaise), le programme indique le nombre restant de mauvaises réponses possibles. Ce message sera affiché en vert lorsque le nombre de coups restants reste élevé, en violet lorsqu’il commence à devenir inquiétant et en rouge lorsqu’il ne reste plus qu’un coup.

La partie peut prendre fin de deux façons :

* soit le joueur gagne car il arrive à découvrir le mot mystère en ayant fait moins de 7 mauvaises propositions de lettres;
* soit le joueur perd car il a proposé 7 lettres qui ne font pas partie du mot et toutes les lettres du mot n’ont pas été découvertes.

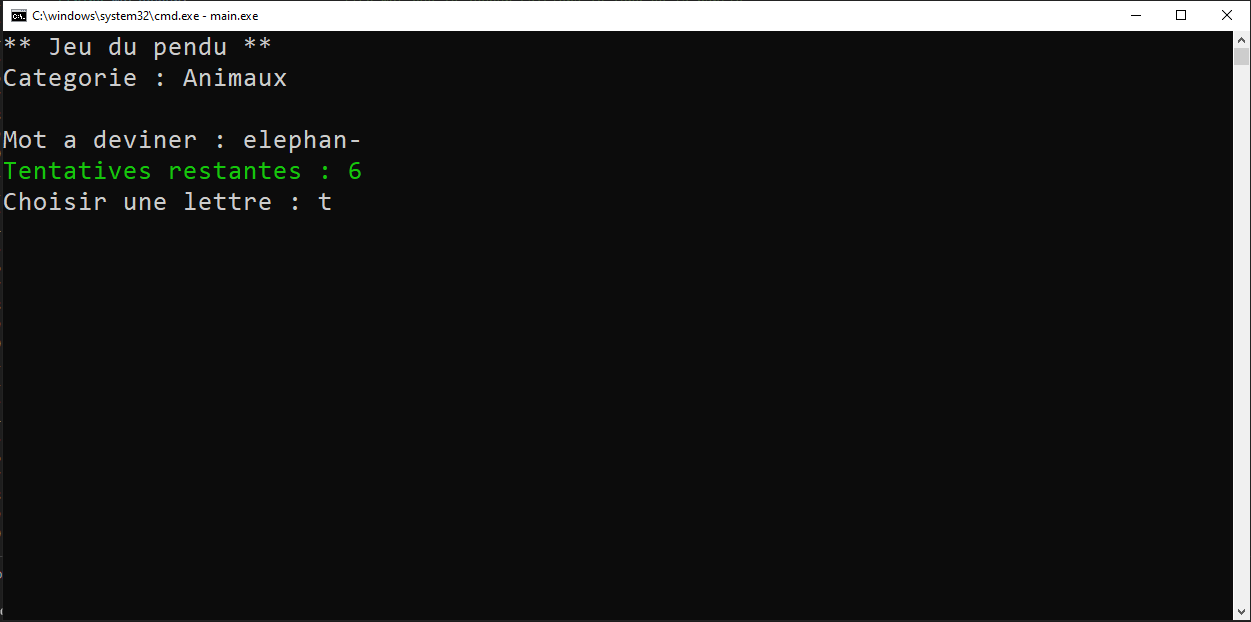
**Spécification du besoin**

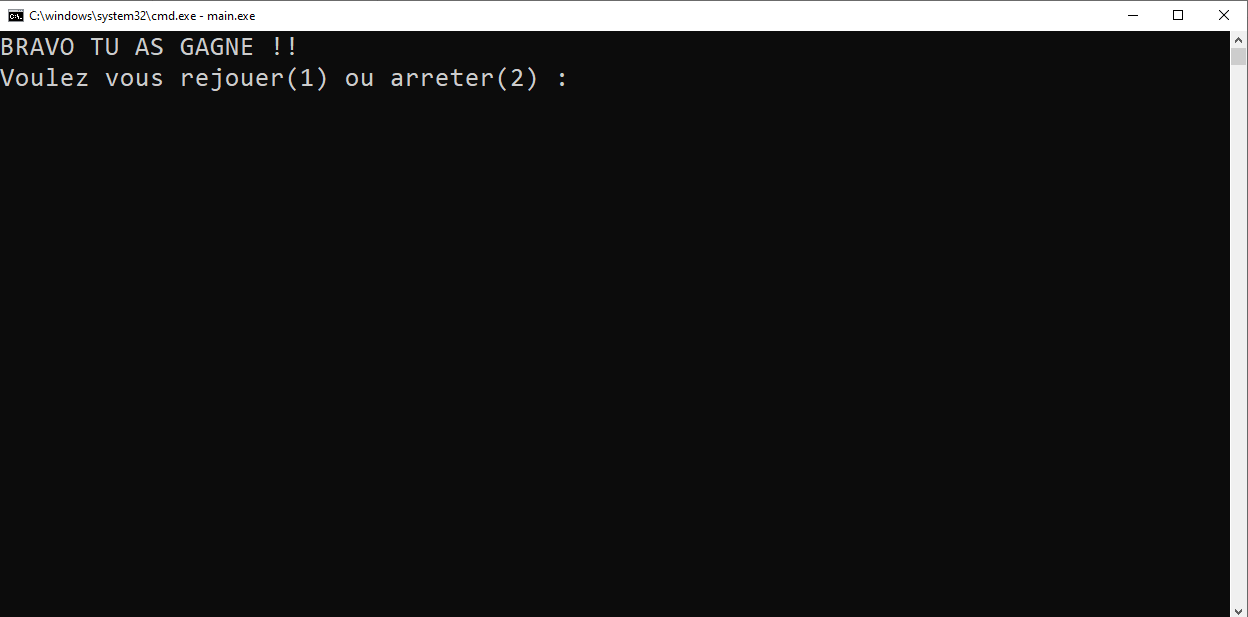
Le jeu s’exécute dans un terminal. Pour simplifier le problème, on considèrera que :

* le joueur peut choisir parmi 3 thèmes de mots;
* le nombre de mots possibles est identique pour chaque thème : Il y aura par exemple 6 mots possibles si le joueur choisit de jouer avec le thème 1, 6 mots possibles pour le thème 2 et 6 mots possibles pour le thème 3 ;
