Rapport SAE 1.01_2022

Projet réalisé par :

Kohsey Dufour
Arthur Chauveau

Sommaire

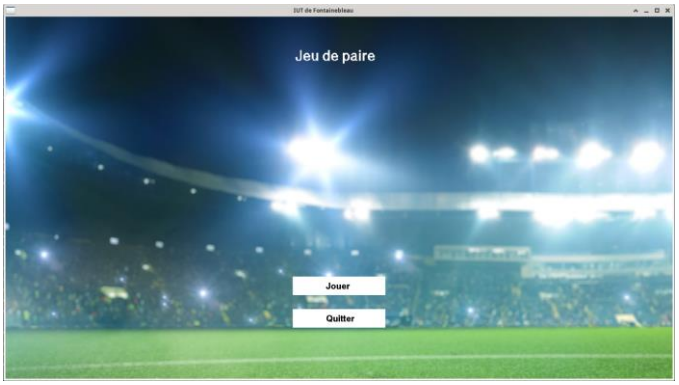
Introduction

Introduction générale du sujet :

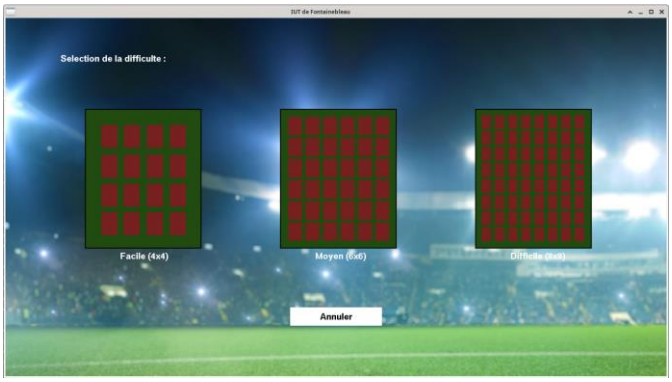
Le projet consiste à créer un jeu de paires grâce au langage de programmation C. Le jeu de paires est un jeu qui se compose de paires de cartes portant des illustrations identiques. Toutes les cartes sont disposées aléatoirement sur la grille de dos. A chaque tour, le joueur retourne deux cartes de son choix. Si les deux cartes sont identiques, elles restent faces découvertes. Sinon, si les cartes ne sont pas identiques, elles se retournent faces cachées à leur emplacement de départ après 1 seconde.
Le jeu se termine quand toutes les paires de cartes ont été découvertes.

Fonctionnalités du programme

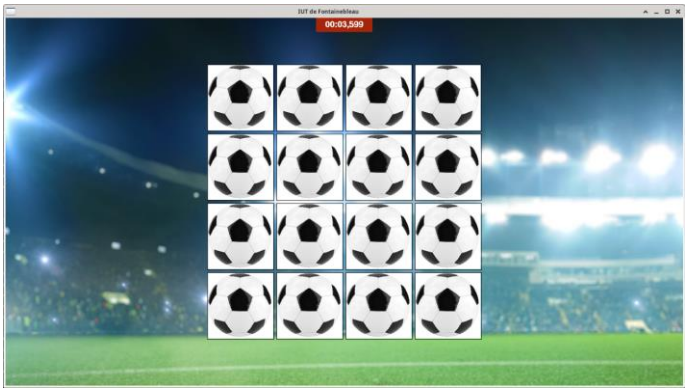
En exécutant le programme, ce dernier affiche cette fenêtre. Vous pouvez choisir de jouer ou de quitter le jeu.



