

Morpion

PROJET PRÉ-TPI

Crée par :

Diogo Vieira Ferreira
Rue du Cheminet 2
1400 Yverdon-les-Bains

Réalisé au :

CPNV de Ste-Croix

1. Table des matières

1. Table des matières.....	1
2. Analyse préliminaire.....	4
2.1. Introduction	4
2.2. Organisation.....	4
2.3. Objectifs	4
2.3.1. Première version :	4
2.3.2. Deuxième version :	4
2.4. Planification Initiale	0
3. Analyse.....	0
3.1. MCD.....	0
3.2. MLD	0
3.3. Scénarios (première version).....	1
3.4. Choix Graphiques (1 ^{ère} version).....	4
3.4.1. Page principale.....	4
3.4.2. Choix pour une partie locale.....	4
3.4.3. Demande du nom	5
3.4.4. Partie	5
3.5. UML	6
4. Implémentation.....	7
4.1. DataBase	Erreur ! Signet non défini.
4.1.1. Création de la base de données	7
4.1.2. InsertScore	7
4.1.3. Limite de scores	7
4.1.4. ClearScores	7
4.1.5. ScoreList.....	8
4.2. Test Database.....	8
4.2.1. Initialisation	8
4.2.2. Vérification du dossier / fichier	8
4.2.3. Les insertions	8
4.2.4. 11 Insertions	8

4.3. Controller	9
4.3.1. topMenu	Erreur ! Signet non défini.
4.3.2. main_menu	9
4.3.3. game_int	9
4.3.4. Show_interface	10
4.3.5. AskUserName	10
4.3.6. PopUpUserName	11
4.3.7. ThinkingGame	11
4.3.8. Replay	11
4.4. Model	12
4.4.1. CheckGame	12
4.4.2. checkPossibilities	Erreur ! Signet non défini.
4.4.3. AI (intelligence artificielle en anglais)	12
4.4.4. SaveGame	12
4.5. NetworkCommunication	12
4.5.1. NetworkCommunication	13
4.5.2. SocketSender	13
4.5.3. SocketReader	13
4.6. Test Model	13
4.6.1. Cas d'une égalité	13
4.7. Dossier de réalisation	13
4.7.1. Outils / Applications	13
4.7.2. Fichiers / Dossiers	13
4.8. Erreur restantes	15
4.8.1. Choix de symboles	15
4.8.2. Partie sur le réseau	15
5. Conclusion	16
5.1. Objectifs	16
5.1.1. Atteints	16
5.1.2. Non-atteints	16
5.2. Points positifs	16

Table des matières

5.3. Points négatifs	16
6. Annexes	17
6.1. Résumé du Rapport.....	17
6.1.1. Situation de départ :	17
6.1.2. Mise en œuvre	17
6.1.3. Résultats.....	17
6.2. Journal de travail.....	18
6.3. Manuel D'installation.....	18
6.3.1. Récupérer le projet	22
6.3.2. Logiciel.....	22
6.4. Manuel D'utilisation.....	22
6.5. Sources / Remerciement	22
6.5.1. Sources.....	22
6.5.2. Remerciements	22

2. Analyse préliminaire

2.1. Introduction

Le jeu morpion, aussi appelé Tic-Tac-Toe, est un jeu de réflexion se jouant à deux. C'est un bon moyen de se préparer pour mon projet TPI qui consiste à créer un chat sur le réseau.

Le but de celui-ci est d'arriver avant l'autre joueur à aligner 3 mêmes symboles, que ce soit en diagonale, en vertical ou horizontal.

Dans ce jeu, le joueur devra choisir son symbole avant de commencer la partie. Suite à une partie, le score pourra être sauvegardé et visible depuis une autre fenêtre. Tant que les joueurs relancent une partie sans revenir au menu, le score sera incrémenté d'un pour le gagnant.

2.2. Organisation

Chef de Projet :

Nom, prénom : Andolfatto Frédérique
Email : FREDERIQUE.andolfatto@cpnv.ch
Numéro mobile : 077 206 66 45

Elève, Développeur :

Nom, prénom : Vieira Ferreira Diogo
Email : Diogo.VIEIRA-FERREIRA@cpnv.ch
Numéro mobile : 078 665 05 19

2.3. Objectifs

- ☐ Créer une interface « Friendly user », c'est-à-dire facile d'utilisation dès le premier regard.

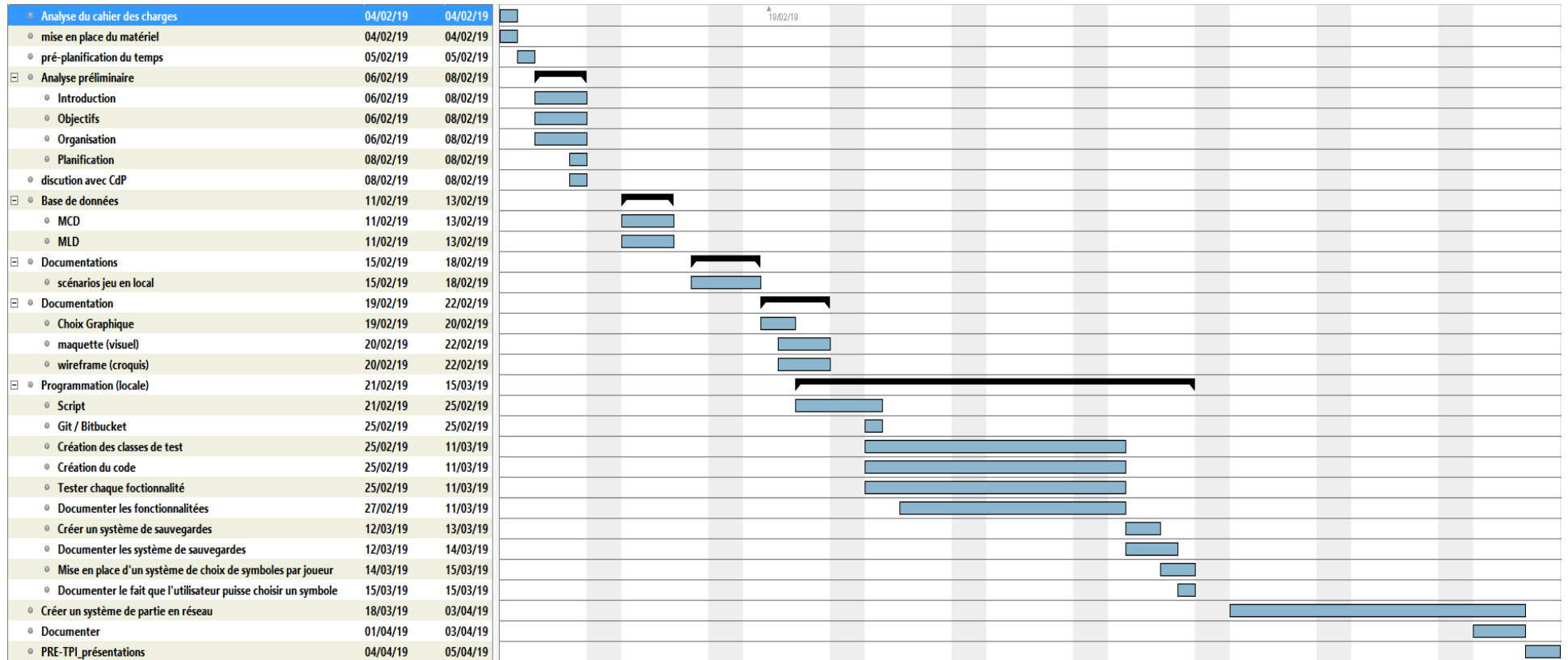
2.3.1. Première version :