* les mots à découvrir sont toujours écrits en minuscules ;
* les mots à découvrir ne comportent pas d’accents, d’espaces ni de guillemets.

**Scénario nominal : Le joueur gagne la partie**

Ce scénario décrit le déroulement du jeu lorsque le joueur gagne la partie. Dans cet exemple, le joueur sélectionne le thème *animaux* puis gagne la partie en découvrant le mot *elephant* et en donnant seulement 1 mauvaises proposition de lettre.

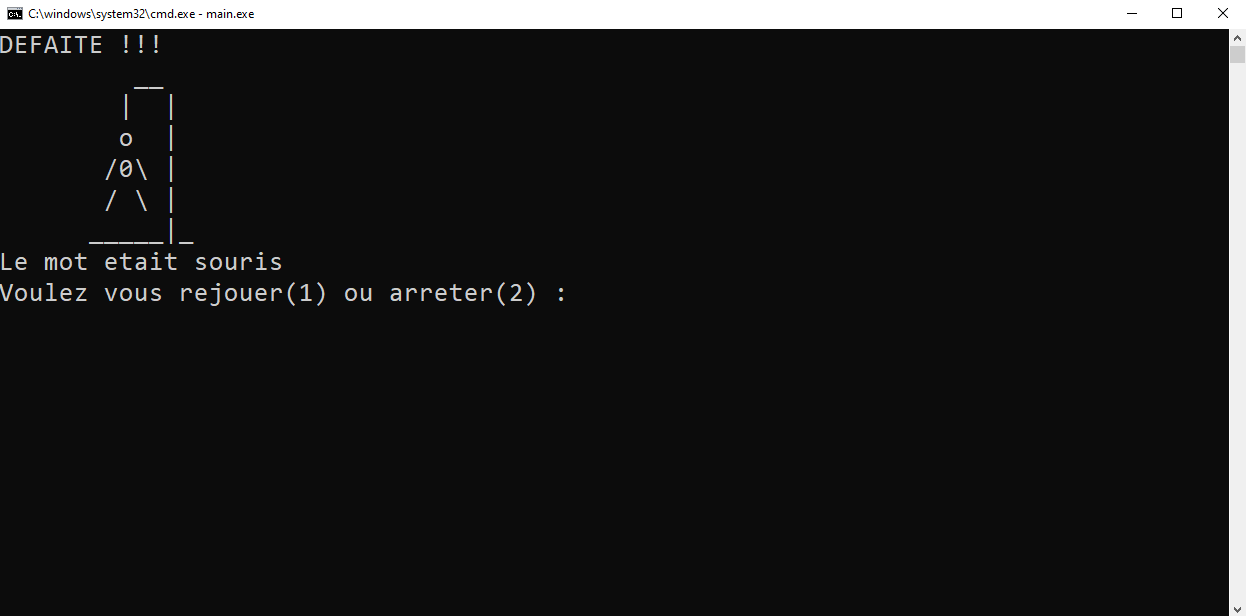




**Scénario alternatif : Le joueur perd la partie**

Ce scénario décrit le déroulement du jeu lorsque le joueur gagne la partie. Dans cet exemple, le joueur sélectionne le thème *informatique* mais ne gagne pas la partie car il ne découvre pas le mot caché *logiciel*. Il fournit 7 mauvaises propositions de lettres qui lui font perdre la partie.