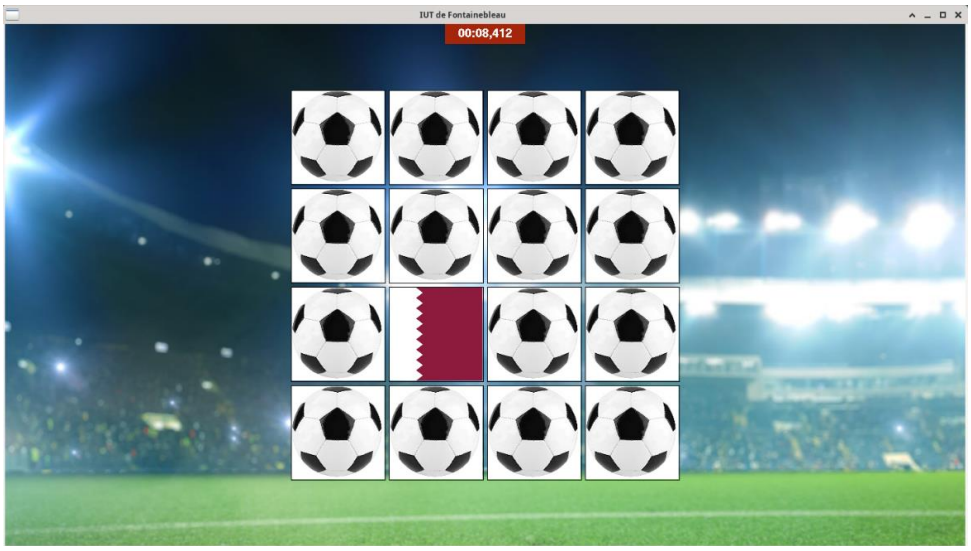
En cliquant sur jouer, trois difficultés vous sont proposées. Le bouton Annuler vous permet de revenir sur l'écran d'avant.



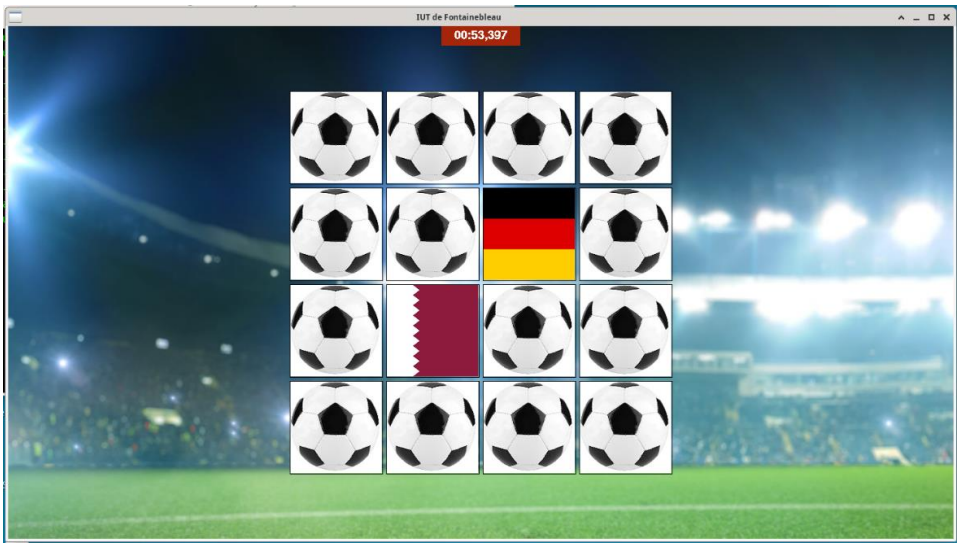
Une fois le jeu lancé, le chrono se lance automatiquement.



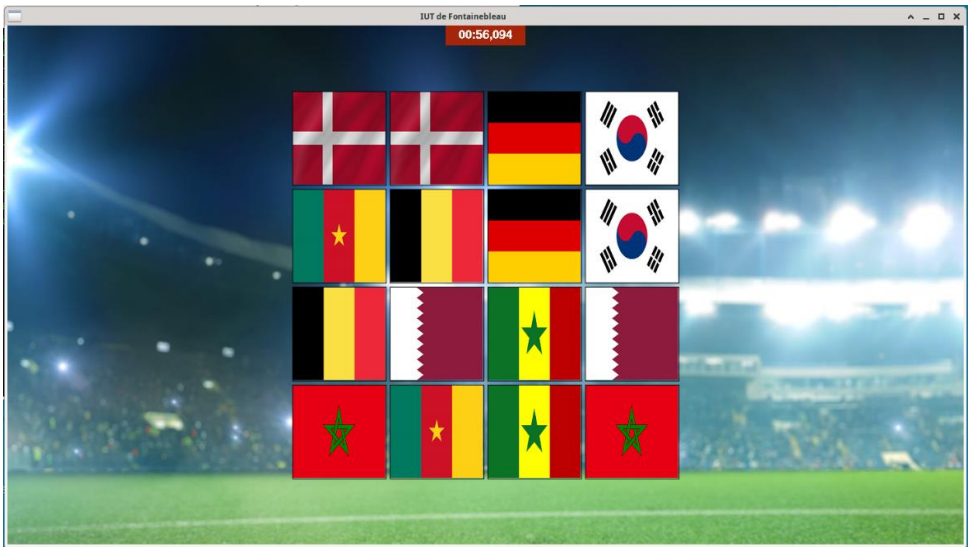
Les cartes peuvent être retournées par un clic de souris.



