Ministère de l'Enseignement Supérieur et de la recherche scienifique
Université de la Manouba
Ecole Nationale des Sciences de l'Informatique

Projet de programmation

Jeu vaches & taureaux

Réalisé par :

BEN AYED Mohamed Nadhmi
REKIK Salah

Encadré par :

Monsieur Mezghich Mohamed Amine

Année universitaire 2014-2015

Remerciements

Au terme de ce travail, nous tenons à exprimer nos sincères remerciements à notre encadrant Monsieur Mezghich Mohamed Amine pour son aide, ses précieuses directives, ses fructueuses explications, ses éclaircissements, sa disponibilité et son encouragement perpétuel pour mener à bien ce travail.

Nous tenons également à exprimer notre reconnaissance à toutes les personnes qui ont contribué de près ou de loin, à la réalisation de ce travail.

Nos remerciements vifs sont aussi à tous nos enseignants à l'ENSI pour leur formation et leur soutien. Nous adressons un remerciement amical et sincère à nos familles et à tous nos collègues et amis qui, par leur courtoisie et leur sympathie, nous ont aidés, encouragé et soutenu.

Table des matières

Introduction

Ce projet s’inscrit dans le cadre du stage de programmation de la première année à l’ENSI. Il s’agit de réaliser un jeu Vache&Taureau.

Le jeu vache&taureau consiste à trouver le nombre auquel pense votre adversaire. D'habitude on joue avec un crayon et du papier, contre un autre joueur. Le premier à découvrir le nombre secret de l'autre gagne .

Chapitre 1 : Règles de gestion

1.1 Règles du jeu

1.Tous les chiffres dans le nombre secret sont différents.

2.Le nombre secret ne peut pas commencer par zéro (dans les règles officielles).

3.Si dans la proposition du joueur il y a des chiffres du nombre secret, aux bons endroits, ils sont des Taureaux.

4.Si dans la proposition du joueur il y a des chiffres du nombre secret, mais pas aux bons endroits, ils sont des Vaches.

5.Le tour d'un joueur consiste à proposer une estimation et déclarer le nombre de vaches et taureaux correspondant à l'estimation de son adversaire .

1.2 Comment utiliser le jeu

1.Au lancement du jeu,l'utilisateur a 4 choix :

- Jouer contre un robot.
- Jouer contre un humain.
- Charger une partie.
- Afficher le tableau des score.
- Consulter l'aide.
- Quitter le jeu.

2.Au cas d'un jeu contre un robot :

- Le joueur commence le jeu en proposant une estimation.
- Le joueur reçoit un déclaration du robot concernant le nombre de vaches et taureaux correspondant à son estimation.
- Le joueur reçoit une estimation de la part du robot.
- Le joueur doit déclarer au robot le nombre de vaches et taureaux correspondant à l'estimation du robot.
- Le joueur propose une autre estimation et l'échange de tour recommence jusqu'à un joueur gagne.

3.Au cas d'un contre un autre joueur :

- Le joueur1 commence le jeu en proposant une estimation.
- Le joueur2 déclare au joueur1 le nombre de vaches et taureaux correspondant à son estimation de ce dernier .
- Le joueur2 propose une estimation au joueur1.

• Le joueur1 doit déclarer au joueur2 le nombre de vaches et taureaux correspondant à l'estimation de ce dernier.

• Le joueur1 propose une autre estimation et l'échange de tour recommence jusqu'à un joueur gagne.

4.Au cas du choix charger une partie :

• Le joueur charger une partie sauvegarder.

5. Au cas du choix score :

• Le joueur consulte la liste des scores.

6.Au cas du choix de l'aide :

• Le joueur peut consulter l'aide du jeu.

7.Au cas du choix de quitter :

• Le joueur quitte la partie.

2.Chapitre 2: Contraintes

2.1 Liste des erreurs possibles

• L'estimation ne doit pas commencer par 0.

