Jeu Motus

Auteurs: Téo Baillot d'Estivaux et Simon Leguilloux.

Dates: du 25 avril au 10 juin 2024.

Sommaire:

Introduction

Le but de ce projet est de reproduire le célèbre jeu télévisé motus en utilisant toutes les connaissances que nous avons pû acquérir pour écrire des programmes en langage C depuis le début de cette année scolaire et la librairie libsx qui nous permettra de concevoir une interface graphique.
Si vous ne le savez pas, Motus est un jeu où les joueurs ont un certain nombre de tentatives (6 dans notre cas) pour trouver un mot. A chaque tentative, le joueur donnera un mot qui sera comparé au mot qu'il doit trouver et obtiendra des indications qui lui permettront au fur et à mesure de la partie de se rapprocher du mot à trouver. Si une lettre est affichée en bleu, ceci indique au joueur que la lettre n'est pas présente dans le mot à trouver, si une lettre s'affiche en rouge, le joueur pourra en déduire que la lettre est dans le mot à trouvé et à la bonne place, et si une lettre est affichée en jaune, il en déduira que la lettre est présente dans le mot à trouver mais pas à la bonne place.
Une première version de ce jeu a été réalisée sans interface graphique puis a été adaptée afin de fonctionner avec l'interface graphique.

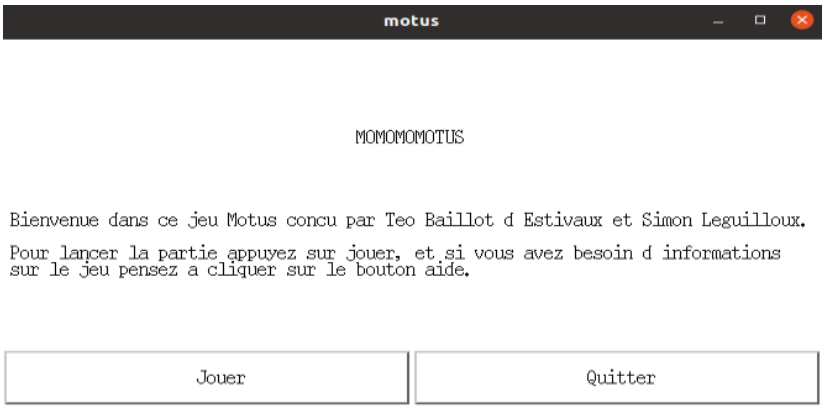
Pour tester le jeu, placez vous dans le répertoire contenant les fichiers des codes du jeu puis exécutez la commande "make" dans votre terminal.
Une fois que ceci est fait, il ne vous reste plus qu'à écrire la commande "./motus" et à vous amuser.

Exemple d'une partie:

Explications des différentes fonctionnalités

Notre projet contient différentes fonctionnalités:

- Au lancement de l'exécutable du jeu, vous vous trouverez dans une fenêtre de démarrage contenant un court paragraphe pour vous donner quelques indications.
  Dans cette fenêtre, vous aurez accès à un bouton "jouer" qui vous permettra de lancer une partie et un bouton "quitter" qui vous permettra de terminer l'exécution du programme. Une fois que vous cliquerez sur le bouton "jouer", vous ne pourrez pas cliquer sur ce bouton une seconde fois car il sera bloqué pour éviter le problème d'avoir plusieurs fenêtres de jeu ouvertes en même temps.

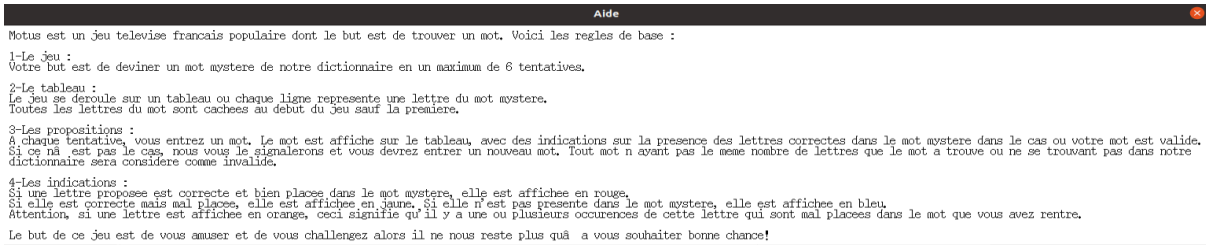


- En cliquant sur le bouton jouer, le jeu ouvrira une nouvelle fenêtre et initialisera le début de la partie. Dans cette fenêtre vous trouverez plusieurs éléments:

  - ➢ Vous retrouverez une nouvelle fois un bouton "quitter" qui vous permettra de terminer l'exécution du programme.

