Courtecuisse Erik
Architecture Ordinateur

Jeu de Nim

Présentation du sujet choisi :

Jeu de Nim : 20 allumettes sont alignées. A tour de rôle, deux joueurs prennent 1, 2 ou 3 allumettes. Celui qui prend la dernière a gagné.

- Variante : plusieurs lignes de plusieurs allumettes.

  La variante pouvait être interprétée de différentes manières (plusieurs lignes qui auraient toutes permises la victoire une fois la dernière allumette d'une ligne enlevée)

  Ou littéralement, celle que nous avons choisie : plusieurs lignes d'allumettes permettant la victoire une fois que la toute dernière allumette est retirée.

Il existe une multitude d'autres variantes du jeu de Nim, il peut y avoir plus ou moins de 20 allumettes et les joueurs peuvent retirer plus ou moins de 3 allumettes.

Nom du fichier qui contient le programme source :

ProjetArchiMain.X68
Pour lancer le projet, il faut exécuter le fichier 'ProjetArchiMain.X68'
Le dossier du projet contient les fichiers suivant :

Le fichier d'appel des sous-programmes (Menu) :

- ProjetArchiMain.X68

Les bibliothèques d'appels systèmes :

- BIBGRAPH.X68
- BIBLIO.X68
- BIBPERIPH .X68

Les sous programmes :

- ProjetArchiMulti.X68
- ProjetArchiSolo.X68
- ProjetArchiSoloImpossible.X68

Les fichiers son :

- Craqueallumette.wav
- Endgame.wav
- Endgame.wav
- Startgame.wav

Autres :

- README.txt
- Projets.pdf
- AnnexecodePapier.rtf
- DossierPapier.docx

Les fonctions utilisées :

Dans BIBGRAPH.X68 :

- DRAW_RECT:
  Utilisée pour dessiner les rectangles dans le menu, pour dessiner un bouton cliquable, le bois des allumettes …
- DRAW_LINE:
  Utilisée pour dessiner les lignes diverses, le tableau des score et des règles …
- DRAW_PIX:
  Utilisée pour sélectionner un pixel afin de coloriser une zone avec 'POT_DE_PEINTURE, utilisé également pour remettre des zones en noir afin de supprimer des éléments.
- SET_PEN_COLOR:
  Utilisée pour sélectionner une couleur d'écriture (blanc).
- SET_FILL_COLOR:
  Utilisée pour sélectionner une couleur de remplissage.
- POT_DE_PEINTURE:
  Utiliser pour sélectionner une couleur et coloriser les allumettes.
- DRAW_ELLIPSE:
  Utilisée pour dessiner les ronds et faire le souffres dans allumettes.

Dans BIBLIO.X68 :

- SAISCAR:
  Utilisée pour attendre l'action d'un joueur au clavier, pour sélectionner le nombre d'allumettes à retirer.
- AFFCAR:
  Utilisée pour afficher des caractères dans de nombreux cas, pour afficher les règles, le contenu des boutons cliquables, le tour du joueur, les informations générales.
- POS_CURS:
  Utilisée pour positionner le curseur avant d'afficher des caractères ou des chaines de caractères.
- PLAY_SOUND:
  Fonction non répertoriée dans la liste d'appels systèmes, elle permet de jouer un son (.wav), notamment le son des allumettes lors de l'action d'un joueur, le début de partie et la fin de partie et le son d'ambiance (mis en commentaire).
- FINPRG:
  Utilisée pour terminer le programme.

Dans BIBPERIPH.X68 :

- RESOLUTION:
  Utilisée pour modifier la résolution de la fenêtre de jeu et ainsi laisser la place au tableau de score et l'affichage des règles du jeu.
- GET_MOUSE:
  Utilisée pour récupérer les informations de la sourie lors du clique du joueur sur une zone cliquable, pour relancer une partie ou non, pour se diriger dans le menu de choix de niveau.

Les algorithmes mis en œuvres :

- Algorithme d'affichage des allumettes
  Cet algorithme est en fait composé de deux boucle quasi identiques, une pour la première ligne, l'autre pour la seconde ligne d'allumettes, dans ces boucles on trouve de nouveau deux boucles semblable , l'une pour dessiner les bâtons des allumettes, l'autre pour dessiner le souffre au-dessus du bâton.

- Algorithme d'affichage d'une chaine de caractère
  Cet algorithme est une boucle qui utilise l'adressage indirecte pour avoir accès à chacun des caractères d'une chaine et les affiches un par un jusqu'à ce qu'il rencontre '0'

- Algorithme permettant de créer une zone cliquable
  Cet algorithme récupère le clique du joueur et sa position dans la fenêtre, si il s'agit du clique gauche l'algorithme va vérifier si la position n'est pas trop haute ni trop basse par rapport au bouton cliquable que l'on souhaite créer, si ce n'est pas le cas le programme continue sinon il boucle.

- Algorithme d'affichage d'un score jusque 99
  Cet algorithme commence par afficher 00 et ajoute 1 à chaque victoire d'un joueur(ou de l'ordinateur). Cet algorithme prend malheureusement beaucoup de place.

- Boucle générale (renvoi aux différents algorithmes selon l'action du joueur)
  Cette boucle est la boucle principale du programme, c'est elle qui attend la saisie d'une touche par le ou les joueurs et renvoi vers la suite du programme en fonction de cette dernière.