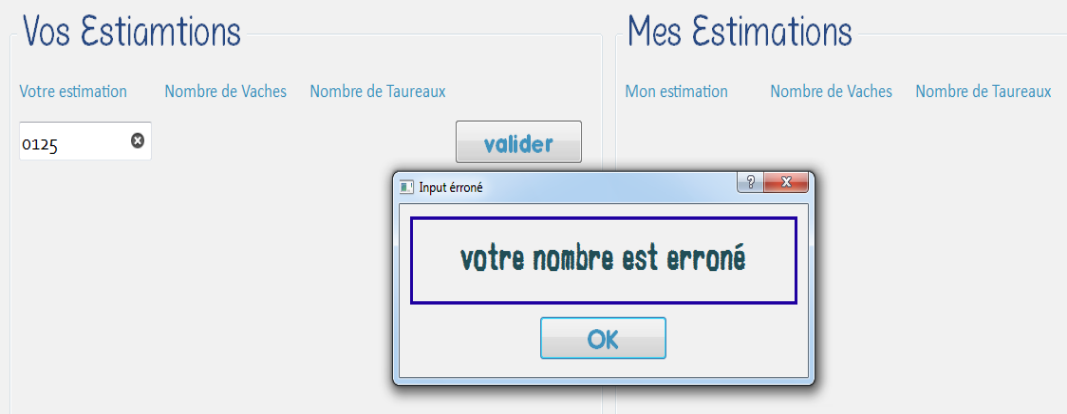

Figure1:Nombre commençant par 0

• L'estimation ne doit pas contenir deux chiffres identiques.

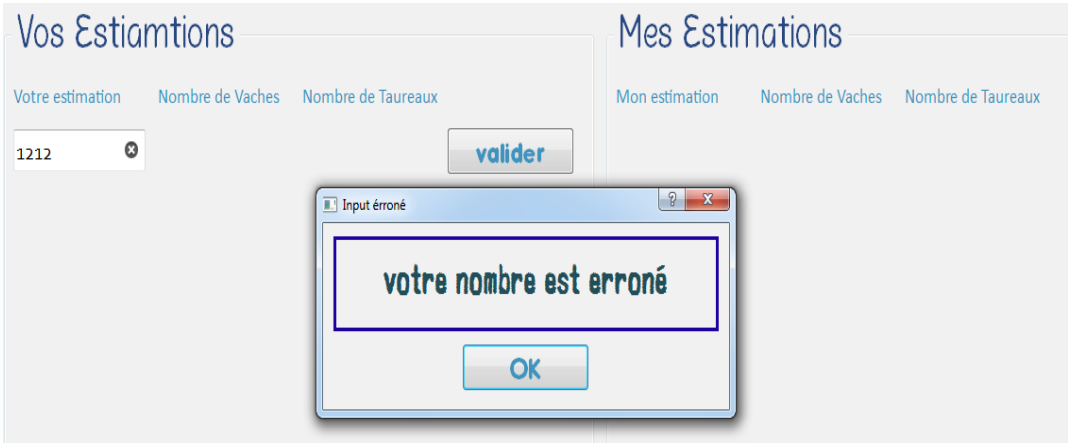

Figure2:Nombre contient deux chiffre identiques

• Le joueur doit entrer une estimation avant de valider.

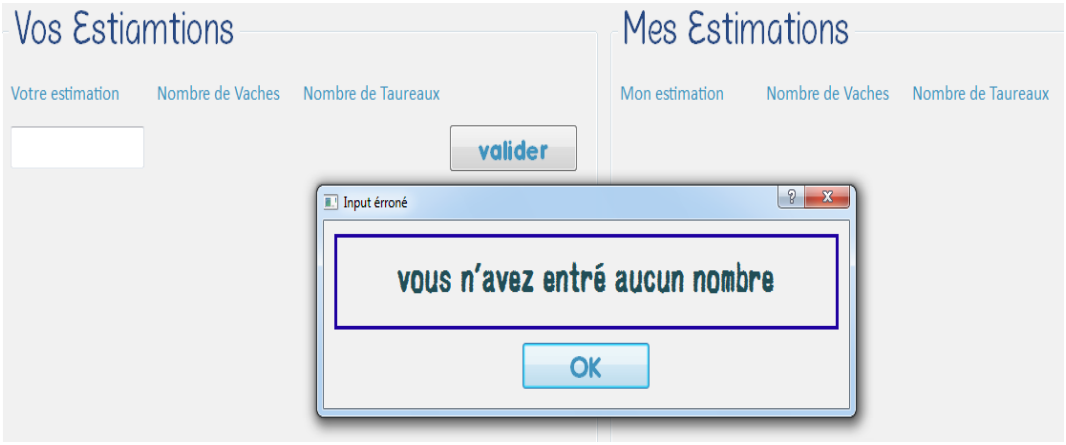

Figure3:Pas d'estimation entrée

• Le joueur ne peut pas charger une partie si elle n'existe pas.