  - ➢ Un bouton "rejouer" qui vous permettra de relancer une nouvelle partie à tout moment.

  - ➢ Un bouton "aide" qui ouvrira une fenêtre d'aide qui vous expliquera les règles du jeu et vous donnera quelques informations complémentaires à celui-ci:



- ➢ Une première zone de saisie “ZoneSaisieNombreEssaisRestants” (celle qui est la plus à gauche) qui nous permettra d’indiquer à l’utilisateur le nombre d’essais qu’il lui reste à tout moment de la partie.

- ➢ Une seconde zone de saisie “ZoneSaisieValideInvalide” (celle qui est au milieu) qui nous permettra d’indiquer à l’utilisateur à chaque fois qu’il rentre un mot si celui-ci est correct ou non.

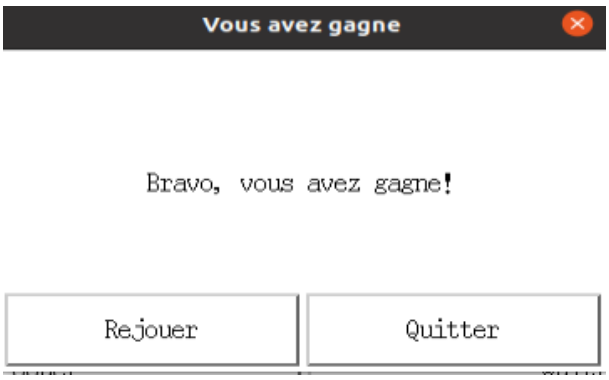
- ➢ Un label qui indiquera à l’utilisateur la zone de saisie qu’il doit utiliser pour entrer ses mots.

- ➢ Une dernière zone de saisie “ZoneSaisieUtilisateur” (celle qui est la plus à droite) qui permettra à l’utilisateur d’écrire les mots qu’il souhaite tester.

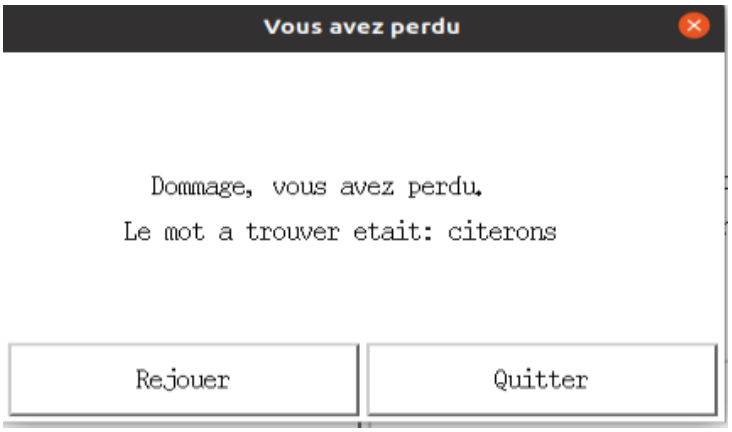
- ➢ Un bouton “valider” qui permettra à l’utilisateur de tester si le mot qu’il a écrit dans la zone de saisie précédente est le même que le mot à trouver.

- ➢ Une zone de dessin “DrawArea” qui affichera la première lettre du mot à trouver suivi de dessin au début de la partie. A chaque mot valide que l’utilisateur testera, une nouvelle ligne apparaîtra pour lui donner des indications qui lui permettront de savoir s’il est proche du mot à trouver ou non. Un mot qui n’a pas la même taille qu’un mot à trouver ou un mot qui n’est pas dans le dictionnaire fourni par le jeu ne sera pas considéré comme valide. Les indications seront les suivantes: si une lettre est en rouge, cela signifie qu’elle est à la bonne place, si la lettre est en jaune, cela signifie qu’elle est dans le mot à trouver mais pas à la bonne place, et enfin, si la lettre est en bleu, cela signifie qu’elle ne se trouve pas dans le mot à trouver.

