

Rapport du projet d'Algorithmique

Développement des jeux de dominos et triominos

Réalisé par :

Michel Steve ANANGA GUEUJUI, Raffaele GIANNICO,
Rayane HAMMOUMI, Arnaud HENCHES, Bochra HOUISSA, Maud
LESTIENNE et Heni LOUDHAIEF étudiant en IATIC 3

Dhekra ABOUDA - Encadrant

Projet IATIC 3 effectué du 20/12/2021 au 27/01/2022

Sommaire

Introduction	4
Présentation du travail effectué	5
Partie conception	5
Modèle de conception	5
Répartition des tâches	6
Diagramme de Gantt	6
Environnement de développement	6
Partie réalisation	8
Fonctions de la librairie graphique	8
Menus	8
Dominos	11
Structures	11
Modèle	12
Contrôleur	13
Triominos	13
Contrôleur	13
Modèle	14
Vue	15
Conclusion	17
Lexique	18

Table des figures et tableaux

Figures

Figure 1 : Modèle MVC	5
Figure 2 : Diagramme de Gantt	6
Figure 3 : Capture d'écran du menu d'accueil	9
Figure 4 : Capture d'écran du menu pour le choix du nombre de joueurs	9
Figure 5 : Capture d'écran d'annonce	10
Figure 6 : Capture d'écran du menu pour le choix du jeu et de la variante	10
Figure 7 : Capture d'écran du jeu de domino	14
Figure 8 : Capture d'écran du jeu de triominos	18

Tableaux

Tableau 1 : Répartition des tâches pour le projet	6
---	---

Introduction

L'univers des jeux est généralement attribué aux applications de type multimédias ayant un rôle divertissant et jouables sur des supports variés. Ces derniers peuvent être de gros ordinateurs ou bien de petits appareils portables tels que les consoles, les ordinateurs personnels ou encore les smartphones.

Pour acquérir l'aptitude à coder un programme permettant de modéliser deux jeux: Dominos et Triominos, il est indispensable de mener des recherches approfondies pour développer une vision globale sur ce domaine diversifié.

La première partie de notre rapport portera sur l'équipe et le projet. La seconde partie, s'articulera sur la conception. La troisième partie, nous mettrons en évidence la réalisation tout en explicitant le développement de ces différentes fonctionnalités que nous avons proposées suite à nos recherches, et enfin nous présenterons les différentes maquettes.

Présentation du travail effectué

I. Partie conception

Dans la partie conception, nous allons décrire comment nous nous sommes organisés puis nous expliquerons quel langage et quelles librairies nous avons utilisé. Nous avons décidé de la répartition des tâches, grâce au modèle de conception que nous avons choisi lors de la première réunion.

1. Modèle de conception

Pour notre projet, nous avons opté pour l'utilisation du modèle MVC (Modèle - Vue - Contrôleur).

Ce modèle est un motif d'architecture d'interface graphique qui consiste à faire le lien entre l'interface utilisateur et les modèles de données.

- Le modèle : contient les données à afficher.
- La vue : contient les différents éléments de l'interface graphique.
- Le contrôleur : permet la communication entre la vue et le modèle, il contient donc la logique des actions effectuées par l'utilisateur.

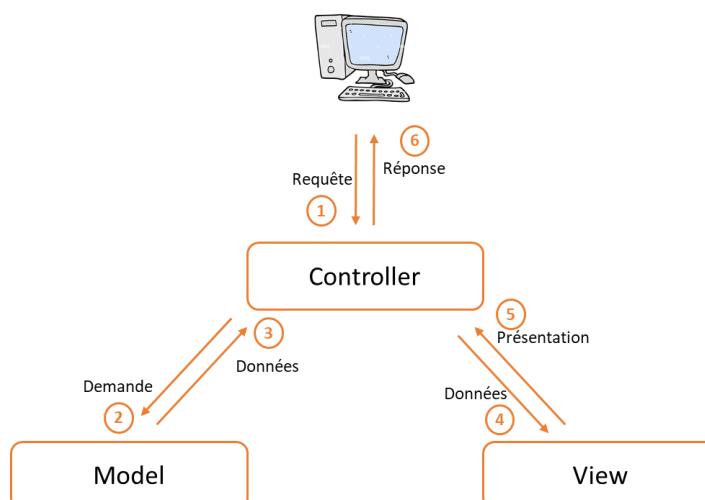


Figure 1 : Modèle MVC

2. Répartition des tâches

	Modèle	Vue
Dominos	Raffaele, Rayane	Heni, Rayane
Triominos	Arnaud, Bochra	Maud, Michel

Tableau 1 : Répartition des tâches pour le projet

Pour la partie contrôleur, chaque groupe travaillant sur un jeu créera le contrôleur de son jeu à la fin en concertation avec les autres membres travaillant sur le même jeu.