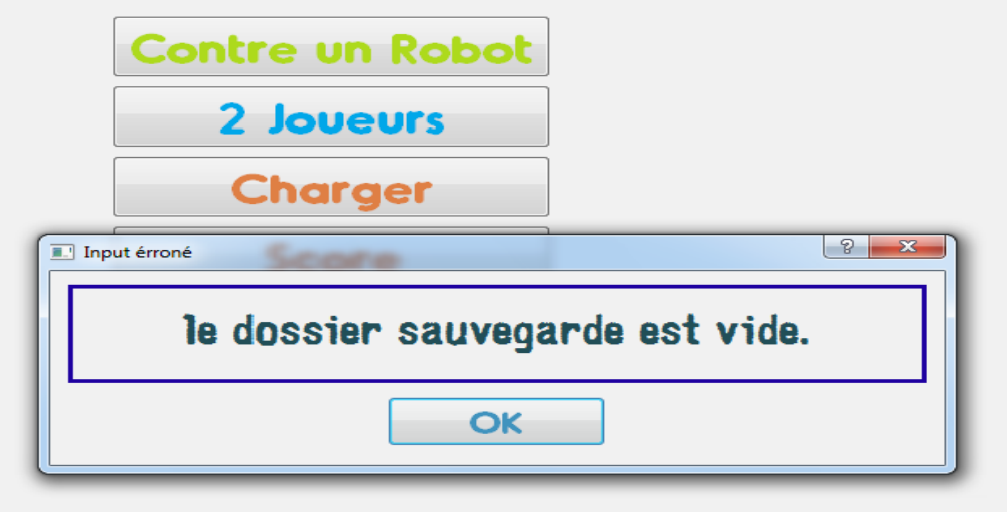

Figure4 :Le dossier sauvegarde vide

• En appuyant sur « pause »,le joueur peut sauvegarder la partie en cours.

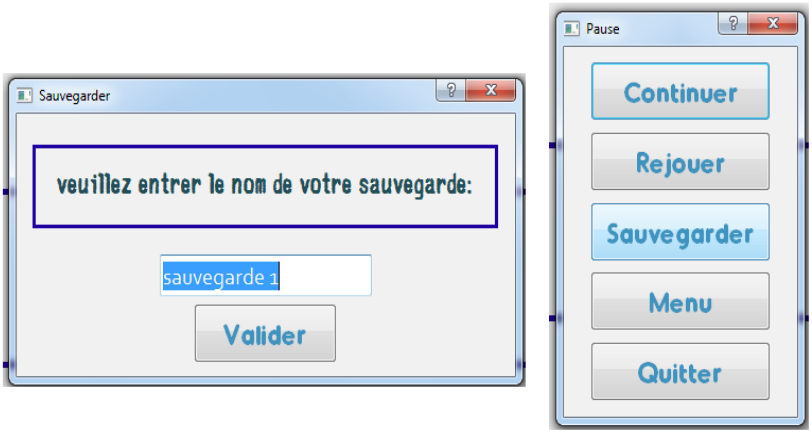

Figure 5:Sauvegarder une partie.

2.2 Etude de cas « Le joueur a menti »

• Si le joueur a menti ou a fait une erreur dans la déclaration des nombres de vaches et taureaux.Une fenêtre indiquant cette action s'affiche.