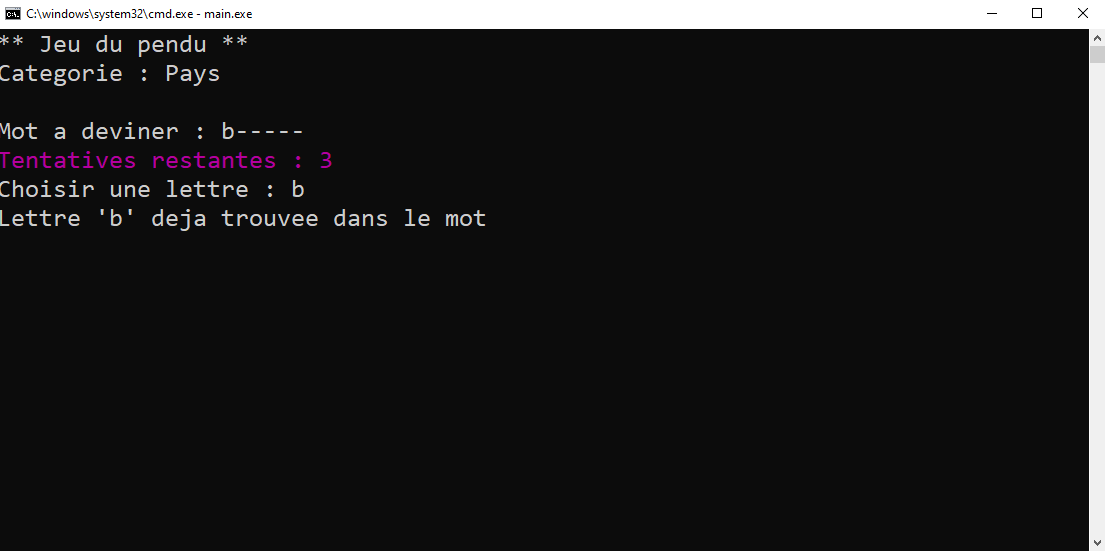
**Remarque** : Au fur et à mesure des mauvaises propositions faites par le joueur, le nombre de coups restants diminue et le message indiquant le nombre de coups restants change de couleur :

* le message est vert lorsque le nombre de coups restants reste élevé ;
* il passe en violet lorsque le nombre de coups restants commence à devenir inquiétant ;
* il devient rouge lorsqu’il ne reste plus qu’un seul coup.

**Scénario alternatif : Le joueur re-propose une lettre déjà découverte**

Ce scénario illustre le comportement du programme lorsque le joueur propose une lettre qui a été précédemment découverte : Le programme indique que la lettre est bien présente dans le mot et ne décompte pas le nombre de coups restants.

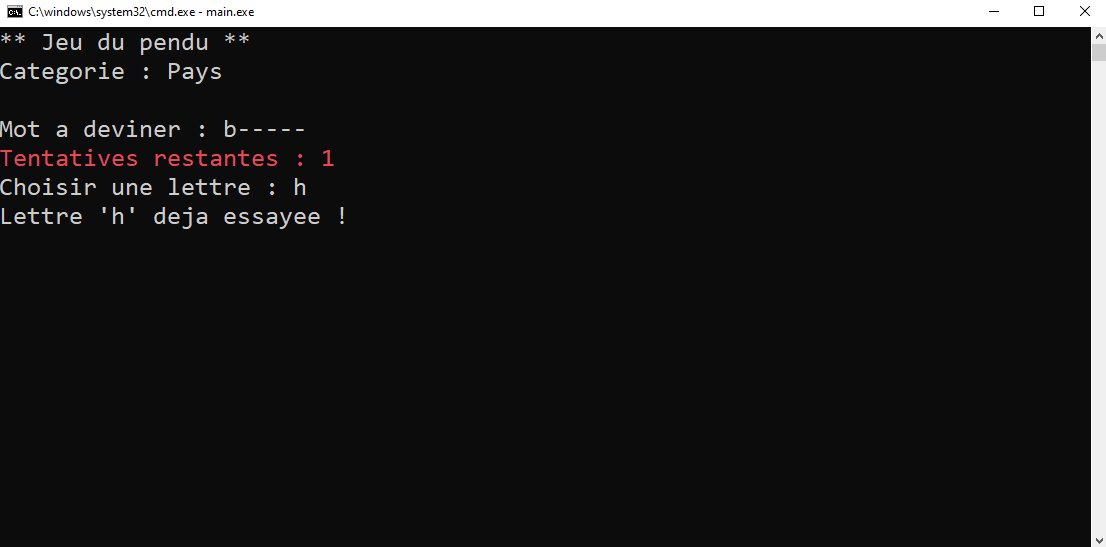
Dans l’exemple qui suit, le joueur propose une première fois la lettre *b*, puis la lettre *h* et à nouveau la lettre *b*. Lorsque la lettre *a* est à nouveau proposée, le programme indique que la lettre fait bien partie du mot et ne décompte pas le nombre de tentatives restantes.