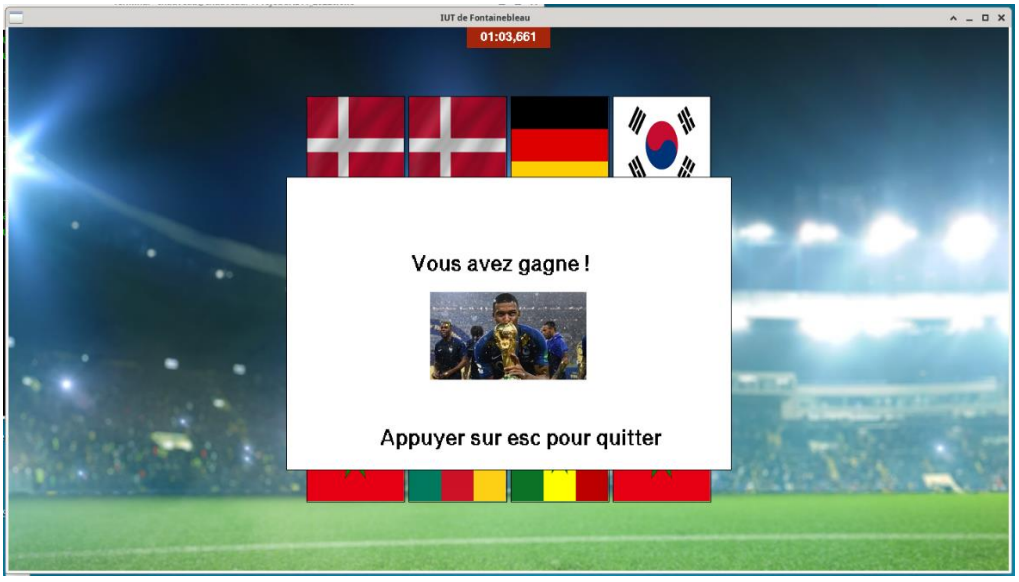
Le temps de mémorisation dure une seconde si les cartes retournées ne sont pas identiques.



Si l’envie vous prend de tricher, presse la touche ‘t’ pour tricher. Toutes les cartes seront visibles jusqu’à ce que vous repressiez la touche ‘t’. Le chrono sera interrompu pendant le mode triche.