- ☐ L'utilisateur peut choisir entre la « X », croix, ou le « O », rond.
- ☐ L'utilisateur peut cliquer sur une des cases.
- ☐ L'utilisateur peut abandonner la partie.
- ☐ Le score s'incrémente de 1 par nouvelle partie (quand l'utilisateur clique sur revanche et celle-ci est acceptée).
- ☐ Le score de l'utilisateur est sauvegardé.
- ☐ L'utilisateur peut choisir entre plusieurs symboles en plus du « X » et du « O ».
- ☐ L'utilisateur peut abandonner la partie.

2.3.2. Deuxième version :

- ☐ L'utilisateur peut jouer sur le réseau avec une autre personne.

2.4. Planification Initiale



3. Analyse

3.1. MCD

Score
Player01
Player02
ScoreP01
ScoreP02

Je suis parti sur un choix d'une seule table pour les raisons suivantes :

Etant donné que c'est un mini jeu, je ne pense pas garder plus de 10 scores.

De plus, il est possible d'avoir des personnes qui possèdent le même prénom que quelqu'un d'autre.

Je ne pense pas que ce soit utile de vérifier si le nom existe sur une autre table et récupérer son id pour 10 scores.

3.2. MLD

Score
idScore INT
Player01 VARCHAR(45)
Player02 VARCHAR(45)
ScoreP01 INT
ScoreP02 INT
Indexes

3.3. Scénarios (première version)

Jouer une partie en local

Solo

En tant que joueur,
Je veux jouer tout seul,
Afin de m'amuser

Action	Condition particulière	Résultat
Lancer l'application		Elle s'ouvre
Lancer une partie solo, soit depuis le menu d'en haut, cliquer sur partie local et « partie en solo » ou bien depuis l'accueil cliquer sur « Solo »		Demander le nom du joueur
Introduire le nom	Le joueur clique sur enter ou clique sur le bouton « OK »	la partie se lance, affichage de l'interface de jeu
	Le joueur introduit que des espaces	Demande à l'utilisateur de rentrer son nom
	Le(s) Joueur(s) écrit un prénom plus grand que 11 caractères	
Cliquer sur une case		Une croix est posée, l'ordinateur pose un rond
Après le tour de l'ordi, on clique sur une case	une croix ou rond existant	Rien ne se passe
		Une autre croix est posée
3 croix du joueur sont alignées		Un message de victoire s'affiche
3 ronds de l'ordinateur sont alignés		Un message de défaite s'affiche
Aucun joueur n'arrive à marquer		Un message d'égalité s'affiche
La partie est finie		Le score du gagnant est incrémenté de 1, puis demande au joueur s'il veut recommencer
Le joueur répond à la demande	oui	Une nouvelle partie est relancée en gardant l'ancien score
	non	Sauvegarde la totalité des points dans la base de données

Multijoueur

En tant que joueur,
Je veux jouer tout seul,
Afin de m'amuser

Action	Condition particulière	Résultat
Lancer l'application		Elle s'ouvre
Lancer une partie multijoueur locale, soit depuis le menu d'en haut, cliquer sur Partie Local et « en Multijoueur » ou bien depuis l'accueil cliquer sur « Multijoueur »		Demande le nom des 2 joueurs
Introduire le nom	Les joueurs cliquent sur le bouton « OK »	la partie se lance
	Le(s) joueur(s) introduit que des espaces	Demande à l'utilisateur de rentrer son nom
	Le(s) Joueur(s) écrit un prénom plus grand que 11 caractères	
Joueur 1, Clique sur une case		Une croix est posée
Joueur 2, Clique sur une case		Un rond est posé
Un joueur clique sur une case	une croix ou rond existant	Rien ne se passe
		Une croix / rond est posé
3 croix du joueur sont alignées		Un message de victoire s'affiche avec le nom du joueur
3 ronds de l'ordinateur sont alignés		
Aucun joueur n'arrive à marquer		Un message d'égalité s'affiche
La partie est finie		Le score du gagnant est incrémenté de 1, puis demande au joueur s'il veut recommencer
Le joueur répond à la demande	oui	Une nouvelle partie est relancée en gardant l'ancien score
	non	Sauvegarde la totalité des points dans la base de données

Jouer une partie en réseau

En tant que joueur,

Je veux jouer sur le réseau,

Afin de m'amuser avec quelqu'un

Action	Condition particulière	Résultat
Lancer l'application		Elle s'ouvre
Lancer une partie sur le réseau, soit depuis le menu d'en haut, cliquer sur « Partie sur le réseau » ou bien depuis l'accueil cliquer sur « Partie sur le réseau »		Affiche un message indiquant que la fonctionnalité n'est pas disponible.

Lire les règles

En tant que joueur,

Je veux jouer lire les règles

Afin de connaître le jeu.

Action	Condition particulière	Résultat
Lancer l'application		Elle s'ouvre
Lire les règles, soit depuis le menu d'en haut, cliquer sur « Règles » ou bien depuis l'accueil cliquer sur « Règles »		Affiche une pop-up avec les règles « Les joueurs cliquent à tour de rôle sur une case de la grille et le premier qui parvient à aligner trois de ses symboles horizontalement, verticalement ou en diagonale gagne un point. »

Lire les Informations

En tant que joueur,

Je veux connaître le développeur du jeu

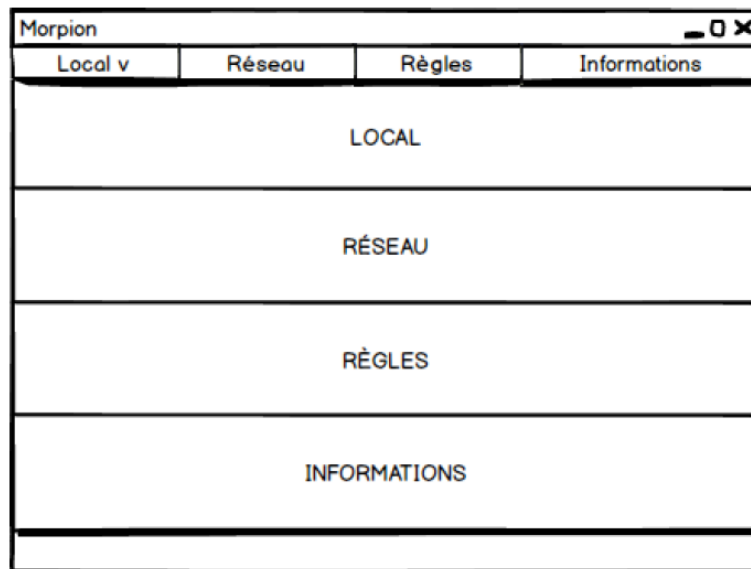
Afin de le remercier.