**Scénario alternatif : Le joueur re-propose une lettre non présente dans le mot**

Ce scénario illustre comment le programme réagit lorsque le joueur re-propose une lettre qui a été précédemment proposée et qui ne fait pas partie du mot à découvrir. Ici cette erreur du joueur (qui n’a pas mémorisé les lettres déjà proposées) est sanctionnée : le nombre de coups restants est diminué.

Dans l’exemple qui suit, le joueur propose deux fois la lettre *p* : le nombre de tentatives restantes est diminué.



**Scénario d’exception : Le joueur saisit un numéro de thème incorrect**

Dans ce scénario, le joueur choisit un numéro de thème incorrect (valeur nom comprise entre 1 et

3. Le programme lui demande de saisir un nouveau numéro de thème.

Dans l’exemple qui suit le joueur demande à jouer avec le thème numéro 4 qui n’existe pas.



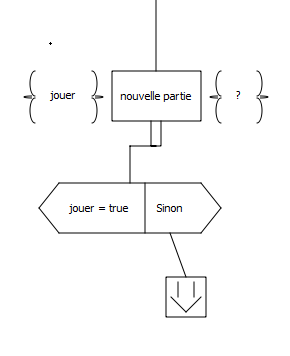
**Dictionnaire des variables**

|  |  |  |
| --- | --- | --- |
| Nom | Type | Signification |
| jouer | booléen | Définit si le joueur continue de jouer |
| presence | booléen | Définit si la lettre trouvée par le joueur est dans le mot à deviner |
| dejaSaisie | booléen | Définit si la lettre saisie par le joueur a déjà été saisie |
| choix | entier | Le nombre qui définit si le joueur souhaite rejouer |
| nbreEssais | entier | Le nombre de chances qu’il reste au joueur |
| categorie | entier | Le numéro de la catégorie choisie par le joueur |
| NBMOTS | entier | Le nombre de mots dans les tableaux de mots à deviner |
| nomCategorie | Chaîne de caractères | Le nom de la catégorie choisie par le joueur |
| lettre | Chaîne de caractères | La lettre saisie par le joueur\* |
| motDevine | Chaîne de caractères | Le mot que le joueur doit deviner |
| motJoueur | Chaîne de caractères | Le mot que le joueur a trouvé au cours de la partie |
| listeMauvaiseLettre | Chaîne de caractères | Contient les lettres incorrectes saisies par le joueur\* |
| MotInformatique | Liste de Chaînes de caractères | Tableau contenant les mots de la catégorie informatique |
| MotPays | Liste de Chaînes de caractères | Tableau contenant les mots de la catégorie pays |
| MotAnimaux | Liste de Chaînes de caractères | Tableau contenant les mots de la catégorie animaux |

**Actions**

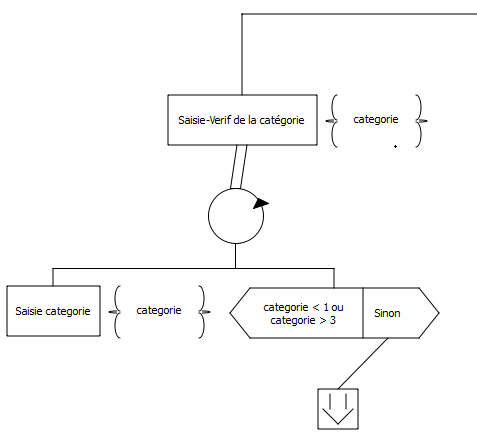
Action 1.1 : Lancer une nouvelle partie

Lance une nouvelle partie en fonction de la variable jouer (initialisée sur true)