3. Diagramme de Gantt

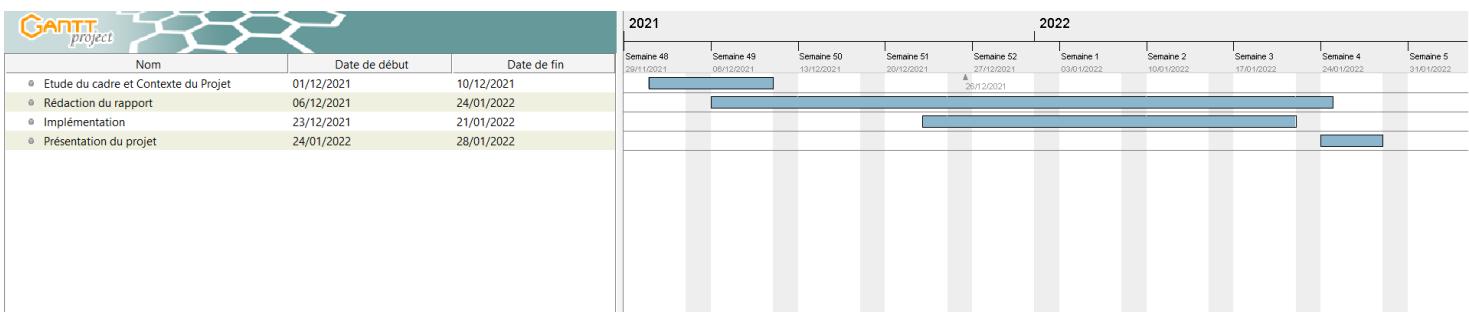


Figure 2 : Diagramme de Gantt

4. Environnement de développement



Langage:

Pour la réalisation de notre jeu, nous allons utiliser le langage C comme langage de programmation.



Bibliothèque :

Nous aurons recours à la bibliothèque basée sur SDL pour pouvoir avancer sur la partie graphique de notre projet.

SDL est une bibliothèque logicielle libre qui aide à créer des interfaces graphiques multimédia (contenant des audios, des images..). Pour construire notre librairie nous avons emprunté certaines fonctions de la librairie fournie en début d'année.



Outils:

Pour sauvegarder les informations des joueurs et leurs scores tout au long du jeu , nous mettons en place des fichiers stockant toutes les données nécessaires.

Pour conclure, nous avons décrit l'organisation en explicitant la répartition des tâches et la planification du projet. L'environnement de développement a été présenté ainsi que la méthode de conception utilisée.

II. Partie réalisation

Dans la partie réalisation, nous allons principalement expliquer comment nous avons codé la solution à présenter au client. Tout d'abord cela passe par une étape de coordination sur les briques de base à utiliser, en détaillant des fonctions, et sur le rendu final.

1. Fonctions de la librairie graphique

Mais pour cela, nous devons commencer par mentionner le rôle des fonctions de la librairie graphique utilisées.

ouvre_fenetre() et ferme_fenêtre permettent d'ouvrir/fermer une fenêtre

rempli_ecran() permet de colorier une fenêtre d'une couleur souhaitée

La fonction affiche_image affiche une image dont on précise le chemin à une coordonnée (x;y) souhaitée.

affiche_texte() permet d'afficher une chaîne de caractère à une coordonée souhaitée.

les fonctions dont le nom est du type dessine_forme(), permettent de dessiner la forme dans le nom de la fonction

actualise_affichage(): sans appeler cette fonction, toute fonction d'affichage utilisant directement ou indirectement SDL ne fonctionnera pas.

attend_clic() attend qu'on clique sur l'écran et renvoie la coordonnée du clic

Ces fonctions sont appelées systématiquement et lorsqu'on fait appel à elle, c'est facile à deviner. Nous ne les mentionnerons donc plus dans ce rapport.

2. Menus

Lorsqu'on lance le programme, un menu d'accueil apparaît (figure suivante). Il attend que l'utilisateur appuie sur le bouton "Jouer".