Action	Condition particulière	Résultat
Lancer l'application		Elle s'ouvre
Lire les informations, soit depuis le menu d'en haut, cliquer sur « Informations » ou bien depuis l'accueil cliquer sur « Informations »		Une pop-up affiche le nom du développeur, la date de création et la version du jeu

3.4. Choix Graphiques (1^{ère} version)

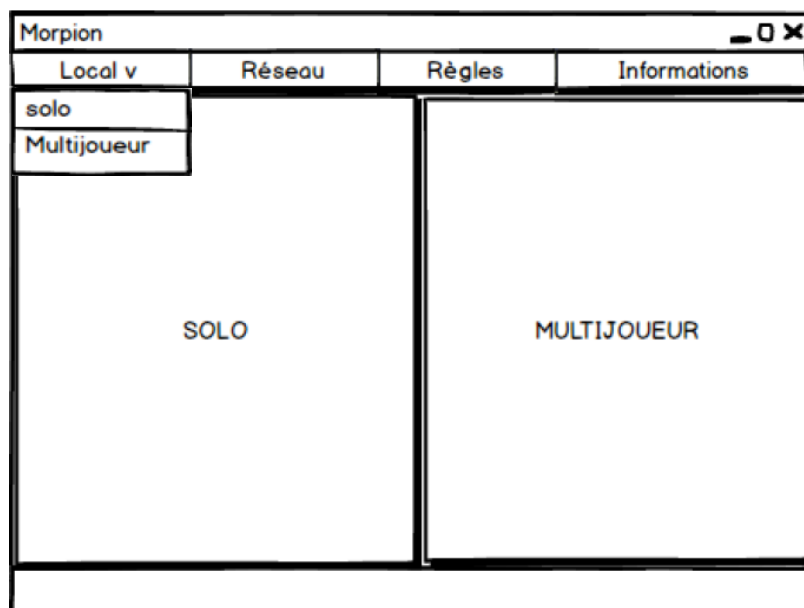
Je me suis basé sur mon wireframe pour faire l'interface finale du jeu, il se peut qu'il y ait une légère différence avec le wireframe et le projet final.

3.4.1. Page principale



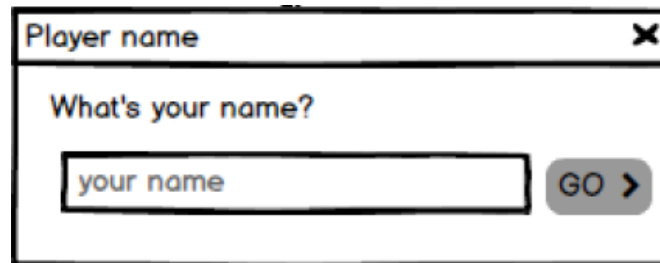
3.4.2. Choix pour une partie locale

Quand l'utilisateur clique sur local du menu en haut il aura le choix entre une partie solo ou multijoueur, c'est-à-dire avec quelqu'un d'autre sur son pc, et il aura les même choix, mais via des boutons s'il passe par le bouton local de l'interface d'accueil



3.4.3. Demande du nom

Après avoir choisi entre une partie solo ou une partie multijoueur, une pop-up demande le nom du joueur.



Player name

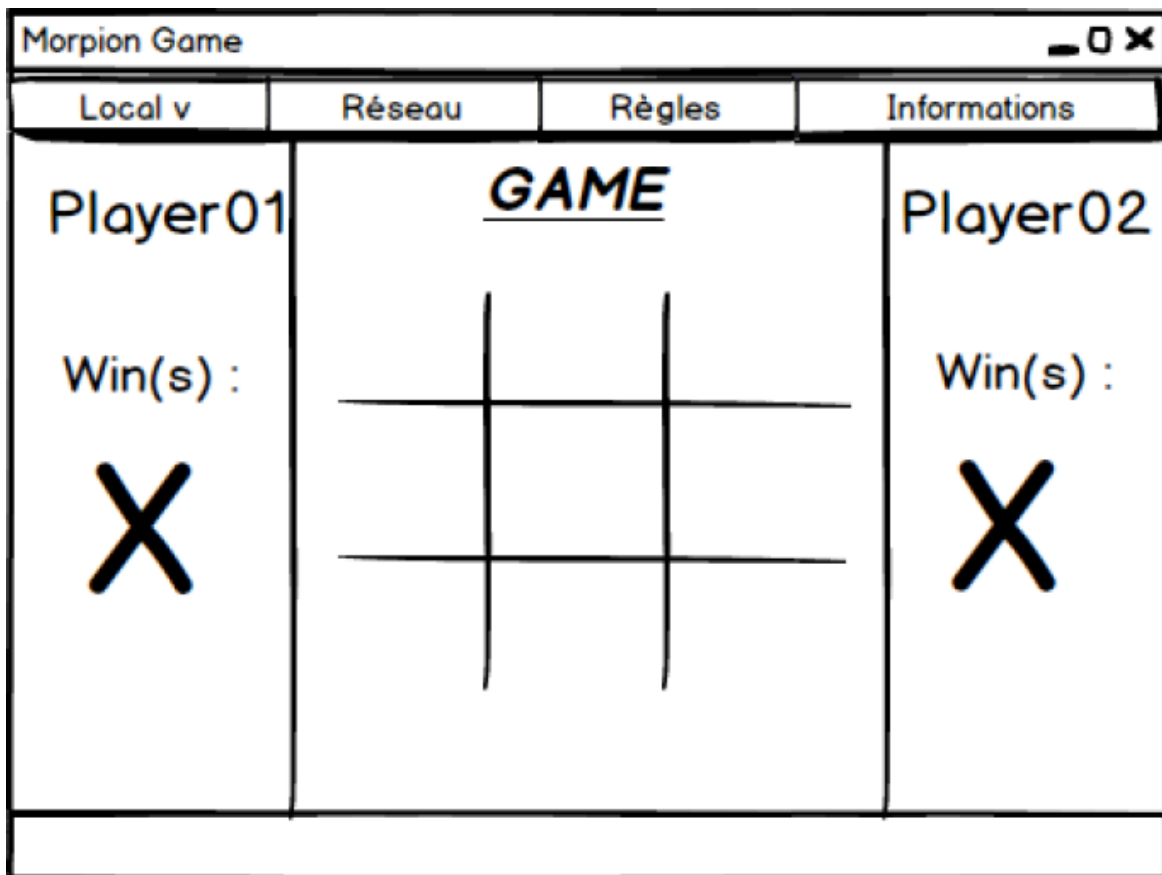
What's your name?