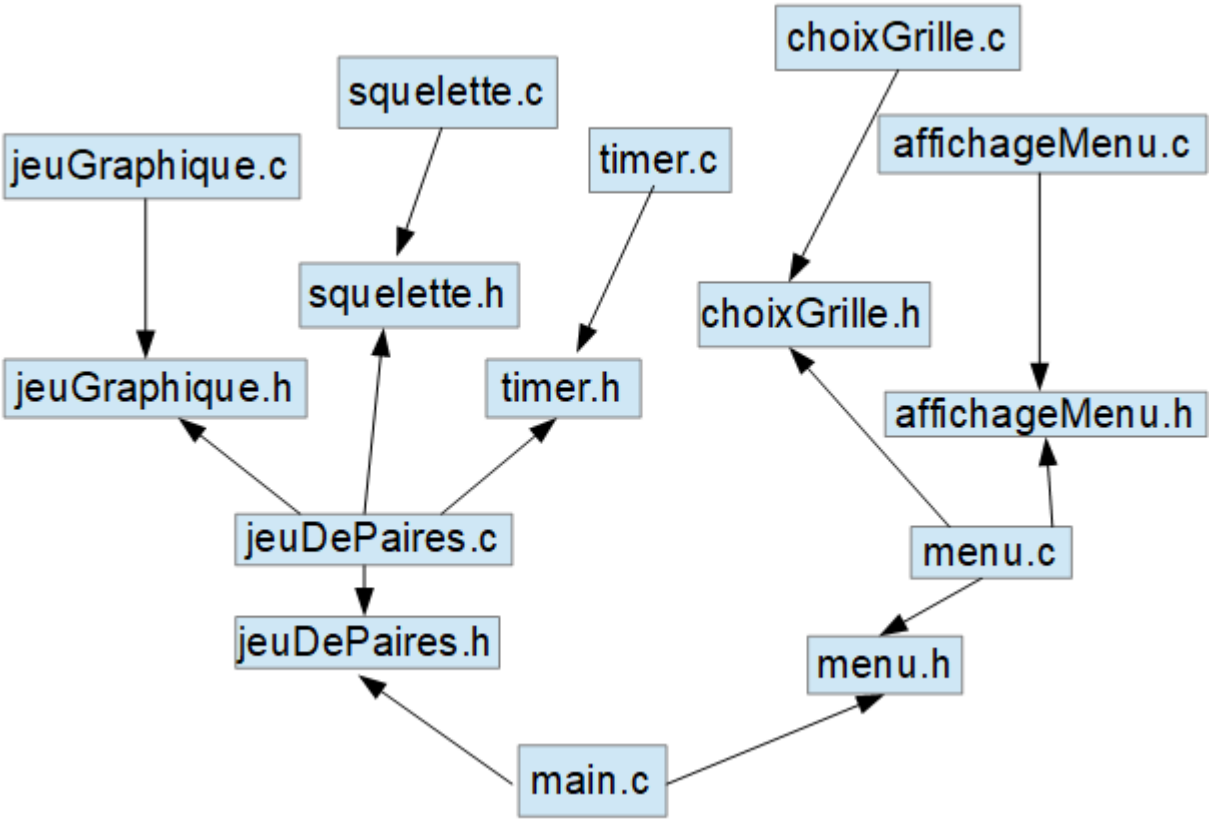


Le jeu se termine lorsque toutes les paires auront été trouvées (avec ou sans triche 😊). Le chrono s'arrêtera et un écran de victoire s'affichera. Vous pourrez appuyer sur la touche d'échappement pour quitter le programme.



Structure du programme

Représentation graphique des différentes dépendances.



Une fois le programme lancé, en entrant « make run » dans le terminal, main est exécuté. La fonction menu_principal est appelée. La fonction menu_principal appelle toutes les fonctions nécessaires pour afficher le menu : Bouton jouer, quitter, le choix de la difficulté.

Si l'utilisateur clique sur quitter, alors la fonction menu_principale renvoie un nombre négatif, la fenêtre se ferme et le programme est terminé.

Si l'utilisateur clique sur un mode de difficulté, la taille du jeu est renvoyée et la fonction jeu se lance.
La grille est générée à partir du tableau des ids du tableau tableauCartes de struct Carte. Les cartes sont toutes de dos (squelette.c), confer la fonction initialiser_visibilite. Ensuite, le graphique du jeu est affiché (jeuGraphique.c).

Dans la boucle while principale qu'est la fonction jeu, le programme attend qu'il ait un clique ou que le bouton 't' soit pressé.

Si le clic est sur une carte, la carte est dévoilée. Si le deuxième clic est sur une autre carte, la fonction est_ce_une_paire vérifie que les ids correspondent ou non, dévoilent les cartes si elles le sont.
Avec le fonction gagnant, le programme vérifie si toutes les cartes ont été retournées.

Si c’est le cas, l’écran de victoire est affiché.

A tout moment dans la boucle while, si la touche ‘t’ est pressée le mode tricheur se lance grâce à la fonction mode_tricheur, et affiche toutes les cartes sans modifier le champ visibilite du struct Carte.

Données qui représentent la grille de jeu

Dans le fichier squelette.h, nous avons défini un struct « Carte » qui a pour champ un premier int id qui représente l'identité d'une Carte. Chaque id étant associé à une image, c'est ce dernier qui permet de savoir quelle image afficher lorsqu'il y a un clic par exemple.
Ayant 32 images uniques, les ids sont compris entre 1 et 32 les deux inclus. Ainsi, toutes les images sont utilisées dans le mode de difficulté difficile (8x8).