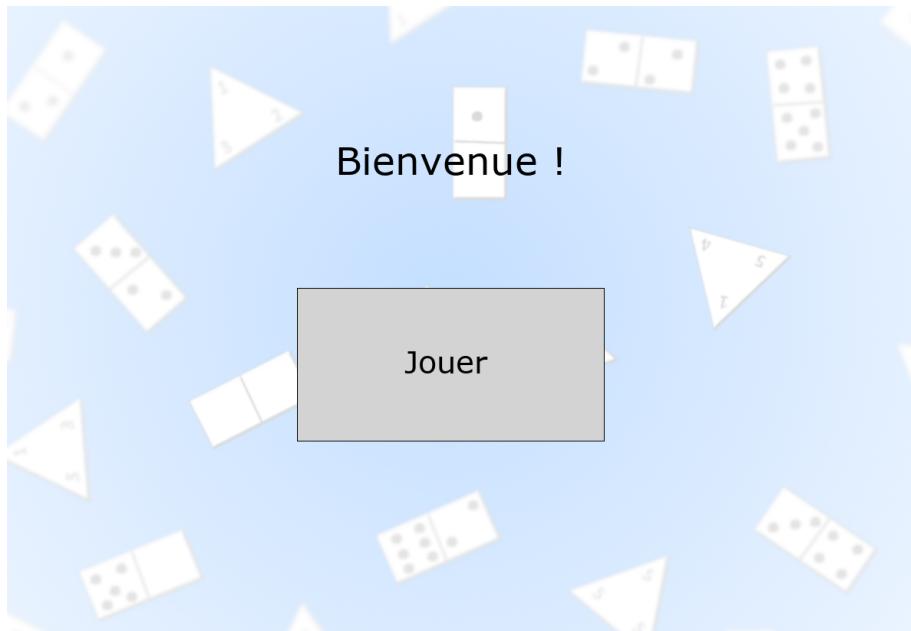


Figure 3 : Capture d'écran du menu d'accueil

Ensuite, il est demandé à l'utilisateur de choisir le nombre de joueurs humains et le nombre de joueurs ordinateurs. Le nombre minimal de joueurs est de deux, avec au minimum un joueur humain. Le nombre de joueurs total ne peut pas dépasser quatre. Le choix se fait en cliquant sur des boutons (voir figure suivante).



Figure 4 : Capture d'écran du menu pour le choix du nombre de joueurs

Une image s'affiche (voir figure suivante) pour indiquer au joueur humain d'entrer son pseudo.



Figure 5 : Capture d'écran d'annonce

Le choix du jeu se fait dans le dernier menu (voir figure suivante). L'utilisateur est invité à choisir le jeu auquel il souhaite jouer ainsi que sa variante. Après ce choix, le jeu choisi se lance.

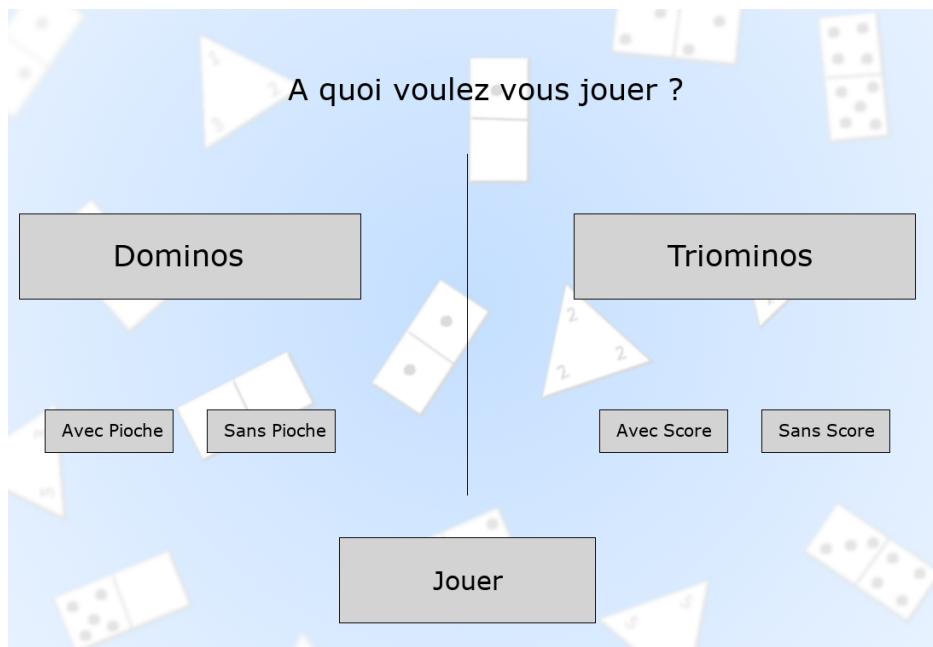


Figure 6 : Capture d'écran du menu pour le choix du jeu et de la variante

A la fin du jeu, le classement de la partie est affiché. L'utilisateur a la possibilité de quitter le programme ou bien de rejouer. S'il choisit la deuxième option, il revient au menu de la figure 6, pour choisir le jeu auquel il souhaite rejouer.

Nous allons à présent vous expliquer le fonctionnement du code source de chaque jeu, en commençant par le jeu Dominos.

3. Dominos

Dans cette partie, nous allons d'abord vous présenter les différentes structures utilisées. Par la suite, nous détaillerons le fonctionnement du modèle. Puis, nous expliquerons les fonctions d'affichage. Enfin, nous verrons comment la fonction contrôleur assure le bon déroulement d'une partie grâce aux fonctions d'affichage et au modèle.

1. Structures

Commençons par présenter les différentes structures utilisées.

DOMINO: 3 champs, 2 qui contiennent chacun la valeur d'un côté du domino, l'autre qui contient l'orientation horizontale ou verticale du domino.