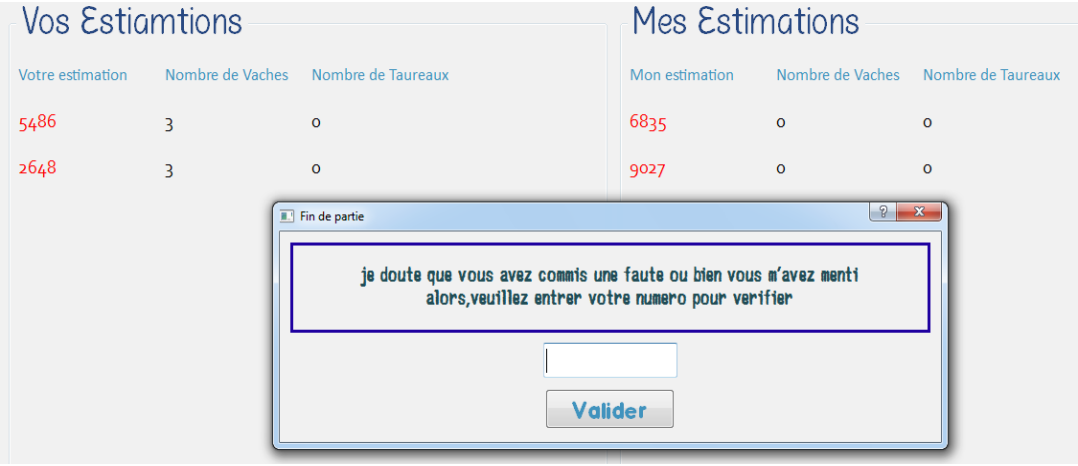

Figure6:Message,le joueur a menti

• En indiquant son vrai nombre choisit, une fenêtre affichant les déclarations incorrectes s'affiche.

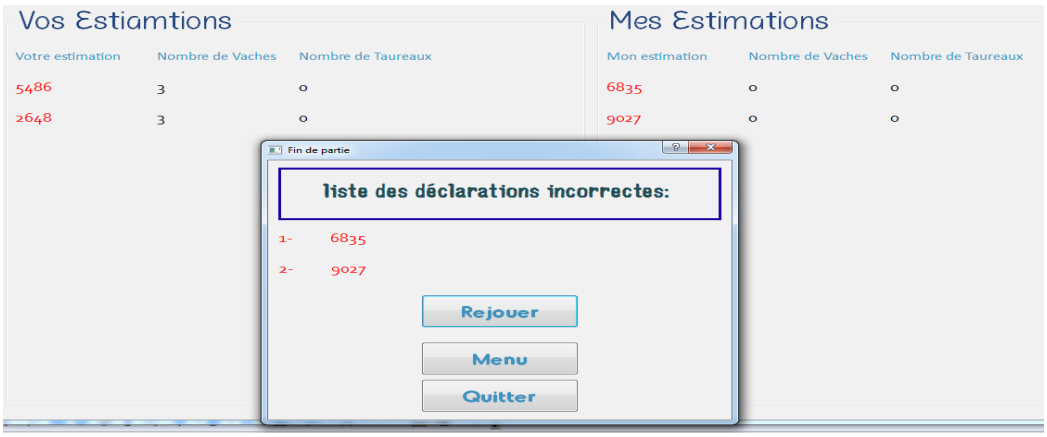

Figure7:Liste des déclarations incorrectes

•Le joueur a le choix entre rejouer,retourner au menu ou bien quitter le jeu après avoir réalisé son erreur.

3.Chapitre 3: Implémentation

L'ensemble des classes :

- classe menu.
- classe calcul_estimation.
- classe sous_fenetre.
- classe tableau_des_estimations.
- classe deux_joueurs.
- classe un_joueur.
- classe aide.
- classe charger
- classe sauvegarder
- classe scroll
- classe score

La classe menu :

Représente l'interface principale du jeu.Elle contient l'ensemble des boutons liées à leurs Slots.

L'ensemble des méthodes dans cette classe :

- menu(QString erreur, int grandeur) : Construire le menu .

- ~*menu*() : Destructeur de classe menu .

- void effacer() : Pour effacer les widgets existants au menu.

N.B : Nous avons éviter de détruire les widgets car Qt détruit automatiquement les widgets liée à leurs parent quand celui-ci est fermé,d'où l'utilisation de la fonction «setVisible ()».

- void sauvegarder(QString file_name) : Appeler les méthodes responsables au sauvegarde d'une partie.

- void charger_une_partie() : Appeler les méthodes responsables au chargement d'une partie.

- void afficher_score() :Afficher les statistiques du jeu.

- void executer() : Afficher l'interface du menu.

- void vs_boot() : Accéder au jeu contre un robot.

- void vs_boot(QString file_name) : Accéder au jeu contre un robot à partir d'un fichier.

- void joueur_vs_joueur() : Accéder au jeu contre un autre joueur.

- void reafficher() : Réafficher le menu avec vérification avec vérification avec un booléen pour connaître la session en cours d'ouverture.

- void rejouer() : Rejouer sans se rendre au menu.

- void rejouer_vs_boot() : Rejouer contre un robot.

- void rejouer_joueur_vs_joueur() : Rejouer en mode deux joueurs.

- void afficher_aide() : Afficher l'aide .