Le deuxième champ est un int visibilité qui nous indique si la Carte est de dos ou de face :

- -1 si la Carte est de dos
- -0 si la Carte est de face

Algorithme de remplissage de la grille

L'algorithme de remplissage de la grille se situe dans le fichier squelette.c, dans la fonction generations_ids.

Dans une boucle for, on incrémente i de 1 à partir de 0, tant que i ne dépasse pas l'id 32 et que le tableau n'est déjà pas rempli (im étant le nombre de paires de cartes qui sont déjà placées et T*T/2, le nombre de paires de cartes à générer).

A chaque début de la boucle for :
-Le nombre de paires de cartes qu'il est possible de générer – i est (re)calculé (randomN).
-Le nombre de paires de cartes uniques restantes à placer est (re)calculé (randomCartesRestantes).

De plus, on verifie :
→Si le nombre aléatoire compris entre 0 et le nombre de paires de cartes qui est possible de générer - i, et que le nombre aléatoire est inférieur au nombre de cartes uniques restantes à placer. Alors l'id i + 1 est placé deux fois dans le tableau Tampon et im est incrémenté de 1. La boucle for continue
→Sinon, la boucle continue.

Dans la fonction affectation_aleatoire_position, les paires d'ids uniques générés par la fonction précédente dans un tableau tampon sont affectées aléatoirement dans le tableau principale (tableauCartes).
La fonction est triviale, un nombre aléatoire compris entre 0 et la taille du tableau est générée, on l'appellera l'index aléatoire (randomPosition). Si l'id associé à l'index n'as pas encore été placé dans le tableau principal, alors l'id est placé dans ce dernier. Sinon, un index aléatoire sera regénéré.

Détail important : moins il y aura d'ids à placer, plus la recherche d'une position pour l'id sera long.

Conclusions personnelles

DUFOUR Kohsey :

J'ai trouvé ce projet plus long que ce que je pensais. Heureusement, avec mon camarade, nous nous sommes mis assez rapidement dans le projet. Notre code en lui-même est je pense loin d'être parfait. Du moins, il est fonctionnel et lisible.

Pour ma part, le projet m'a permis d'approfondir mes connaissances en algorithmie. Je fais particulièrement référence à l'algorithme qui permet de remplir la grille de paires de cartes uniques.

Le fait d'avoir réalisé ce projet à deux m'a permis d'avoir une approche de développement différente de celle d'être tout seul. Nous avons dû nous répartir le travail et communiquer sur ce que nous faisons afin de ne pas à avoir de conflit de version avec git. Cela s'est très bien passé par ailleurs.

A l'avenir, j'aimerai réaliser de plus grands projets avec une équipe plus étendue afin d'améliorer mes compétences de communications et de développements. Il ne suffit pas d'être bon en code, il faut surtout avoir une bonne communication. Cette dernière a un effet de coefficient multiplicateur sur l'avancée/technique du projet, bien plus que d'être bon en code.

Voilà ce que le projet m'a appris !

CHAUVEAU Arthur :

Pour ma part, ce projet m'a permis de me rendre compte qu'il faut commencer un projet dès qu'on nous le donne car celui-ci sera souvent très long comme le projet que l'on a dû faire, c'est-à-dire le jeu de paires. Comme l'a dit mon camarade, même en ayant commencé rapidement le projet, nous avons trouvé que le projet était plus chronophage que l'idée que l'on se faisait au début.

Ce projet m'a permis d'acquérir de nouvelles connaissances en programmation et particulièrement la programmation avec une bibliothèque graphique. Ça change des TP avec un affichage pas très jolie sur le terminal. De plus, pour avoir fait le Makefile, ce dernier prend tout son sens dans la réalisation d'un tel projet.

Enfin, il m'a aussi permis de me rendre compte qu'un projet sera souvent plus facile avec une équipe ou en groupe et en étant organisé que tout seul. Le démarrage peut être long car nous devons nous concerter sur qui fait les choses. Mais cela est nécessaire. Le temps est vite rattrapé avec une bonne organisation et l'on est plus efficace !