- ❖ Si l’utilisateur gagne la partie, une fenêtre s’ouvrira pour le lui indiquer à l’aide d’un label qui affichera un message. Dans cette fenêtre vous trouverez un bouton “rejouer” qui vous permettra de relancer une nouvelle partie, et un bouton quitter qui vous permettra de terminer l’exécution du programme:



- De même, si l'utilisateur perd la partie, une fenêtre s'ouvrira pour le lui indiquer à l'aide d'un label qui affichera un message. Dans cette fenêtre vous trouverez également un bouton "rejouer" qui vous permettra de relancer une nouvelle partie, et un bouton quitter qui vous permettra de terminer l'exécution du programme:



Les différents choix effectués

- Dans la structure de de donnée, nous avons créé une structure "DonneePartagee" contenant différents données essentielles dont nous aurons besoin dans les fonction de Callbacks que nous pourrons ainsi manipuler sans manipuler directement le modèle:

  - Un pointeur sur le premier caractère du mot à trouver "MotATrouver" qui nous permettra notamment de pouvoir initialiser le mot à trouver en début de partie et le réinitialiser si l'utilisateur clique sur un des boutons "rejouer". Ceci nous permettra également d'indiquer à l'utilisateur le mot qu'il devait trouver dans le cas où celui-ci perd la partie.

  - Un pointeur sur le premier caractère du mot qui sera entré par l'utilisateur "MotUtilisateur", qui nous permettra d'actualiser le mot entré par l'utilisateur à chaque fois qu'il clique sur le bouton valider, vérifier si le mot entré est valide, comparé le mot entré par l'utilisateur au mot à trouver pour pouvoir donner les

indications dont on a parlé précédemment à l'utilisateur, et actualiser le dernier mot valide entré par l'utilisateur.

- ➢ Un entier "tailleMotATrouver" qui sera actualisé à chaque partie et qui nous permettra principalement de gérer correctement l'affichage du jeu en dessinant autant de carré dans la zone de dessin "DrawArea" qu'il y a de lettres dans le mot à trouver.

- ➢ Un entier "numeroEssai" qui nous permettra d'actualiser combien d'essais l'utilisateur a utilisé afin d'ouvrir la fenêtre pour lui indiquer qu'il a perdu dans le cas où il a utilisé tous ses essais (quand le numéro d'essai est le même que le nombre d'essais max définis dans le header data.h par "nombreEssaiMax"). Chaque fois que l'utilisateur appuiera sur le bouton "valider", ceci sera comptabilisé comme une tentative même si le mot entré par l'utilisateur est invalide.

- ➢ Un entier "premiersMotsInvalides" qui servira de booléen et qui indiquera si l'utilisateur a entré un mot valide depuis le début de la partie afin de gérer l'affichage. Si l'utilisateur entre un mot Invalide et qu'il n'avait jamais entré de mot valide depuis le début de la partie, on écrira sur une nouvelle ligne la première lettre du mot à trouver suivi de tirets. Cependant, si l'utilisateur entre un mot invalide mais qu'avant il avait écrit un mot valide, on écrira sur une nouvelle ligne le dernier mot valide qu'il a entré (et les indications qui vont avec).

- ➢ Ainsi, pour pouvoir récupérer le dernier mot valide entré par l'utilisateur qui nous permettra de faire le fonctionnement décrit dans le paragraphe précédent, nous avons également mis un pointeur sur un caractère "dernierMotValide" qui pointera constamment sur le premier caractère du dernier mot valide entré par l'utilisateur.

- ➢ Enfin, nous avons utilisé un pointeur sur le premier caractère d'une chaîne de caractères qui contiendra des 'o', des 'x', ou des '-' "indications" qui nous permettra de gérer l'affichage afin de donner des indications décrites précédemment à l'utilisateur en parcourant tous les caractères du mot de la chaîne de caractère dont le premier caractère est pointé par ce pointeur. Si le caractère est '-', cela signifie que la lettre n'est pas dans le mot, si le caractère est 'o' cela veut dire que la lettre est présente dans le mot à trouver mais pas à la bonne place. Si la lettre est 'x', ceci indiquera que la lettre est à la bonne place.