La classe calcul_estimation :

L'ensemble des méthodes dans cette classe :

- calcul_estimation() : Construire tous les groupes de chiffres qui respectent les conditions du jeu(210 groupes en total) .

- calcul_estimation(QString file_name, qint64 pos) : Construire tous les groupes de chiffres qui respectent les conditions du jeu ainsi que les différents changements effectués sur le jeu sauvegardé à partir d'un fichier.

- ~calcul_estimation() : Destructeur de la classe calcul_estimation.

- void sauvegarder(QString file_name) : Utilisée pour sauvegarder d'une partie.

- void construire_liste_des_nombres_du_groupe(int numero) : Construire la liste des nombres dans chaque groupe de chiffre.

par exemple : soit 1234 un groupe de chiffre, la liste des nombre de ce groupe est constituée par tout les nombres qui vérifient les conditions du jeu.

- void copier_un_nombre(char* nombre_recepteur,char* nombre_source) : Copier un nombre.(Nous avons utilisé des char* pas des entiers).

- int calcul_chiffres_communs(char* nombre1,char* nombre2) : Calculer le nombre des chiffres communs entre deux nombres.

- int calcul_vaches(char* nombre1,char* nombre2): Calculer le nombre des « taureaux » entre deux nombres.(Le nombre des chiffres qui sont communs et au même emplacement au nombre 1 qu'au nombre 2) .

- int calcul_taureaux(char* nombre1,char* nombre2) : Calculer le nombre des « vaches » entre deux nombres.(Le nombre des chiffres qui sont communs mais à des emplacements différents au nombre 1 qu'au nombre 2).

- void traitement() : Le traitement principal pour la génération de l'estimation du robot à chaque tour.On calcule la somme des vaches et taureaux entre les deux nombres,puis selon cette somme on réduit le nombre des groupes de chiffre.Puis, on réduit la liste des nombres de chaque groupe restant en testant sur le nombre des taureaux.

- int calcul_effectif() : Calculer le nombre total des nombres qui ne sont pas éliminés.

- void ajouter_estimation(char* nombre,int vache,int taureau) : Ajouter l'estimation courante dans le tableau des estimations(qu'on utilisera pour détecter si le joueur a menti) .

- char* estimation_en_cours() : Retourner l'estimation que le robot va proposer et qui fait partie des nombres existant dans la liste des nombres de chaque groupe de chiffre.

- char** mentir(QString &nombre,int& ligne) : Vérifier si le joueur à menti en testant sur le nombre des taureaux et des vaches.

La classe sous_fenetre :

Nous avons utilisé cette fenêtre pour construire toute fenêtre secondaire (Les messages d'erreur,la fenêtre indiquant l'entrée d'une valeur fausse de vaches ou taureaux …).
Nous avons réservé une seul sous_fenêtre, puis nous modifions son contenu selon le besoin.D'où la déclaration d'une méthode « effacer »
L'ensemble des méthodes dans cette classe :

- Sous_fenetre(menu *Parent) : Constructeur de la classe sous_fenetre.

- ~*Sous_fenetre*(): Destructeur de la classe sous_fenetre.

- void pause() :Afficher la sous_fenetre contenant des boutons qui apparaît avec le clic sur le bouton pause.

- void afficher1(QString ch, int input) :Afficher sous_fenetre pour s'assurer que la proposition du joueur vérifie les conditions.

- void afficher2():Afficher sous_fenetre en relation avec le mensonge du joueur contre le robot.

- void afficher3(QString ch, int input) : Indiquer la fin de la partie.

- void afficher4(QString temps) : Indiquer que le joueur a gagné.

- void effacer() : Effacer le sous_fenetre courant.

- void *closeEvent*(QCloseEvent* event) : Sur-définit la méthode *close()* qui fait partie de la classe QWidget.

- void set_generateur(calcul_estimation* generateur) : Affecter le générateur des estimations.

- void input_valide(QString nombre) : Extraire la liste des déclarations incorrectes données par le joueur depuis le tableau des estimations.

- void verifier_input(QString nombre) : Vérifier si la proposition du joueur vérifie les conditions.

- void set_temps(QTimeLine* Temps) : Affecter le temps.

- void valider() : Fermer la sous_fenetre en validant.

- void Quitter() : Fermer la sous_fenetre.

- void Sauvegarde() : Sert à sauvegarder une partie.

- void reafficher_menu() : Réafficher le menu.