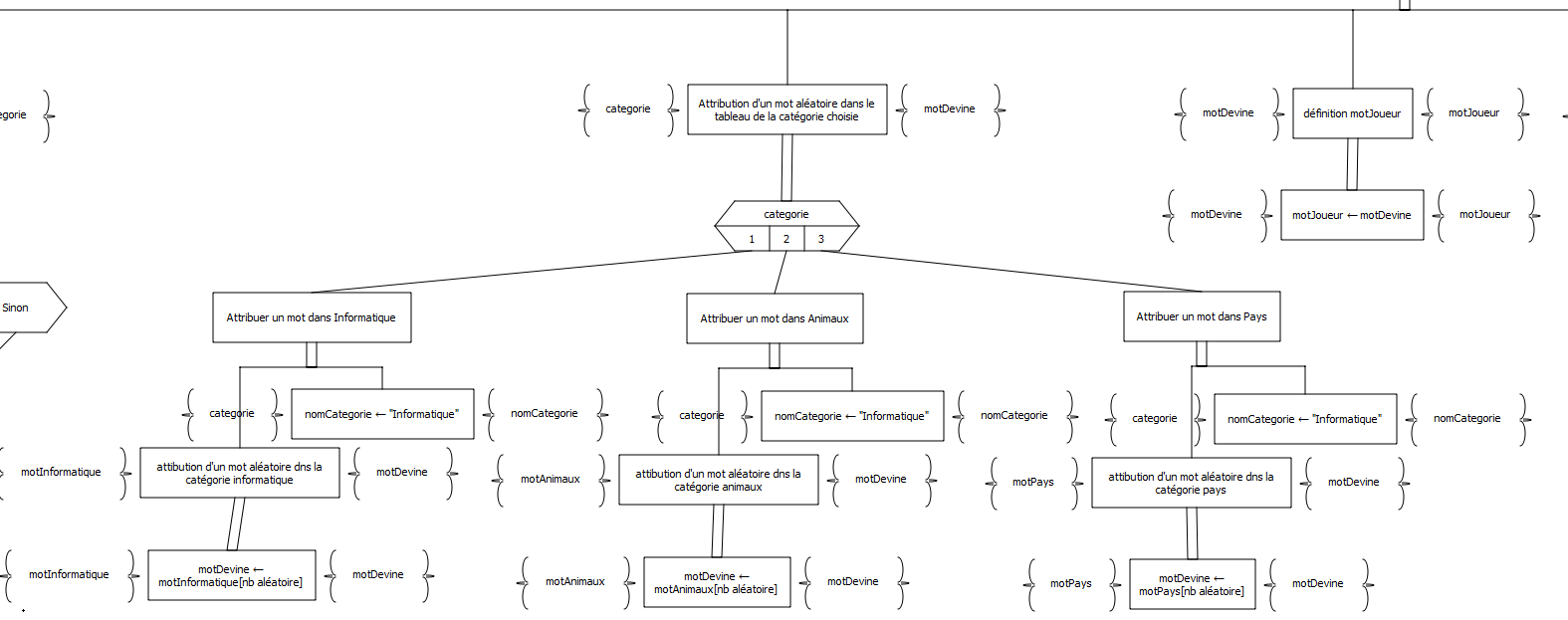
Action 1.2 : Choisir la catégorie

Une saisie verif est effectuée jusqu’à ce que le joueur choisisse une numero de catégorie valide



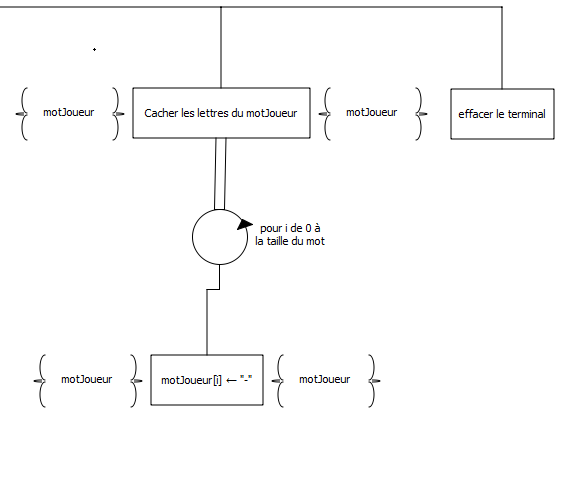
Action 1.3 : Initialisation des mots

Initialise motJoueur et motDevine en déterminant un mot aléatoire dans la catégorie choisie