- ❖ Pour ce qui est de l'algorithme, l'idée est d'avoir une interface dynamique qui réagit en fonction des actions de l'utilisateur, si celui-ci ne fait rien, le programme attendra jusqu'à recevoir une action de l'utilisateur. L'utilisateur pourra agir avec les différents éléments de l'interface décrits précédemment. Décrivons plus en détail le fonctionnement de l'algorithme:

- Lorsque l'utilisateur clique sur le bouton "jouer" dans la fenêtre de démarrage, toutes les données partagées sont initialisées et la fenêtre principale sera également initialisée et s'ouvrira.

- Dans cette nouvelle fenêtre, rien ne se passera si l'utilisateur n'interagit pas avec les différents éléments décrits précédemment.

- La principale fonctionnalité que nous n'avons pas encore décrite en détail est ce qui se passe lorsque l'utilisateur clique sur le bouton "valider" pour entrer un mot.

    - Lorsque cet événement se produit, la première chose que nous vérifions est la taille du mot entré par l'utilisateur. Si l'utilisateur entre un mot qui a une taille supérieure à la taille maximum autorisée +1 (la taille maximum autorisée est défini dans le header data.h par "tailleMotMax"), on remplace la chaîne de caractère entrée par une chaîne de caractères vides. L'allocation mémoire dédiée à la chaîne de caractère étant limitée, ceci permet d'éviter d'avoir une segmentation fault si le mot entré par l'utilisateur est trop long).

    - Par la suite, nous actualisons le mot entré par l'utilisateur dans les données partagées.

    - Puis, on efface le contenu de la zone de saisie pour que l'utilisateur n'ait pas à le faire entre chaque tentative.

    - Dans la suite, on teste si le mot entré par l'utilisateur est valide.

    - Si c'est le cas, on indique à l'utilisateur via la zone de saisie décrite précédemment que son mot est valide, et on actualise dans la donnée partagée le dernier mot entré par l'utilisateur et l'entier indiquant que l'utilisateur a entré un mot valide depuis le début de la partie. On compare également le mot entré par l'utilisateur avec le mot entré et on actualise ainsi dans les données partagées les indications à donner à l'utilisateur. Ensuite, on vérifie si le mot entré par l'utilisateur est le même que le mot à trouver et si c'est le cas on ouvre la fenêtre pour lui indiquer qu'il a gagné. Enfin on gère l'affichage comme décrit précédemment en fonction de nos données partagées.

    - Si ce n'est pas le cas, nous devrons traiter deux autres cas. Le premier est le cas où le mot est invalide et l'utilisateur a entré un mot valide dans un de ses essais précédents. On indique donc à l'utilisateur que le mot qu'il a entré est invalide et on gère l'affichage en indiquant les informations relatives au dernier mot valide qu'il a entré en gérant l'affichage.

    - Le second cas a traiter est si le mot entré par l'utilisateur est invalide et l'utilisateur n'a pas entré de mot valide depuis le début de la partie. Dans ce

cas, on signale également à l'utilisateur que le mot qu'il a entré est invalide et on affiche de nouveau la première lettre du mot à trouver suivi de tirets.

- Enfin, on décrémente le nombre d'essais qu'il reste à l'utilisateur et s'il ne lui en reste plus, on affiche la fenêtre indiquant qu'il a perdu que nous avons décrite précédemment.

Conclusion

Les problèmes rencontrés

Nous avons finalement réussi à réaliser un jeu motus fonctionnel tout en travaillant son ergonomie mais également sa fiabilité en prenant en compte différentes actions qu'un utilisateur voulant tester les limites du jeu aurait pu faire. Malgré tout, nous avons rencontré de nombreuses difficultés tout au long du projet que nous allons décrire dans les paragraphes suivant:

- La fonction qui nous a posé le plus de problèmes est celle qui permet de donner les indications à l'utilisateur. En effet, nous avons fait différentes versions de cette fonction mais n'avons pas réussi à réaliser exactement la fonction voulue. Par exemple, nous voulions que si par exemple, il y a trois occurrences d'une lettre dans un mot, si l'utilisateur entre un mot avec une occurrence et cette lettre bien placée et deux occurrences de cette lettre mal placés, Il y aura seulement une des deux occurrences mal placés qui s'affiche en orange. De plus, si deux occurrences d'une lettre sont à la bonne place et que dans la suite du mot de l'utilisateur il y a d'autres occurrences de ces lettres qui ne sont pas à la bonne place, celles-ci ne seront pas affichées en orange. Nous avions fait une version qui réglait ce problème en faisant des conversions de caractères en entier mais avons décidé de ne pas la garder dans la version finale par peur que ces conversions ne renvoient pas les mêmes résultats sur tous les systèmes d'exploitation et donc que notre programme ne fonctionne pas. Dans la version du rendu final, il y aura donc deux cas particuliers qui ne fonctionnent pas parfaitement mais ces cas particuliers ont été expliqués dans les règles via le bouton "aide" afin que l'utilisateur les prenne en compte.

- La deuxième chose qui nous a posé le plus de difficultés a été de tester les différentes actions que pourrait faire un utilisateur qui ne veut pas simplement jouer au jeu, mais qui voudrait tester ses limites. Nous nous sommes donc mis à la place d'un utilisateur qui voudrait tester les limites du jeu en faisant des actions auxquelles on pourrait ne pas penser comme fermer la fenêtre principale quand une fenêtre s'ouvre devant, on a bloqué le bouton jouer une fois qu'il a été cliqué une fois afin d'empêcher l'utilisateur d'avoir 2 fenêtres de jeu ouvertes en même temps ce qui posait des problèmes si l'utilisateur écrivait dans les deux fenêtres, nous avons

empêché que l'utilisateur puisse ouvrir plusieurs fenêtres d'aides en même temps pour éviter que l'interface n'ait trop de fenêtres ouvertes, et bien d'autres.

- Un autre problème que nous avons rencontré et qui est assez gênant est que si la résolution de l'écran de l'utilisateur est trop petit, l'utilisateur ne voit pas le contenu des zones de saisie à cause de la zone de dessin "DrawArea" qui apparaît comme trop grande, l'utilisateur ne voit donc pas le mot qu'il écrit dans la zone de saisie (mais il peut quand même écrire dedans), et les indications indiquant si le mot saisie est valide ou non et le nombre d'essais qu'il lui reste. Nous avons donc fait le choix de réduire la taille de cette zone de dessin pour limiter au mieux ce risque malgré le fait que ceci rend les boîtes contenant les caractères moins visibles par l'utilisateur.

Les améliorations envisageables:

Il y a plusieurs points que nous aurions pu améliorer dans notre jeu:

- Tout d'abord, l'amélioration la plus importante à apporter serait de faire une fonction qui donne les indications qui marche parfaitement pour tous les cas particuliers que l'on peut imaginer.

- Nous aurions également pu imaginer instaurer plusieurs niveaux de difficultés dans le jeu permettant ainsi à l'utilisateur de choisir la difficulté qui lui convient le mieux (ce que nous n'avons pas fait par manque de temps).

- De plus, nous aurions pû apporter quelques améliorations au design de l'interface afin de la rendre encore plus jolie et agréable à jouer.

- Par la suite, nous aurions pu proposer une fonctionnalité permettant à l'utilisateur d'ajouter des mots de son choix au dictionnaire avant de commencer une partie.

- Puis, le manque de temps nous a posé problème car les explications que nous avons reçues au sujet de la librairie pour faire l'interface graphique ont été données peu de temps avant la date de rendu.

- Enfin, nous aurions pû ajouter des mots au dictionnaire car certains mots courants de la langue française ne figurent pas dans le dictionnaire et sont donc comptés comme invalide lorsque l'utilisateur les teste, ce qui lui fait perdre de façon injuste des tentatives.

Finalement, ce projet a été un challenge car il nous a permis de mettre en pratique toutes les connaissances que nous avons acquises au cours de cette année scolaire afin de faire quelque chose de concret. Cela nous a permis d'apprendre à nous débrouiller par nous même et chercher des fonctions que nous ne connaissions pas afin de réaliser différentes tâches. Nous avons ainsi eu un

aperçu plus concret des possibilités qu'offrent le langage C et avons donné le meilleur de nous même pour rendre un projet fonctionnel et agréable dans le temps imparti.