La classe tableau_des_estimations :

L'ensemble des méthodes dans cette classe :

- tableau() : Constructeur de la classe tableau_des_estimations

- tableau(QString file_name, qint64 pos) : Constructeur de la classe tableau_des_estimations à partir d'un fichier.

- ~*tableau()*: Destructeur de la classe tableau_des_estimations.

- char* get_derniere_estimation() : Avoir la dernier nombre estimé par le robot.

- void sauvegarder(QString file_name):Utilisée pour sauvegarder d'une partie.

- void get_derniere_information(int& vache,int& taureau,int& somme_v_t) : Avoir le nombre de vaches,taureaux,et leur somme qui correspondent à la dernière estimation.

- void get_information(int i, char*& estimation, int& vache, int& taureau):Avoir le nombre de vaches,taureaux,et leur somme qui correspondent à l'estimation d'indice i.

- void ajouter_estimation(char* nombre,int vache,int taureau) : Ajouter l'estimation en cours accompagnée des nombre de vaches et taureaux correspondants.

- int get_nombre_des_estimation() : Avoir le nombre des estimation.

La classe deux_ joueurs :

Nous avons utilisé beaucoup de Slots dans cette classe pour éviter la déconnexion puis la reconnexion des boutons à chaque tour.Les Slots : valider_joueur11 et valider_joueur12 correspondent au premier joueur et les deux autres au deuxième joueur.

L'ensemble des méthodes dans cette classe :

- deux_joueurs(menu *Parent, QGridLayout *Layout,Sous_fenetre* fenetre) : Constructeur de la classe deux_joueurs .

- ~*deux_joueurs*() : Destructeur de la classe deux_joueurs .

- void executer() : Accéder au jeu mode 2 joueurs.

- void effacer() : Effacer l'interface du jeu courant.

- void pause():Faire apparaître la sous_fenetre reliée au bouton pause.

- void saisir_estimation_joueur1() : Permettre au premier joueur d'entrer son proposition.

- void saisir_estimation_joueur2() : Permettre au premier joueur d'entrer son proposition.

- void valider_joueur11() :Vérifier le nombre tapé par le joueur 1.

- void valider_joueur21() :Vérifier le nombre tapé par le joueur 2.

- void valider_joueur12() :Régler les combobox pour le joueur 2.

- void valider_joueur22() :Régler les combobox pour le joueur 1.

- void input_valide_joueur1(QString nombre):Donner le tour au joueur 2 pour qu'il indique le nombre de vache et taureaux correspondants à la proposition du joueur 1.

- void input_valide_joueur1(int Vache, int Taureau) : Régler les combobox.

- void input_valide_joueur2(QString nombre) :Donner le tour au joueur 1 pour qu'il indique le nombre de vache et taureaux correspondants à la proposition du joueur 2.

- void input_valide_joueur2 (int Vache, int Taureau) : Régler les combobox.

- void modifier_vache(int n) : Régler le combobox des vaches.

- void modifier_taureau(int n) : Régler le combobox des taureaux.

- bool verifier_nombre(QString nombre) : Vérifier si la proposition du joueur vérifie les conditions.

La classe un_ joueur :

C'est la classe qui assure le jeu contre le robot.

L'ensemble des méthodes dans cette classe :

- un_joueur(menu *Parent, QGridLayout *Layout, Sous_fenetre *fenetre):QObject():Constructeur de la classe un_joueur .

N.B : Nous avons réservé 10 lignes car par un simple calcule de probabilité,le robot à besoin de 8 essaies au maximum pour trouver le nombre de l'adversaire.

- ~*un_joueur*() : Destructeur de la classe un_joueur.

- void executer() : Afficher l'interface du jeu et déclaration du générateur des estimations qui va contenir l'estimation à proposer.

- void executer(QString file_name) :Afficher l'interface du jeu et déclaration du générateur des estimations qui va contenir l'estimation à proposer à partir d'un fichier sauvegardé.

- void effacer() : Effacer le contenu de la fenêtre en cour pour rejouer ou bien revenir au menu.

- void sauvegarder(QString file_name):Sauvegarder une partie pour la charger une autre fois en appelant d'autre méthodes de sauvegarde.

- void changer():Régler et actualiser le QLabel qui affiche le compteur du temps.

- void saisir_estimation_joueur():Donner la main au joueur pour proposer un nombre.