Cette structure est d'abord utilisée pour nos deux variables globales: plateau[] et pioche[].

Ce sont des tableaux contenant des DOMINO.

plateau[] contient les dominos joués par les joueurs, tandis que pioche[] contient les dominos de la pioche.

Cette structure est utilisée très régulièrement dans le code.

POINT: utilisée pour stocker les coordonnées d'un point dans la fenêtre.

1 champ d'entier x, 1 champ d'entier y

Utile pour les fonctions d'affichage notamment, qui vont dessiner ou afficher une image à une coordonnée (x;y) dans la fenêtre.

COORDONNEES: on utilise cette structure pour stocker l'emplacement des deux dominos aux extrémités du jeu. D'une part, dans plateau[] grâce au champ indice et au champ colonne. D'autre part dans la fenêtre, grâce au champ POINT.

AIDE_PLACEMENT: aide la fonction responsable de stocker dans plateau[] ou non le domino qu'un joueur veut jouer.

Le 1er champ nommé compatible est un booléen indiquant si le domino choisi a une valeur en commun avec une extrémité du jeu.

Le 2e champ, extrémité, est une enum. Il indique si les deux extrémités sont compatibles, si aucune n'est compatible. Sinon, dans le cas où une seule extrémité est compatible, si celle compatible est la plus à gauche, à droite, en haut ou en bas.

JOUEUR: nous est utile pour stocker des informations sur les joueurs.

Le 1er champ est pseudo[] qui contient des char. On va stocker dans ce tableau le pseudo d'un joueur.

Le 2e champ, mainJoueur[], contient les dominos de la main d'un joueur.

Le 3e est un entier qui contient le score d'un joueur.

2. Modèle

3.

Dans cette partie, nous allons détailler toutes les fonctions que nous avons utilisées dans le modèle.

Pour commencer, nous initialisons plateau[] avec des dominos que l'on va appeler : "domino vide", la valeur de ce domino est |-1 -1|. De même, on initialise les joueurs en mettant des dominos vide dans le tableau des mains de chaque joueur qui se trouve dans infos_joueur.

Ensuite, vient la fonction qui génère la pioche, grâce à une double boucle for, on peut créer les 28 dominos et les stocker dans le tableau pioche.

Avant de distribuer les dominos, on appelle la fonction determine_nb_dominos_main qui va déterminer le nombre de domino à donner (6 ou 7 selon le nombre de joueurs).

Selon le nombre de dominos à donner, la fonction distribue_premier_domino appelle 6 ou 7 fois la fonction pioche_un_domino.

Cette dernière permet de récupérer un domino au hasard de la pioche, tout en le supprimant de celle-ci (en remplaçant le domino par un domino vide). On ajoute ensuite le domino pioché à une case vide de la main du joueur.

Les dominos étant distribués, nous pouvons à présent définir le premier joueur qui va jouer (fonction definit_premier_joueur). Le premier joueur qui va jouer est celui qui possède le plus grand double. On va parcourir la main de chaque joueur en comparant chaque double trouvé pour savoir qui a le plus grand (en mettant à jour à chaque fois le plus grand trouvé). Si personne n'a de double, on va vérifier celui qui a le domino le plus fort(|6 5|,|6 4|, etc...). Dans le cas où les informations du joueur qui commence ne sont pas déjà dans la première case du tableau, on réorganise JOUEUR infos_joueur[]. Pour cela, on échange les informations de joueur qui sont dans la première case avec celles du joueur qui doit commencer.

Le programme récupère le domino choisi par l'utilisateur grâce à la fonction recuper_choix_domino_main. Cette fonction fait appel à la fonction gere_clic qui nous permet de savoir où-est-ce que l'utilisateur a cliqué (s'il a cliqué sur un domino de sa main, sur la pioche ou dans le vide par exemple). Si l'utilisateur clique sur un domino on renvoie donc ce domino. S'il clique sur la pioche ou sur "passer son tour", on renvoie un domino spécial avec comme valeur |-2 -2|. Comme cela, il est possible de distinguer ce que l'utilisateur a choisi.

Après avoir récupéré le domino que l'utilisateur a choisi, il reste à appeler la fonction place_domino qui va d'abord vérifier si ce domino est égale à la valeur d'une extrémité (Gauche ou droite) sur le plateau (grâce à la fonction verifie_compatibilite_domino).

Si ce domino est compatible, on le place sur le plateau et dans la vue grâce à la structure COORDONNEE. qui nous indique l'emplacement des dominos aux extrémités.

Désormais les joueurs peuvent jouer, la fonction joue_IA, prend le premier domino compatible dans sa main et si celui-ci est compatible avec une des extrémités il le place.

Si l'IA ne peut pas jouer alors il passe son tour tout seul ou pioche tout seul jusqu'à ce qu'il ait un domino compatible (selon la variante). On renvoie une valeur précise pour distinguer le fait de jouer et de piocher/passer son tour.