your name

GO >

3.4.4. Partie

Suite à l'insertion de son nom, qui doit être inférieur à 11 pour qu'il ne dépasse pas la séparation, l'interface change pour laisser place au jeu et le(s) joueur(s) peut jouer contre son adversaire jusqu'à la fin de partie



Morpion Game

Local v Réseau Règles Informations

Player01

Win(s) :

X

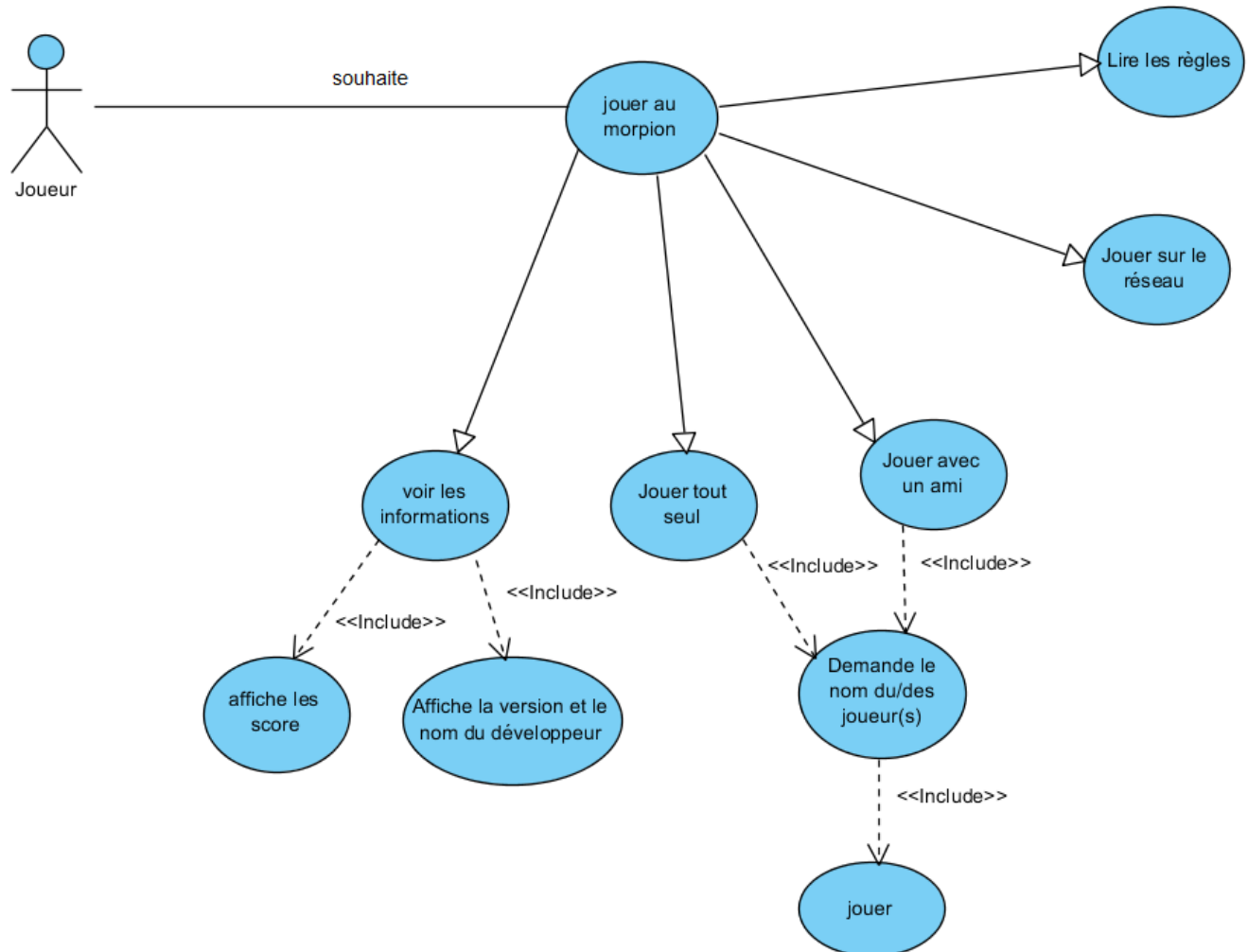
GAME

Player02

Win(s) :

X

3.5. UML



4. Implémentation

4.1. DataBase

Pour la base de données, j'ai opté pour une base de donnée SQLITE au lieu de MySQL.

J'ai choisi SQLITE :

D'abord, le but est de pouvoir jouer avec l'application sans avoir besoin d'un accès quelconque à une base de données extérieur.

Ensuite, pour 10 score je ne trouve pas que le fait de devoir créer une base de donnée sur le réseau soit nécessaire.

La lecture aux données est beaucoup plus rapide.

L'inconvénient est que la base de données occupera un peu de place sur le disque de l'utilisateur.

4.1.1. Création de la base de données

Quand l'utilisateur lancera l'application, le jeu va vérifier si la base de donnée existe et créera celle-ci au besoin.

4.1.2. InsertScore

Quand le joueur ne souhaite plus continuer de jouer, le score est sauvegardé dans la base de données.

J'ai protégé la requête d'insertion avec une commande qui prépare la requête avant de l'exécuter, pour une meilleure représentation, voici le code :

```
SQLiteCommand command = new SQLiteCommand("INSERT INTO Scores(Player01, Player02, ScoreP01, ScoreP02) VALUES (@Player01,@Player02,@ScoreP01,@ScoreP02)", _dbConnection);
command.CommandType = System.Data.CommandType.Text;
command.Parameters.Add("@Player01", System.Data.DbType.String).Value= userName01; //insère la valeur de la variable userName01
command.Parameters.Add("@Player02", System.Data.DbType.String).Value= userName02;
command.Parameters.Add("@ScoreP01", System.Data.DbType.Int32).Value= score01;
command.Parameters.Add("@ScoreP02", System.Data.DbType.Int32).Value= score02;
command.ExecuteNonQuery();
command.Parameters.Clear();
command.Dispose();
```

On insère le nom du joueur 01, puis du 02 et les 2 scores

Chaque fois qu'un score est inséré, le jeu vérifie que la base de données ne possède que la limite de données choisie.

4.1.3. Limite de scores

J'ai décidé de garder dans la base de données un total de 10 scores, si nous souhaitons garder plus il nous suffit de modifier la valeur dans le code. Je pense que dans la prochaine version je vais faire un fichier texte.