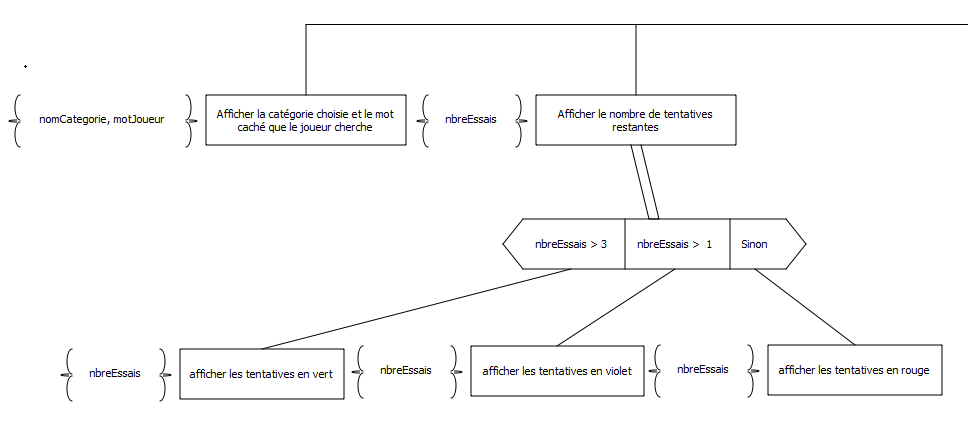
Action 1.4 : Cacher le motJoueur

Cache chaque lettre du mot par un tiret

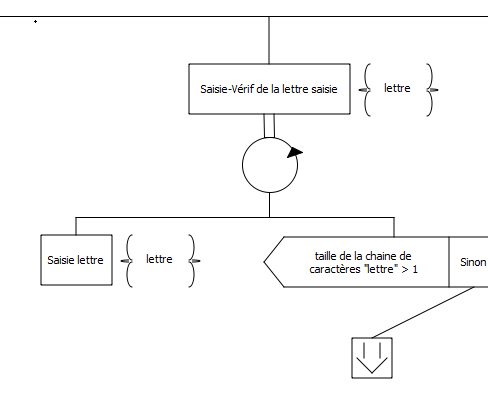


Action 2.1 : Affichage

Affiche toutes les informations nécessaires au joueur

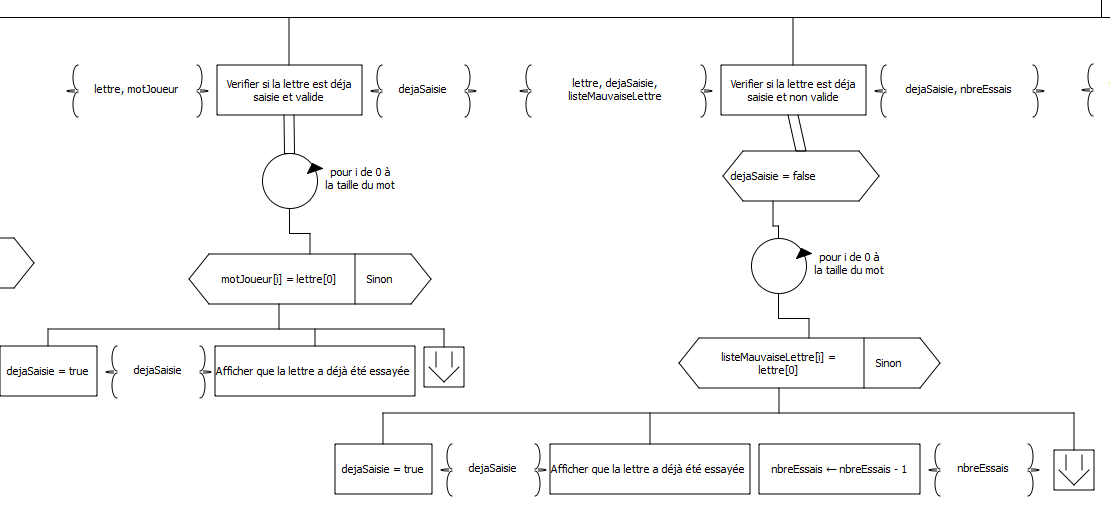