La fonction joue_joueur est presque identique à celle de joue_IA sauf que cette fonction appelle recuper_choix_domino_main pour que l'utilisateur choisisse soit de jouer un domino en particulier, soit de piocher, soit de quitter la partie en appuyant sur ce qui est affiché.

La fonction verifie_gagnant vérifie en premier lieu s'il quelqu'un a posé tous ses dominos. Elle vérifie par la suite si les joueurs peuvent continuer à jouer. Puis, si personne ne peut jouer, elle vérifie s'il y a un gagnant ou égalité.

Pour ce faire, elle appelle notamment la fonction verifie_compatibilite_main qui va vérifier, s'il y a au moins un joueur qui à un domino compatible ou que l'on peut encore piocher (variante avec pioche). Dans ce cas là, on renvoie -1.

Si il n'y a pas de coup compatible alors la fonction verifie_gagnant vérifie qui à le moins de domino dans sa main. Si les joueurs ont le même nombre de domino, elle indique qu'il y a une égalité. Sinon, si un joueur n'a plus de domino dans sa main, on renvoie le numéro du joueur. C'est le gagnant.

4. Vue

Pour ce jeu de dominos, nous avons utilisé plusieurs images:

- plusieurs images découpées d'un tapis de jeu
- des images de dominos répartis dans 4 dossiers (2 dossiers d'images horizontales dont 1 qui contient les mêmes images mais pivotées à 180 degrés. Pareil pour les 2 dossiers d'images verticales.)
- des images de dominos face cachés pour la main de l'IA (dont on ne doit pas voir les dominos) et pour la pioche. Elles se démarquent pas la présence du logo de l'ISTY dessus

Le paragraphe suivant vous explique le rôle de chaque fonction d'affichage.

affiche_fond() colorie d'abord la fenêtre en blanc. Puis, elle affiche l'image du tapis de jeu. Elle n'est appelée qu'une seule fois dans le programme. En effet, la rappeler reviendrait à effacer tout ce qui a été affiché précédemment.

La fonction `affiche_domino` permet d'afficher un domino pris en entrée à la coordonnée souhaitée. L'image à afficher est choisie dans l'un des 4 dossiers en fonction des valeurs du domino pris en entrée et de son orientation.

En appelant `affiche_domino()` plusieurs fois, nous pouvons donc afficher la main du joueur. C'est ce que fait la fonction `affiche_main()`.

`affiche_interface()` se charge d'afficher une image du bas du tapis, le bouton "Quitter" et, selon la variante, le bouton passer son tour ou celui de la pioche.

`affiche_tour()` affiche le nom du joueur dont c'est le tour de jouer.

A l'aide de la fonction contrôleur et de ces fonctions d'affichage, nous obtenons l'affichage complet du jeu:

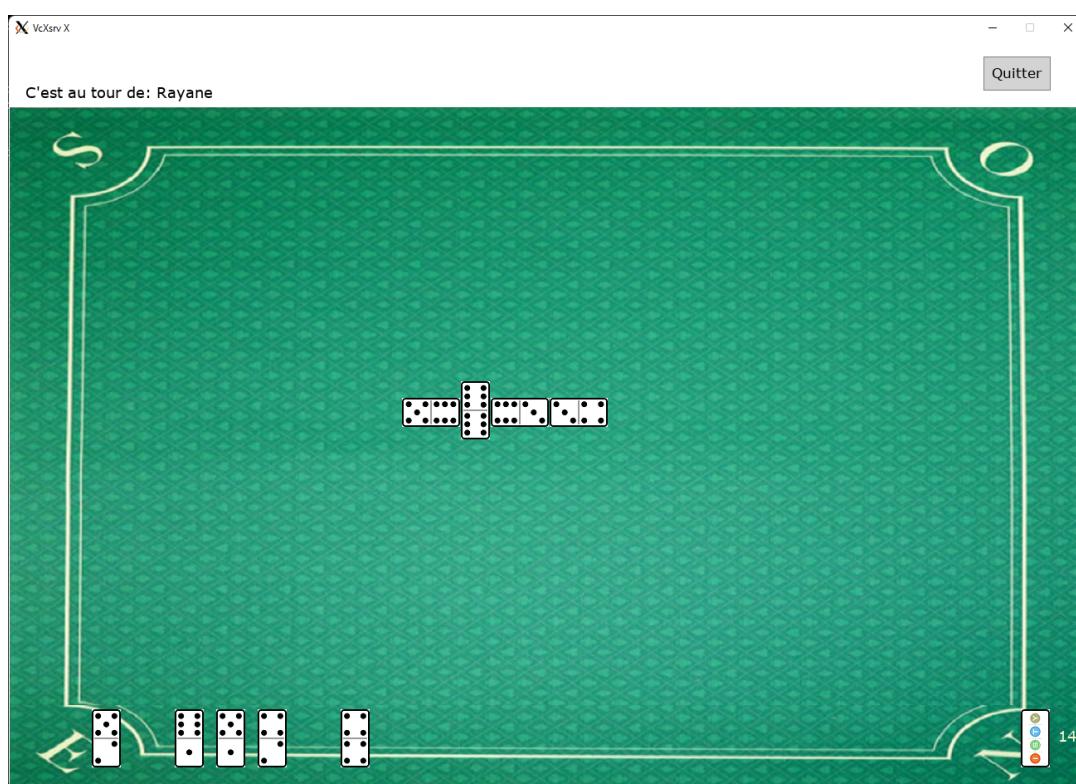


Figure 7 : Capture d'écran du jeu de dominos

5. Contrôleur

La fonction `main_dominos()` est l'unique fonction contrôleur dans le dossier domino.

Quand elle est appelée par le contrôleur de la partie menu, elle récupère la variante (avec ou sans pioche), le nombre de joueurs et les informations des joueurs contenu dans un tableau de JOUEUR.

Ensuite, le contrôleur initialise l'affichage de l'interface graphique du jeu domino en appelant affiche_fond(), affiche_interface() et affiche_tour(). Le modèle est initialisé en appelant les fonctions initialise_plateau(), initialise_main(), distribue_premiers_dominos() definit_premier_joueur().

La partie commence, et on ne s'arrêtera pas de jouer tant que la condition de fin de partie est vraie (grande boucle while).

On affiche tout d'abord une première fois la main du joueur actif.

On entre à présent dans une autre boucle do while. On va boucler jusqu'à ce que le joueur ait fini son tour. C'est-à-dire, jusqu'à ce qu'il ai effectué un choix valide (ex: jouer un domino valide, piocher, passer son tour ou quitter).

Puis, le choix du joueur est effectué grâce à l'appel de la fonction joue_joueur pour faire jouer le joueur. Si le joueur a pu placer un domino, alors il est affiché sur le tapis de jeu grâce à un appel de affiche_domino(). Si le joueur a pioché, alors on rajoute le domino dans le tableau de la main du joueur.

Chaque tour de boucle, on vient appeler afficher_interface() afin d'effacer certains affichages du tour précédent. Par exemple, le fait d'afficher une image du tapis découpé permet d'effacer la main du joueur précédent. On vient aussi effacer l'affichage du nombre de dominos qui étaient dans la pioche le tour précédent.

La grande boucle while s'arrête lorsqu'une condition de fin de partie est vraie. Par exemple, si un utilisateur appuie sur le bouton quitter, ou si quelqu'un gagne (grâce à la fonction verifie_gagnant).

Dans ce cas, un écran de fin de partie s'affiche pendant un certain temps grâce à SDL_delay(), les scores sont écrits dans un fichier puis s'affiche le classement. Enfin, le menu nous propose de rejouer ou de quitter le programme.

4. Triominos

1. Contrôleur

On récupère les informations données par le contrôleur de la partie menu et comme la variante, le nombre de joueurs et le nombre d'ordinateurs et le pseudo des joueurs.

Puis vient une partie initialisation du modèle et l'affichage de l'interface graphique du triominos. On va lancer la fonction qui commence. Et puis le jeu commence. Avec une boucle while, tant que la partie n'est pas terminée le jeu continu et à chaque itération on change le joueur actuel. Si le joueur actuel est un ordinateur, alors on appelle la fonction jeu ordinateur si il peut jouer on place le triomino et on renvoie le score et sinon on le fait piocher jusqu'à ce qu'il puisse jouer et on lui enlève des points selon les règles. Si le joueur est humain alors on attend son clic. Selon les coordonnées du clic on récupère sur quelle partie de l'interface il a cliqué : dans sa main (sur un triomino), sur le plateau, sur la pioche, sur quitter si il veut quitter. Et si son clique est vraiment significatif alors on le traite. Sinon on redemande un clique. Et lorsqu'il a fini son tour, on affiche le tout de nouveau à l'utilisateur et les changements qu'il a effectué pendant son tour. Lorsque plus personne ne peut jouer on renvoie le gagnant et les scores.

2. Modèle

Pour pouvoir mettre en place le jeu triominos, nous avons commencé par la mise en place des différentes structures dont nous aurons besoin tout au long de l'implémentation de cette partie.