4.1.4. ClearScores

Si l'utilisateur désire supprimer tous les scores pour une raison X ou Y, j'ai mis en place un système de suppression de toutes les données.

Je vais l'ajouter à l'interface des informations.

4.1.5. ScoreList

Cette méthode permet d'envoyer à la vue « Informations » une liste contenant les scores des 10 dernières parties Affichés de la manière suivante :

Nom J1	Nom J2	Score J1	Score J2
Beta Testeur	Ordinateur	4	3
noxcaedibux	Ordinateur	0	0
diogo	développeur	0	2
développeur	diogo	0	1
Michael	Ordinateur	1	0
yannick	Ordinateur	3	0
jonathan	Ordinateur	0	1

Le joueur 1 sera toujours le premier nom et le joueur 2 / ordinateur sera toujours celui de droite.

4.2. Test Database

J'ai mis en place des classes de test afin de vérifier que la création, l'insertion, la suppression, etc...

4.2.1. Initialisation

Afin de pouvoir effectuer mes tests, j'initialise ma base de données en lui donnant un nom spécifique, pour notre cas ça sera « Morpion ».

On définit une limite de scores dans la base de données et on indique l'emplacement des fichiers pour la base de donnée en plus de son dossier.

4.2.2. Vérification du dossier / fichier

On regarde que le dossier existe bien et on fait pareil dans un autre test pour le fichier contenant notre base de données

4.2.3. Les insertions

Je vais en premier lieu récupérer toutes les entrées dans la base de données. Ensuite, je compte le nombre d'entrées et lui incrémente de 1 cette valeur afin de définir la valeur attendue après une nouvelle insertion.

Maintenant, j'insère une nouvelle entrée et je compare le nombre d'entrées avec celles calculées précédemment.

4.2.4. 11 Insertions

J'effectue 11 insertions et je test que la base de données garde seulement 10 entrées.

4.3. Controller

Dès que l'application démarre, le contrôleur est appelé en premier, celui-ci appelle la classe « Model » et « View ».

Dans le contrôleur, je crée toutes les parties graphiques (buttons, labels, etc..) et je demande au modèle dès que j'ai besoin d'informations spécifiques telles que les scores, les noms des joueurs, la partie, etc...

4.3.1. TopMenu

Permet de mettre un menu en haut de la vue.

Il possède les boutons / menus suivants :

Le menu « Local » qui permet de choisir entre une partie :

- « Solo », qui demandera le nom du joueur et lance une partie contre l'ordinateur.

- « Multijoueur », qui demandera le nom des joueurs lance une partie locale multijoueur.

- « Partie sur le Réseau », Demande avec qui nous voulons jouer sur le réseau, Actuellement indisponible.

- « Règles », affiche les règles du jeu.

- « Informations », Affiche la version du jeu, les 10 scores de la base de donnée et le nom du développeur sera mentionné.

4.3.2. Main_menu

Interface d'accueil, ayant les mêmes boutons que « topMenu » :

- Solo

- Multijoueur

- Partie sur le réseau

- Règles

- Informations

4.3.3. Game_int

Interface de jeu, cette interface contacte le modèle afin d'avoir le nom des joueurs, même les données de l'ordinateur y sont stockées.

Elle y récupère aussi le score de chaque joueur et initialise le tableau de jeu.

Pour les 9 cases de jeu, j'ai opté pour des Picture Box avec un événement au clic sur chacune d'elles afin de lui mettre le symbole du joueur ayant cliqué sur la case.

Sur la page suivante vous pourrez voir une image du code pour générer les 9 Picture Box aux bons emplacements.


```
for (int i = 0; i < 9; i++)
{
    picCases[i] = new PictureBox();
    picCases[i].Name = i.ToString();
    picCases[i].Size = new Size(90, 90);
    picCases[i].BackColor = Color.LightGray;
    picCases[i].SizeMode = PictureBoxSizeMode.StretchImage;
    picCases[i].Click += UserClick;
    if (i == 0 || i == 3 || i == 6)
    {
        x = 264;
    }

    if (i < 3)
    {
        y = 33;
    }
    else if (i < 6)
    {
        y = 130;
    }
    else
    {
        y = 229;
    }
    picCases[i].Location = new Point(x, y);
    _view.Controls.Add(picCases[i]);
    x += 97;
}
```

Au moment du clic, je retourne toute la Picture Box.

```
private void UserClick(object sender, EventArgs e)
{
    ThinkingGame((PictureBox)sender);
}
```

4.3.4. Show_interface

Cette fonction attend une valeur de type int pour afficher les différentes interfaces de l'application.

Sur toutes les interfaces on affiche le menu du haut.

L'interface 0, celle par défaut, correspond à l'interface d'accueil.

L'interface 1, affiche le jeu avec les noms des joueurs et les scores.

4.3.5. AskUserName

Crée une pop-up et demande le nom d'utilisateur, le bouton de validation envoie sur la méthode CmdOk_Click et un Enter sur les Textbox appelle la méthode MsgBoxAskUserName_KeyDown.

Ces deux méthodes vont appeler la méthode suivante « PopUpUserName »

4.3.6. PopUpUserName

Cette méthode vérifie en premier lieu si nous faisons une partie locale en solo ou en multijoueur, puis vérifie que le nom de chaque joueur fait moins de 11 lettres et qu'il ne soit pas composé de vide / espaces.

Quand toutes les conditions sont remplies, l'interface de jeu est chargée et le ou les joueurs peuvent jouer.

4.3.7. ThinkingGame

ThinkingGame est appelé par la méthode UserClick générée après l'événement de clic des Picture Box.

Cette méthode récupère le nom de la Picture Box, qui correspond à son ID pour ensuite voir si elle n'est pas déjà utilisée, c'est-à-dire qu'un des deux joueurs a déjà son symbole dessus, puis on demande au modèle de nous dire si la partie est terminée.

Quand une partie est terminée on informe qui a gagné, puis une demande de revanche est affichée par le biais de la méthode « Replay ».

4.3.8. Replay

L'utilisateur confirme qu'il veut rejouer une partie, l'interface de jeu est remise à neuf, et on reprend les scores pour savoir lequel des deux joueurs est en train de gagner.

Le joueur ne veut pas refaire une partie, les scores et les noms des joueurs sont sauvegardés dans la base de donnée.

Ensuite on met à 0 les scores et on affiche l'interface d'accueil.

4.4. Model

Le modèle est la classe qui contient toutes les données pour le bon fonctionnement du jeu, de plus celle-ci fait le lien avec la classe « DataBase » dès son initialisation afin de pouvoir interagir avec celle-ci.

4.4.1. CheckGame

Vérifie l'état de la partie afin d'annoncer au joueur si celle-ci est terminée ou si une égalité a été effectuée. Puis change le tour du joueur.

4.4.2. CheckPossibilities

Vérifie pour chaque ligne, colonne et diagonales s'il y a une possibilité pour terminer la partie, si ce n'est pas le cas une valeur de 999 est retournée.