- void saisir_vache_taureau() : Donner la main au joueur pour joueur saisir le nombre de vache et taureau correspondant à la proposition du robot.

- void pause() : Afficher les boutons de la sous_fenetre se déclenchant en cliquant sur le bouton pause.

- void valider1() : Détecter l'erreur relatif à une entrée incorrecte.

- void valider2() : Détecter l'erreur relatif à un déclaration de vache et taureau incorrecte.

- void input_valide1(QString nombre) : Donner la main au joueur pour déclarer le nombre de vaches et taureaux correspondant à la proposition du robot.

- void input_valide2(int Vache,int Taureau) : Le traitement fait par le robot pour indiquer le nombre de vaches et taureaux correspondant à la proposition du joueur,et pour tester s'il y a un gagnant ou égalité.

- int calcul_vaches(char* nombre1,QString nombre2) : Calculer le nombre de vaches entre de nombres.

- int calcul_taureaux(char* nombre1,QString nombre2) : Calculer le nombre de taureaux entre de nombres.

- void modifier_vache(int n) : Régler le combobox des vaches.

- void modifier_taureau(int n) : Régler le combobox des taureaux.

- bool verifier_nombre(QString nombre) : Vérifier le nombre entré.

- void fin_de_partie(int mode) : Actualiser les statistiques des jeux contre un robot.

La classe aide :

Cette classe illustre l'aide .

L'ensemble des méthodes dans cette classe :

- aide(menu *Parent, QGridLayout *Layout) : Constructeur de la classe aide.

- void executer() : Afficher l'aide.

- void effacer() :Effacer les widgets de la fenêtre aide.

- void afficher_menu() : Retourner au menu.

- void retourner_focus():Rendre le bouton « retour » toujours actif.

La classe charger :

Classe pour créer l'interface responsable du chargement d'une partie sauvegardé.

L'ensemble des méthodes dans cette classe :

- charger(menu *Parent, QGridLayout *Layout,Sous_fenetre* fenetre) : Constructeur de la classe charger.

- *~charger()* : Destructeur de la classe charger.

- void executer() :Afficher l'interface du chargement.

- void effacer() : Effacer l'interface du chargement.

- void selecter(int row1, int column1,int row2, int column2) : Vérifier si le joueur a sélectionné une partie sauvegardée ou non.

- void charger_une_partie() : Appeler l'ensemble des méthodes responsables au chargement d'une parties.

- void supprimer() : Supprimer une partie sélectionnée.

- void afficher_menu() :Quitter l'interface du chargement vers le menu.

La classe sauvegarder :

Classe pour créer l'interface responsable du sauvegarde d'une partie dans une sous_fenetre.

L'ensemble des méthodes dans cette classe :

- Sauvegarder(menu *Parent,Sous_fenetre *fenetre, QVBoxLayout *Layout) : Constructeur de la classe sous_fenetre.

- *~Sauvegarder()* : Destructeur de la classe sauvegarder.

- void effacer() : Effacer la sous_fenetre .

- void preexecuter() : Responsable de la modification de la sous_fenetre selon le cas.

- void executer(int compteur): Donner la main au joueur pour entrer le nom de son sauvegarde.

- void valider() : La dernière étape du sauvegarde(modifier la sous_fenetre pour vérifier si un fichier avec le même nom existe déjà,répertoire sauvegarde existe ou non …

- void Oui() : Terminer le sauvegarde.

- void Non() : Annuler le sauvegarde.

La classe scroll :

Cette classe optimise le QscrollArea définit par Qt.

L'ensemble des méthodes dans cette classe :

- Scroll(menu *Parent,QPushButton *Bouton);

- Scroll(menu *Parent,QPushButton *Bouton, QLineEdit *ligne);

- virtual void *scrollContentsBy*(int dx, int dy);

La classe score:

Cette classe crée l'interface responsable du règlement des meilleurs scores et des statistiques du jeu contre un robot.

L'ensemble des méthodes dans cette classe :

- score(menu *Parent, QGridLayout *Layout,Sous_fenetre* fenetre): Constructeur de la classe score.

- ~*score*(): Destructeur de la classe score.

- void executer(): Afficher l'interface montrant les statistiques du jeu.

- void effacer() : Effacer l'interface liée au score.

- void afficher_menu() : Afficher le menu.