- **TRIOMINOS** : En commençant l'étude du projet et du développement des triominos, nous avons remarqué que les triominos sont uniques, et que les valeurs contenues dans chaque triomino sont ordonnées de la petite à la plus grande valeur dans le sens des aiguilles d'une montre. Nous avons donc pensé à représenter nos triominos sous forme de structure TRIOMINOS composée de 3 entiers : min (minimum), sec (second) et der (dernier), représentant les valeurs d'une tuile dans l'ordre croissant.
- **EMPLACEMENT** : c'est une structure contenant un triomino, une pointe et une direction. Ces trois variables décrivent la position d'une tuile sur la grille. Cela nous servira pour remplir le tableau d'emplacements représentant la grille du jeu dans la partie modèle. Ce tableau est en effet un tableau de deux dimensions de structures «EMPLACEMENT».
- **JOUEUR** : La structure joueur est une sorte d'identifiant de chaque joueur triomino. Cette dernière contient son pseudo sous forme de chaîne de caractères, sa main, qui est aussi une structure de triominos, son score cumulé à chaque coup et une variable booléenne "estHumain" qui vérifie si le joueur est humain ou si c'est un ordinateur.
- **MAIN_J_TRIOMINOS** : cette structure contient les triominos qui composent la main du joueur. Ils sont piochés en début de partie et gardés, ainsi qu'un entier représentant la taille de la main c'est-à-dire le nombre de triominos du joueur.
- **PIOCHE_TRIOMINOS** : une structure composée d'un tableau de tous les triominos du jeu (au début de la partie) et la taille qui présente le nombre de ces triominos.

Nous avons eu recours à la mise en place de quelques énumérés et ce pour faciliter la compréhension de plusieurs vérifications et tests, on cite:

- HEXAGONE : cet énuméré permet de vérifier s'il y a une forme d'hexagone en plaçant un triomino. (0 si il n'y en a pas, 1 si hexagone, 2 si double hexagone et 3 si triple hexagone)

Nous passons ensuite aux fonctions:

- Pour commencer le jeu, nous commençons par initialiser la pioche en créant l'ensemble des triominos.
- Pour l'initialisation du tableau d'emplacement, nous mettons tous les triominos à (-1), les directions alternées en partant du centre de la grille là où sera placé le premier triomino dirigé vers le nord
- Pour commencer le jeu, il est nécessaire de savoir quel joueur commence. Pour ce faire, nous avons développé une fonction qui distribue un pion pour chaque joueur et retourne le joueur ayant obtenu la somme de valeurs la plus élevée sur son triomino.
- Test fin : teste les possibilités de jeu de tous les candidats et si plus personne ne peut jouer renvoie Vrai.
- Rearrange main joueur : cette fonction nous a permis d'avoir les mains des joueurs toujours mises à jour. À chaque coup joué, la main du joueur concerné sera mise à jour et le triomino joué est donc enlevé de sa main.
- Placer un triomino dans la grille : c'est grâce à cette fonction qu'un coup est joué. Quand le joueur clique dans un emplacement souhaité, ses coordonnées sont récupérées, traduites en indices et entrées en paramètres de la fonction «placer_trio» qui vérifie la possibilité de placer le trio dans la case souhaitée du tableau d'emplacements en vérifiant les cases adjacentes si le triomino peut y être collé ou pas. Cette fonction fait donc appel à la fonction «verif_coup_valide» qui compare les valeurs du triomino à placer avec les valeurs du côté du triomino existant et renvoie un booléen comme valeur de retour.
- Jeu ordinateur : Pour que l'ordinateur puisse jouer, nous avons opté pour un parcours du tableau, ensuite les cases où les coups sont possibles. On les stocke dans un tableau, même pour les coups impossibles. Après le parcours de la grille, l'ordinateur choisit au hasard un coup du tableau des coups possibles et les place dans la grille. Cette fonction «jeu_ordinateur» fait appel à la fonction «trouve_coups_legaux», qui permettra la vérification des coups possibles et leurs faisabilités.
- Les scores des joueurs sont calculés au fur et à mesure du jeu et sont retournés à l'appel de la fonction «placer_trio» et/ou «jeu_ordinateur». En effet, à chaque coup valide, le score est incrémenté de la somme des valeurs du trio joué et de la bonification si une forme est obtenue. Ce calcul est fait suite à un appel des fonctions «est_pont» et «est_hexagone» qui, après chaque coup valide, vérifient si en plaçant le triomino on obtient une forme, et renvoie le bonus de chaque forme obtenue.