4.4.3. AI (intelligence artificielle en anglais)

Cette méthode permet de générer un ordinateur ayant trois types d'intelligence, facile, moyen, difficile.

L'IA de niveau facile :

Va choisir où jouer de manière aléatoire en vérifiant que la case où elle joue ne soit pas déjà utilisée.

L'IA de niveau moyen :

Place son premier coup de manière aléatoire.

Va essayer d'empêcher le joueur d'aligner 3 de ses symboles.

L'IA de niveau difficile :

Place son premier coup de façon aléatoire.

Va d'abord essayer de voir si elle a moyen de gagner en alignant 3 de ses symboles, puis vérifie si elle peut bloquer le joueur et si aucune de ses conditions n'est possible, elle joue un coup aléatoirement.

4.4.4. SaveGame

Va appeler la classe « DataBase » afin de lui envoyer les données à sauvegarder.

4.5. NetworkCommunication

Cette classe n'est utilisée que dans la branche « dev » sur git, puisque je n'ai pas encore réussi à mettre en place la partie réseau.

Voici comment je procède :

Je récupère l'IP du joueur et je prépare la mise en place du serveur. Quand l'utilisateur clique sur partie en réseau, dans la branche « dev » sur git, j'affiche l'IP du joueur et j'invite l'utilisateur à rentrer l'IP de la personne avec qui il souhaite jouer.

J'ai vérifié avec Telnet si l'accès au serveur était bien effectué, celui-ci est accessible. N'ayant pas le temps de le résoudre, je mets ci-dessous le nom des méthodes de cette classe afin d'avoir un petit résumé de ce que j'essaie de faire

4.5.1. NetworkCommunication

Actuellement cette méthode est en cours de développement, malheureusement je suis bloqué sur celle-ci.

Je récupère l'IP du joueur et je prépare la mise en place du serveur en plus des données pour le client.

4.5.2. SocketSender

Cette méthode envoie un socket, c'est une trame sur le réseau qui contient l'ip source, l'ip de destination, le message que j'ai codé en ASCII et le port utilisé.

4.5.3. SocketReader

Cette méthode ne sert qu'à écouter les trames venant d'un autre joueur, c'est-à-dire qu'il nous envoie un socket avec l'état du jeu après son coup.

4.6. Test Model

Je vérifie que la création de mon tableau de jeu fonctionne et ensuite je vérifie toutes les possibilités possibles pour gagner une partie.

4.6.1. Cas d'une égalité

Pour celui-ci, je crée une exception dans le model dès que les joueurs ne peuvent plus poser de symboles. De ce fait je dois faire un test sur une exception, ce qui m'a pris beaucoup de temps pour créer ce simple test.

J'ai dû ajouter en dessous du `[testMethod]` la ligne suivante : `[ExpectedException(typeof(Exception))]`.

4.7. Dossier de réalisation

Pour que mon projet voit le jour, j'ai eu besoin de :

4.7.1. Outils / Applications

Windows (j'utilise Windows 10 éducation)

Visual Studio 2017 Enterprise

Gantt Project 2.8.1 build 2024

Microsoft Office Professionnel Plus 2016

Git Kraken, c'est un logiciel de versionning comme Git avec une interface graphique

4.7.2. Fichiers / Dossiers

4.7.2.1. CDC (Cahier des charges)

Ce dossier contient le cahier des charges vu avec le chef de projet

4.7.2.2. Code

Contient 2 dossiers :

4.7.2.2.1 DB

Comprend le script pour créer la base de données, j'ai préféré créer en premier lieu la base de données et la tester avant de le faire directement depuis le code morpion.

4.7.2.2.2 Morpion

Morpion est le dossier du projet, celui-ci possède tous les fichiers et dossiers pour générer le projet. On retrouve à l'intérieur de celui-ci :

- Morpion avec toutes les classe et les exécutables
- Packages avec tous les Framework ayant permis la création du jeu
- TestMorpion correspond au dossier contenant toutes mes classes de test
- TestResults contient tous les tests effectués jusqu'à maintenant avec les classes
- FINAL est le dossier contenant la version release du projet qui sera rendue.

4.7.2.3. MCD et MLD

Dossier contenant les fichiers qui m'ont permis de créer mes MCD et MLD en plus d'un format PDF.

4.7.2.4. Doxygen

Il a toute la documentation technique en format HTML de tout mon code.

A la racine, un fichier « Morpion.xml » a été ajouté, il s'agit de la documentation technique générée par Visual Studio.

4.8. Erreur restantes

4.8.1. Choix de symboles

Actuellement mon code ne permet pas de choisir le symbole des utilisateurs en début de partie.

De ce fait, le premier joueur aura toujours une croix et le deuxième un rond, mais cela n'empêche point de jouer au jeu normalement.

Dans une prochaine version, je pourrais ajouter la fonctionnalité.

4.8.2. Partie sur le réseau

Je n'ai pas assez bien géré mon temps, ce qui m'a mis énormément en retard et je n'ai donc pas réussi à mettre en place la partie en réseau.

Ce qui oblige les joueurs à jouer sur le même ordinateur s'ils désirent jouer à plusieurs.

Comme action, j'ai envisagé d'afficher un message annonçant que c'est en cours de développement.

5. Conclusion

5.1. Objectifs

5.1.1. Atteints

J'ai réussi à mettre en place un système de partie solo en local, le tout avec trois types de difficultés :

Facile : L'ordinateur placera aléatoirement chaque coup

Moyen : L'ordinateur a pour objectif d'empêcher le joueur d'aligner ses symboles

Difficile : L'ordinateur essaie d'aligner ses symboles avant d'empêcher le joueur d'aligner les siens

Possibilité de faire une partie locale en multijoueur.

Les joueurs peuvent faire des revanchent à l'infini et quand ils arrêtent le score est sauvegardé.

L'utilisateur peut voir les 10 derniers scores.

5.1.2. Non-atteints

Je n'ai pas réussi à mettre en place une version permettant de jouer sur le réseau, ce qui m'a énormément dessus, car mon objectif premier était de faire un ce jeu pour pouvoir faire des parties sur le réseau.

Les joueurs étaient censés pouvoir choisir le symbole avec lequel ils allaient jouer, mais malheureusement je n'ai pas eu le temps de le mettre en place.

5.2. Points positifs

J'ai beaucoup apprécié faire ce projet en ayant la possibilité d'essayer de mettre en place une structure MVC, ce qui m'a permis de simplifier mon code et de créer que le strict minimum au niveau des fonctions.

J'ai pu voir que l'insertion de données dans les base de données, je ne pensais pas rencontrer autant de difficultés sur ce point.

5.3. Points négatifs

Mauvaise gestion de mon temps, actuellement j'écris cette partie de la documentation sur mon temps pour prévu à l'ajout de la fonctionnalité réseau.