- Algorithme de saut de ligne
  Cet algorithme a été directeur dans la programmation du projet, en effet il consiste en la vérification de la dernière allumette retirée à chaque tour de boucle générale, ce faisant il permet de renvoyer sur la suppression de une, deux ou trois allumettes dans la deuxième ligne en fonction du nombre d'allumettes restantes dans la première lors de l'action du joueur.

- Algorithme de test du gagnant et changement de joueur
  Cet algorithme permet de vérifier quel est le gagnant de la partie grâce au passage dans le code qui définit à qui est le tour. Si le jeu se termine avec le joueur 1 dans le registre qui l'enregistre alors ce sera le gagnant et inversement.

- Algorithme de timming :
  Cet algorithme est une boucle qui part de 0 dans le registre D1 et qui n'en sort que lorsqu'elle a atteint un nombre énorme. De cette manière nous avons saturé la mémoire et généré une sorte d'arrêt du programme. Cela donne l'impression que le programme réfléchis alors que celui-ci est parfaitement fluide en temps normal. Il est alors possible de faire patienter le joueur plus longtemps en augmentant la taille du nombre à atteindre par le registre.
  Cet algorithme est très simple, il est utilisé dans le mode solo et impossible pour donner l'impression de réflexion de l'ordinateur ainsi que dans le menu principal pour éviter que le joueur ne sélectionne le mode facile automatiquement car le bouton cliquable se trouve au même endroit que le bouton de mode solo.

Les difficultés rencontrées :

Nous avons rencontré quelques difficultés par moment mais rien d'insurmontable, dans l'ensemble tout ce qui a été entrepris a été terminé.

Lorsque nous nous trouvions fasse a une difficulté nous faisions d'abord un programme sans rapport avec le Projet jusqu'à ce que celui-ci fonctionne pour l'implémenter ensuite au code du projet.

Différentes difficultés rencontrées :

- Commençons par la plus grande difficulté du projet, celle de faire deux lignes d'allumettes.
  Nous avons décidé de faire le code de sorte à avoir deux lignes d'allumettes et ainsi valider la variante du projet tel que nous l'avions comprise. C'est finalement autour de cette variante que tout le code s'est écrit. Le fait de dessiner deux rangées d'allumettes nous a obligés à gérer beaucoup de choses comme le saut de ligne et la suppression des allumettes de la deuxième ligne en fonction de la première. Cette variante est donc à l'origine d'un alourdissement considérable du code du projet.

- Une des premières difficultés a été de définir a qui le tour de jouer.
  En effet nous avons attaqué cette partie du projet alors qu'il était déjà possible de supprimer les allumettes. Peut-être qu'en ayant commencé cette partie au début du projet cela aurait été plus simple. Nous avons finalement décidé d'utiliser un registre pour enregistré le tour du joueur et gérer tout ce qui allait avec en fonction de ce dernier.

- L'écriture du code du score des joueurs qui ne change pas même lors du lancement d'une nouvelle partie a fait partie des défis intéressants à relever. En effet le simple fait de créer un compteur pour atteindre 99 ne nous est pas apparue immédiatement et la version finale du compteur est loin d'être optimisée et prend malheureusement beaucoup de place.

- L'une des dernières grandes difficultés a été de gérer les actions de l'ordinateur à la place du joueur 2, en effet cela paraissait pourtant simple en aperçu mais il s'en est avéré autrement pour trouver une place dans un registre non utilisé afin de sauvegarder l'action du joueur pour la modifier.

- La recherche du moyen d'intégrer le son nous a également pris un moment, en effet les docs sur l'assembleur 68k afin de trouver la fonction ne sont pas si courant sur internet.

Conclusion, améliorations :

Pour conclure, nous dirons que le code aurait pu être fait de différentes manières, en effet nous aurions pu utiliser plus d'adressage indirect.

Lors d'une partie, nous aurions également pu effacer tout l'écran et le réafficher sans les allumettes supprimées, cependant le code aurait était plus conséquent.

Finalement le jeu que nous avons développé comporte 3 modes de jeux différents :

- Solo en mode facile dans lequel l'ordinateur effectue les mêmes actions que le joueur.

- Solo en mode impossible dans lequel l'ordinateur effectue un complément a 4 de l'action du joueur : Lorsqu'un joueur sélectionne 1 allumette par exemple, si l'autre joueur en sélectionne 3 il fait ce qu'on appelle le complément à 4 (1+3=4).
  Si le premier joueur en sélectionne 2, cela sera 2 et enfin ça 1 si le premier joueur en sélectionne 3. En utilisant cette méthode le second joueur (qui est l'ordinateur) ne peut pas perdre.

- multijoueur(le mode de base demandé) dans lequel le programme attend la saisie d'action de deux joueurs différents sans s'arrêter jusqu'à ce que l'un des deux joueurs soit déclaré gagnant.

Ce jeu, bien que très complet et fonctionnel, aurait pu être amélioré dans les axes suivants :

- Tout d'abord en ajoutant une sauvegarde des scores dans un fichier, cela aurait permis au joueur d'avoir un historique des résultats pour une meilleure expérience de jeu.

- Faire varier les règles selon une variante différente du jeu, par exemple permettre au joueur qui retire l'une des dernières allumettes d'une des deux lignes de gagner. Ainsi si un des joueurs prend la dernière allumette d'une ligne il gagne, cela aurait permis de développer l'aspect stratégique du jeu.