3. Vue

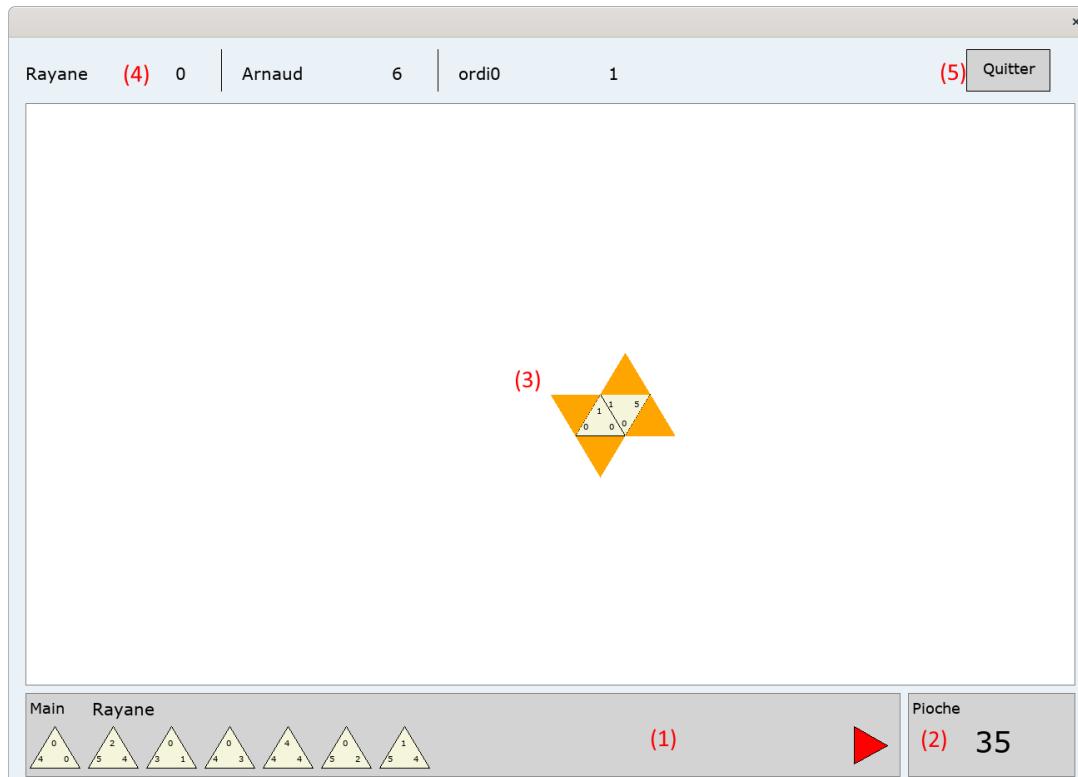


Figure 8 : Capture d'écran du jeu de triominos

L'interface du triominos se décompose en cinq parties principales.

La main du joueur courant (1), contient les triominos du joueur. La main peut contenir jusqu'à vingt-cinq triominos. Pour avoir accès à tous les triominos de sa main, le joueur peut naviguer grâce à la flèche rouge. Pour poser un triomino, le joueur clique sur celui qu'il souhaite placer, celui-ci s'entoure de rouge pour montrer qu'il a été sélectionné. Si le joueur ne veut pas ou ne peut pas jouer, il peut piocher.

La pioche (2) contient le nombre de tuiles restantes dans la pioche du jeu. Si le joueur clique dessus, un triomino s'ajoute à sa main, il peut alors tenter de le placer sur le plateau.

Le plateau (3) contient les tuiles posées et les emplacements disponibles (triangles oranges) pour placer un nouveau triomino. Une fois que le joueur a sélectionné une tuile dans sa main pour pouvoir le placer, il doit cliquer sur l'un des emplacements. Soit le triomino est posé, soit il reste dans la main du joueur si la position n'est pas bonne.

Les scores et les pseudos (4) des joueurs sont affichés. Les scores n'apparaissent que si la variante "avec score" a été choisie au début de la partie. Les scores évoluent au cours de la partie.

Le bouton quitter (5) permet à l'utilisateur d'arrêter la partie s'il le souhaite. Il aura le choix de rejouer ou bien de quitter le programme.

Conclusion

Ce projet a été mené dans le cadre d'une demande spécifique et fait également partie d'un effort continu de recherche impliquant différents groupes pour espérer fournir, dans un long terme, un programme permettant de modéliser le jeu Dominos et le jeu Triominos.

Grâce à ce projet, nous avons pu comprendre et expérimenter les différentes étapes de conception et de réalisation d'un programme, en commençant par l'analyse, en outre la programmation nous a permis d'améliorer nos connaissances du langage C.

En plus d'être un projet pédagogique il est aussi ludique et nous a donné beaucoup de liberté dans le code et dans la modélisation. C'est la première fois que nous travaillons en groupe sur un projet avec un but bien défini.

Ce projet nous a permis de consolider nos connaissances générales dans la gestion de projet et dans la programmation. Nous sommes satisfaits de ce que nous avons réalisé, les deux jeux fonctionnent bien malgré quelques petits défauts. Au niveau de la gestion du projet en équipe, nous avons réussi à bien nous répartir les tâches afin de réaliser nos objectifs dans les temps.

Lexique

Bibliothèque logicielle: est un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.

Booléenne: type de variable à deux états généralement notés *vrai* et *faux*, destiné à représenter les valeurs de vérité de la logique et l'algèbre booléenne.

Contrôleur : est un dispositif matériel ou un programme logiciel qui gère ou dirige le flux de données entre deux entités.

Énumérés: est un type de données qui consiste en un ensemble de valeurs constantes.

Ergonomie: Adaptation d'un environnement de travail (outils, matériel, organisation...) aux besoins de l'utilisateur.

SDL (Simple DirectMedia Layer): est une bibliothèque logicielle libre