6. Annexes

6.1. Résumé du Rapport

6.1.1. Situation de départ :

Ce projet est parti de deux objectifs, utiliser les threads et les sockets.

Le but de ce projet était de faire en premier lieu un morpion en local, puis mettre en place une fonctionnalité multijoueur sur la même machine et sur une autre version faire en sorte que nous puissions le faire jouer sur le réseau.

En plus de cela, à la fin de chaque partie, le score des joueurs est sauvegardé en local, dans une base de données avec la possibilité de les voir.

6.1.2. Mise en œuvre

Pour la réalisation de ce projet, j'ai utilisé le langage C# .net de Microsoft avec la technologie de WindowsForm sur leur logiciel « Visual Studio 2017 Enterprise » avec la librairie SQLite.

Le tout tournais sur une machine :

Windows 10 Éducation

Intel i7-6700

16 GB de RAM

Pour la création de mes images j'ai utilisé l'outil gratuit Paint.net.

6.1.3. Résultats

Pour conclure, mon application fonctionne en local avec la possibilité de jouer contre des intelligences artificielles et ce avec trois niveaux de difficulté.

Le joueur a aussi la possibilité de jouer avec un collègue sur un même ordinateur.

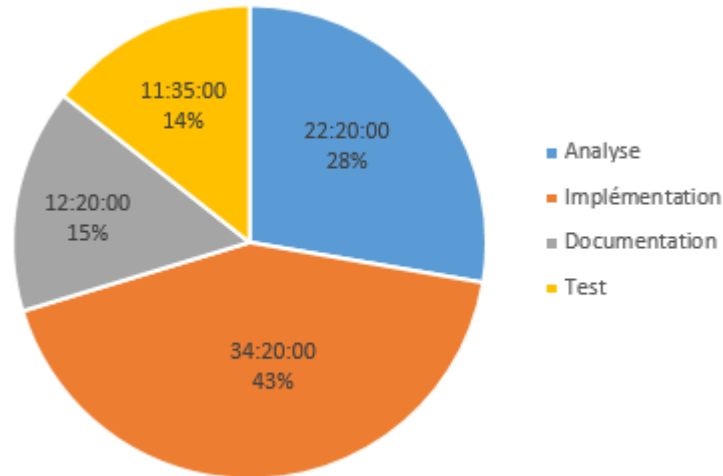
Pour les deux cas, il y a la possibilité de faire des revanches autant de fois que désiré, dès qu'ils refusent une revanche le score est sauvegardé.

Malheureusement mes objectifs liés au réseau n'ont pas été atteints et le fait que le joueur puisse choisir un symbole n'a pas été atteint.

J'ai pris beaucoup de retard sur ma planification la dernière ligne droite, le temps alloué pour apprendre à utiliser les sockets et tester sur le projet a été grandement réduit pour me laisser le temps de faire la doc suite au retard des différentes parties de code et surtout que j'ai malheureusement oublié d'introduire les vacances dans la planification.

6.2. Journal de travail

Durant ce projet, j'ai séparé les différentes tâches pour être le plus possible en accord avec le cahier des charges et voici le résultat final.



Date	Type	Activité	Temps	Temps journée
04.02.2019				
	Analyse	Lecture et compréhension du cahier des charges + contacter le CdP	01:30:00	01:30:00
05.02.2019				
	Documentation	Début de la création de la planif. Initiale	00:45:00	02:15:00
06.02.2019				
	Documentation	continuation de la planification et début de la documentation	01:10:00	03:25:00
	Implémentation	installation / vérification des outils à disposition	00:20:00	03:45:00
07.02.2019				
	Documentation	Finalisation de la première partie de la doc, création du MCD + MLD	02:00:00	05:45:00
08.02.2019				
	Analyse	discussion avec le chef de projet	00:45:00	06:30:00
	Implémentation	mise en place d'un repo GitHub pour la livraison des docs	00:45:00	07:15:00
	Documentation	création des scénarios et maquettes	01:30:00	08:45:00
11.02.2019				
	Implémentation	refonte du mcd+mld	00:15:00	09:00:00
	Analyse	recherches pour le choix entre MySQL ou SQLite, avantages/désavantages et les besoins nécessaires.	01:15:00	10:15:00
12.02.2019				
	Analyse	création du wireframe	00:55:00	11:10:00
	Implémentation	mise en place de l'interface en fonction du wireframe	00:35:00	11:45:00

13.02.2019				
	Implémentation	problèmes avec Visual studio, j'ai dû réinstaller tout le programme	00:45:00	12:30:00
	Analyse	UML	00:45:00	13:15:00
14.02.2019				
	Implémentation	création du code	00:45:00	14:00:00
15.02.2019				
	Test	création des différentes classes, ajout de SQLite au projet, début de la programmation de la base de données. J'ai été bloqué sur les tests unitaires...	04:15:00	18:15:00
18.02.2019				
	Test	test réussis pour la base de donnée, je commence l'affichage du jeu	00:20:00	18:35:00
	Implémentation	mise en place du menu en haut du form et check des fonctionnalités de celui-ci	01:00:00	19:35:00
19.02.2019				
	Analyse	recherches de comment mettre en place le mvc en c#	00:30:00	20:05:00
	Implémentation	mise en place de l'interface graphique en mvc	01:00:00	21:05:00
20.02.2019				
	Documentation	recherches de comment récupérer les valeurs des champs dans un formulaire	00:45:00	21:50:00
	Implémentation	mise en place de la demande du nom d'utilisateur	00:45:00	22:35:00
21.02.2019				
	Implémentation	j'ai essayé de mettre le tout en mvc, début de l'interface de jeu	01:30:00	24:05:00
22.02.2019				
	Implémentation	mise en place du MVC réussie	05:15:00	29:20:00
04.03.2019				
	Implémentation	interface terminée, j'ai réussi à mettre la structure en mvc	01:30:00	30:50:00
05.03.2019				
	Implémentation	mise en place d'évènements sur les picturebox afin de pouvoir mettre une croix dessus et dire que la partie est terminée dès que nous alignons 3 croix	01:30:00	32:20:00
05.03.2019				
	Implémentation	j'ai terminé le fait que le joueur 1 puisse placer une croix et 1 un rond pour le joueur 2.	00:45:00	33:05:00
	Implémentation	possibilité de jouer une partie entre 2 joueurs	00:20:00	33:25:00
	Implémentation	création d'une intelligence artificielle de niveau débutant, elle place des coups aléatoirement	00:15:00	33:40:00
	Implémentation	début de l'intelligence artificielle de niveau moyen	00:10:00	33:50:00
06.03.2019				
	Analyse	j'avais un bug dans ma manière de vérifier les cases, je cherchais un moyen autre moyen de faire mon idée	00:45:00	34:35:00
	Implémentation	implémentation d'une IA plus intelligente	01:30:00	36:05:00