Action 2.2 : saisie-vérif de la lettre



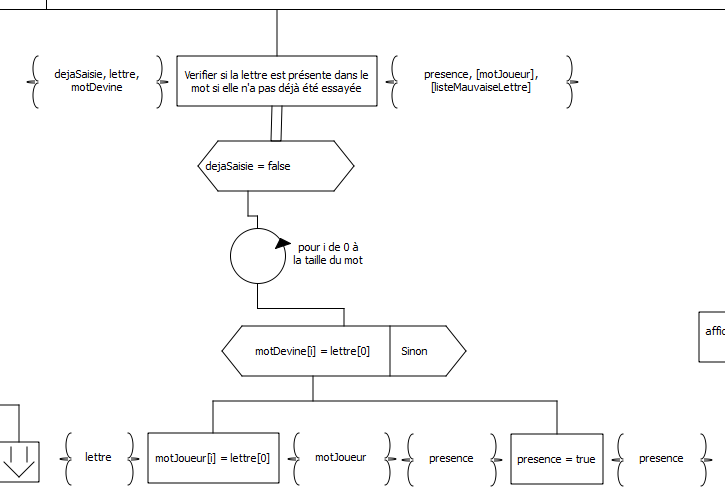
Action 2.3 : vérifier si la lettre a déjà été essayée

Vérifier si la lettre a déjà été saisie et correcte ou non



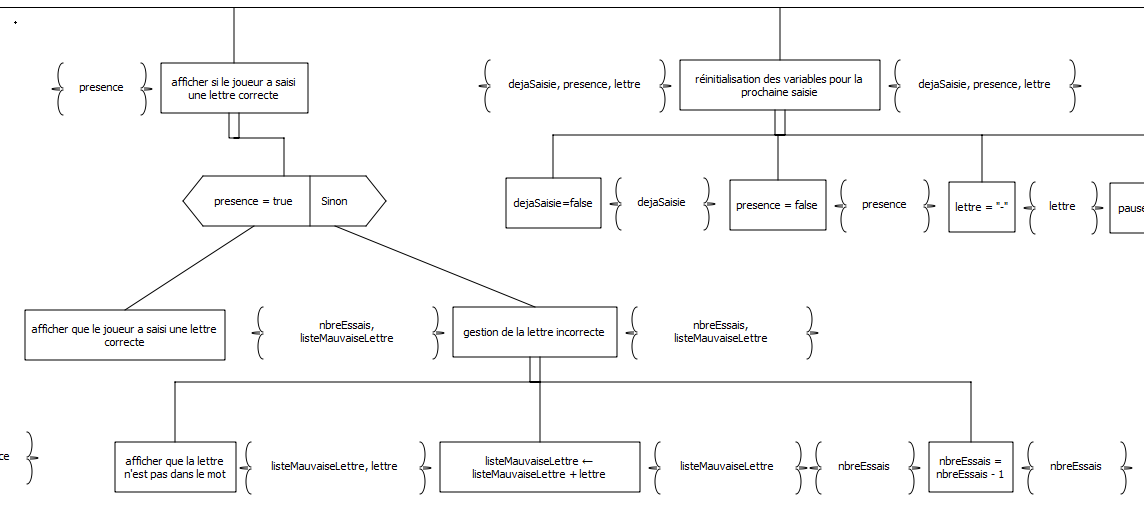
Action 2.4 : vérifier si la lettre est dans le mot