07.03.2019				
	Analyse	Documentation, refonte des scénarios	02:15:00	38:20:00
	Implémentation	modification de l'IA	01:30:00	39:50:00
	Implémentation	finissions des bugs liés à mes 3 IA	01:30:00	41:20:00
08.03.2019				
	Test	mise en place de classes test, création des IA facile, moyenne et difficile.	03:00:00	44:20:00
	Analyse	documentation des différents points	01:30:00	45:50:00
	Analyse	mon IA plante subitement, je recherche via le débbugger le problème	00:30:00	46:20:00
	Implémentation	j'ai corrigé le bug et j'ai mis en place un système qui demande si nous souhaitons une revanche	00:55:00	47:15:00
11.03.2019				
	Implémentation	j'ai mis en place la liaison à la base de données	00:10:00	47:25:00
	Analyse	j'ai des exceptions lancées par ma requête, j'ai fait une recherche sur mon problème	00:50:00	48:15:00
	Implémentation	la requête s'envoie enfin de manière correcte	00:40:00	48:55:00
12.03.2019				
	Implémentation	après avoir ajouté la base de données, de manière inexplicquée mon jeu ne répond plus	01:30:00	50:25:00
13.03.2019				
	Implémentation	correction des bugs, enregistrement des scores + suppression et centrage de toute les fenêtres	01:30:00	51:55:00
14.03.2019				
	Analyse	j'ai refait plusieurs points dans la doc et je refais la partie implémentation	02:15:00	54:10:00
15.03.2019				
	Analyse	j'ai mis en place toute la documentation pour mon projet	05:00:00	59:10:00
	Test	correction des test suite à un changement dans la classe database	00:15:00	59:25:00
17.03.2019				
	Implémentation	j'ai effectué la pop-up message, début de la page informations	01:30:00	60:55:00
18.03.2019				
	Implémentation	j'ai terminé toutes les pages	01:30:00	62:25:00
19.03.2019				
	Implémentation	Mise en place de l'interface Informations, correction du code des bases de données	01:30:00	63:55:00
20.03.2019				
	Documentation	documentation des dernières parties de la première version du code	00:25:00	64:20:00
	Documentation	Modification de la documentation	00:55:00	65:15:00
	Implémentation	génératiion d'une doc doxygen	00:10:00	65:25:00
21.03.2019				

	Documentation	j'ai continué à faire la documentation	01:30:00	66:55:00
22.03.2019				
	Analyse	j'ai discuté avec le CdP et nous avons fait un bref point sur la documentation	00:10:00	67:05:00
	Analyse	j'ai regardé la solution pour communiqué par le réseau fournie par e CdP, mais cela ne correspond pas à mes besoins, car l'utilisateur doit lancer une commande sur le cmd pour pouvoir communiquer avec quelqu'un d'autre, je regarde une autre solution utilisant les sockets	01:20:00	68:25:00
25.03.2019				
	Implémentation	j'ai réussi à faire un mini form qui utilise les thread et les sockets	01:30:00	69:55:00
26.03.2019				
	Implémentation	ajout des fonctions dans la classe network	00:45:00	70:40:00
	Analyse	résumé de rapport	00:45:00	71:25:00
27.03.2019				
	Test	j'ai créé l'interface demandant l'adresse IP de l'adversaire et j'essaie de faire que quand on met l'IP un test de connexion est effectué. Actuellement je bloque sur les exception que cela génère.	01:30:00	72:55:00
28.03.2019				
	Test	tests sur l'implémentation des sockets dans le projet, mais celui-ci me fait des erreur qui passent outre le try and catch	02:15:00	75:10:00
29.03.2019				
	Absence	malade	05:15:00	80:25:00
01.04.2019				
	Implémentation	J'ai essayé de mettre un petit timer avant chaque coup de l'ordinateur, mais malheureusement le timer impacte aussi la pose du joueur..., je l'ai donc enlevé	00:45:00	81:10:00
02.04.2019				
	Analyse	problème dans l'historique de mon journal de travail, correction de celui-ci, j'avais mon journal en différé à cause des différentes branches.	00:20:00	81:30:00
	Documentation	correction de quelques petits points, plus ajout de mes recherches dans la documentation	00:20:00	81:50:00
	Analyse	J'arrive à envoyer mon socket en local, mais j'essaie de comprendre pourquoi mon code plante dès qu'il faut envoyer à un autre pc. Via Telnet nous arrivons à communiquer avec le server	01:00:00	82:50:00
03.04.2019				
	Documentation	Mise en place de l'environnement pour le rendu final	00:15:00	83:05:00
	Documentation	création du power point, plus réflexion des différents points à mettre	01:15:00	84:20:00
04.04.2019				

	Documentation	correction de plusieurs erreurs dans la documentation et ajout d'informations + jdt	01:30:00	85:50:00
--	---------------	---	----------	----------

6.3. Manuel D'installation

6.3.1. Récupérer le projet

6.3.1.1. Avec Git

Il faudra au préalable demander les accès au projet.

Ensuite ouvrir git sur le répertoire désiré et tapez la commande :

```
git clone https://github.com/DiogoVieiraFerreira/Morpion\_Pre-TPI.git
```

Pour naviguer entre les branche, Dev ou master, il vous suffit d'écrire :

```
git checkout branch
```

Au lieu de « branch », insérez le nom de la branche désirée.

6.3.1.2. Sans Git

Téléchargez le dossier compressé, si vous souhaitez télécharger la version livrée :

https://github.com/DiogoVieiraFerreira/Morpion_Pre-TPI/archive/master.zip

par contre si vous souhaitez la version en cours de développement :

https://github.com/DiogoVieiraFerreira/Morpion_Pre-TPI/archive/dev.zip

6.3.2. Logiciel

Il vous faudra Visual Studio 2017 afin d'avoir les mêmes conditions de travail que lors du projet Morpion.

6.4. Manuel D'utilisation

Pour cette partie de la documentation, je l'ai mise à part pour la raison suivante.

Le manuel est destiné à un utilisateur, de ce fait nous n'avons pas besoin de lui fournir tout un rapport s'il ne souhaite qu'avoir un manuel lui expliquant le fonctionnement de l'application.

6.5. Sources / Remerciement

6.5.1. Sources

J'ai utilisé les différents sites pour me documenter : « StackOverflow » et « Openclassroom »

J'ai aussi lu les informations sur activeMQ pour savoir le quel correspondrait le plus à mon utilisation.

<https://www.pmichaels.net/2016/09/29/a-c-programmers-guide-to-installing-running-and-messaging-with-activemq/>

6.5.2. Remerciements

Je remercie Yannick Tercier pour son aide, grâce à lui j'ai pu repérer plusieurs problèmes sur mes IA.

Je remercie Madame Andolfato pour son aide tout au long du projet et Monsieur Glassey pour m'avoir aidé à sur l'arrêt des threads.