si la lettre n’est pas déjà essayée

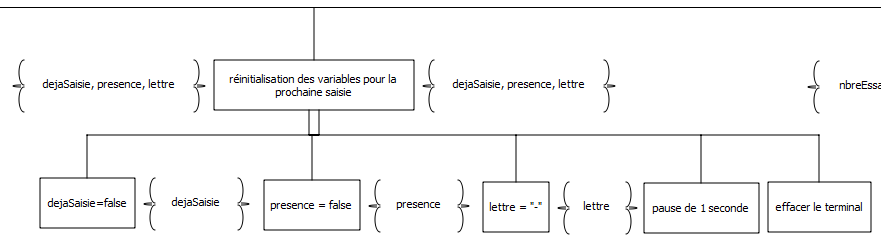


Action 2.5 : Indiquer au joueur si sa lettre est correcte

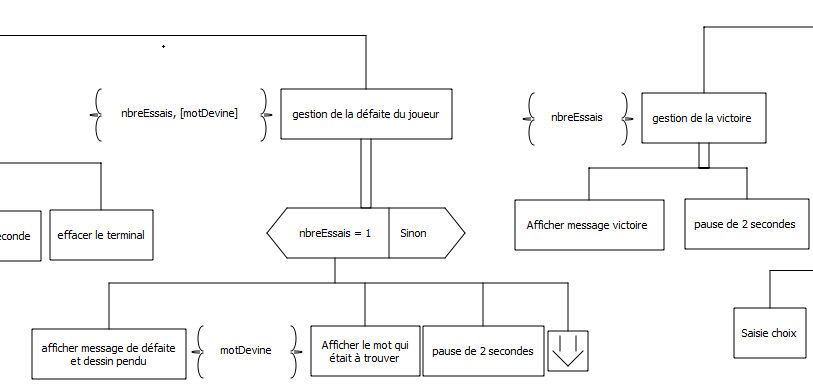
Traite aussi la décrémentation des tentatives et l’ajout de la lettre à la listeMauvaiseLettre si elle n’est pas correcte



Action 2.6 : Réinitialiser des variables pour le prochain tour

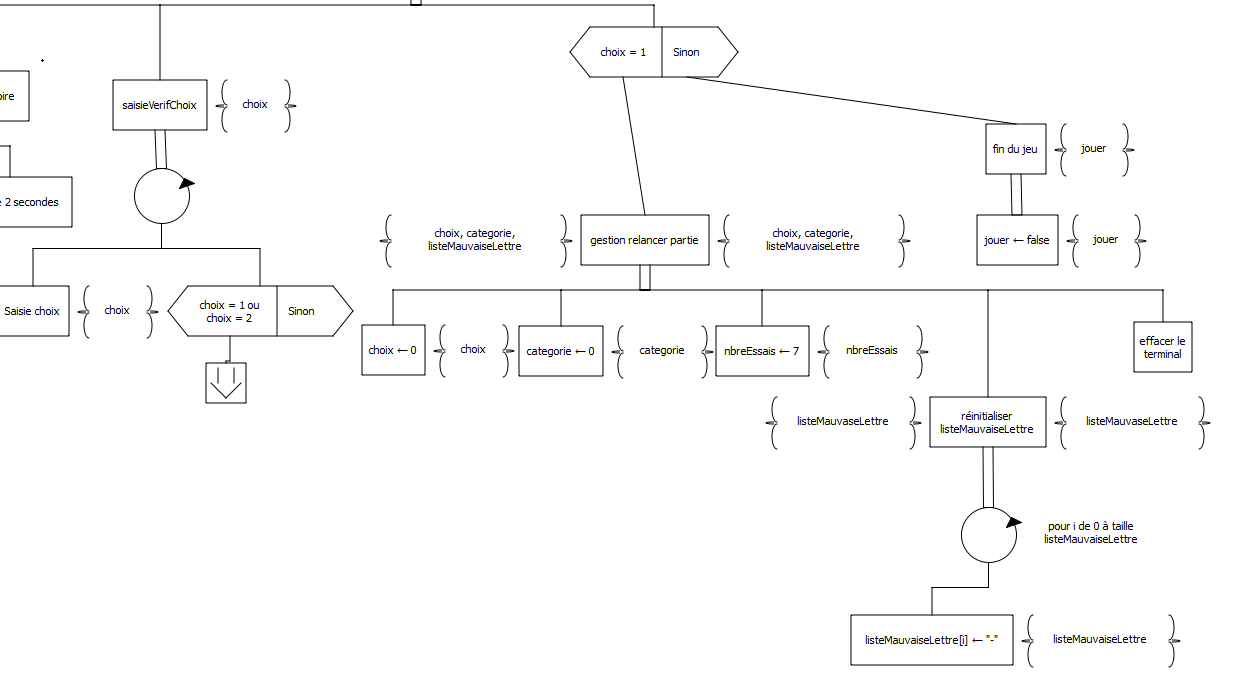


Action 3.1 : Afficher la défaite ou victoire du joueur



Action 3.2 : Proposer une nouvelle partie

Et réinitialise les variables nécessaires



**Remarques**

* + Les deux extensions ont été élaborées, le programme affichera si la lettre a déjà été essayée dans la situation où la lettre est dans le mot et dans celle où elle n’y est pas.
  + Le programme utilise la bibliothèque game-tools
  + \*précision sur les variables :
  + - lettre est une chaîne de caractères pour que le joueur puisse saisir les lettres uniquement une par une en effectuant une saisie-verif
  + - \*listeMauvaiseLettre enregistre les erreurs du joueur pour pouvoir vérifier si la lettre a déjà été saisie
  + Il est aussi possible de rejouer, le programme entier a été mis dans une boucle jusqu’à ce que le joueur souhaite arrêter.