4.Chapitre 4: Environnement matériel&logiciel

- Microprocesseur : Intel® core™ 2Duo @ 1,6 GHz
- Mémoire : 2 Go Disque Dur : 160 Go
- Microprocesseur : Intel® core™ i7 @ 3GHz
- Mémoire : 16 Go Disque Dur : 1TA
- Système d'exploitation : Windows 7 Ultimate
- Framework multiplateforme : Qt version 5.5

5.Chapitre 5:Exemples

5.1 : Aperçu du cas « Le robot est gagnant »

Figure8:Cas robot gagnant

5.2 : Aperçu du cas « Le joueur est gagnant »

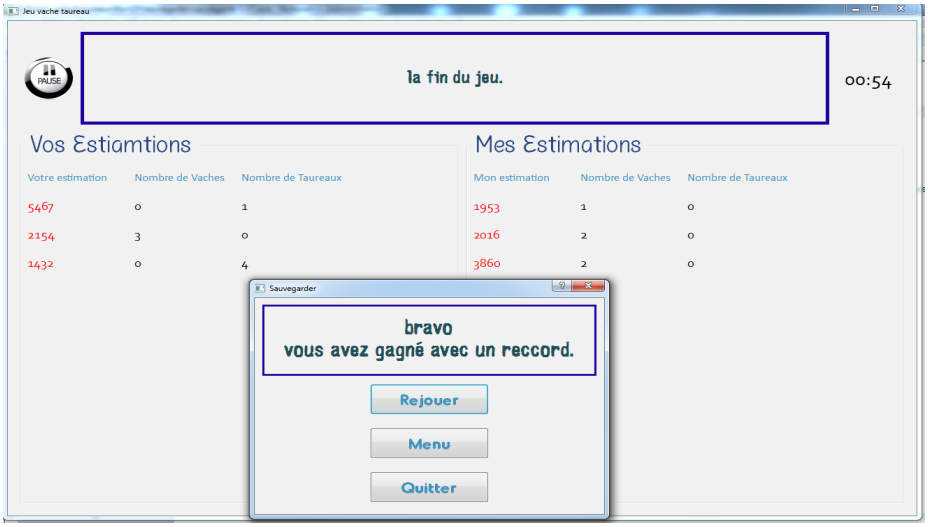

Figure9:Cas joueur gagnant.

5.3 : Aperçu du cas « Le joueur charge une partie »

Figure10:chargement d'une partie.

5.4 : Aperçu du cas « Egalité entre les deux joueurs»

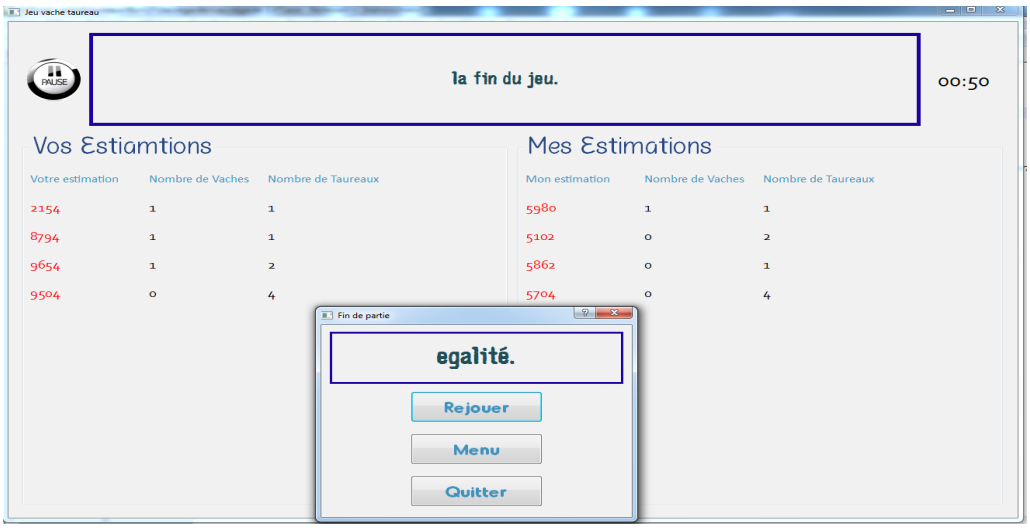

Figure11:Egalité

Conclusion

Vu que les joueurs tendent de nos jours à s'attacher au jeu sur ordinateur , nous avons eu l'idée d'implémenter ce jeu pour attirer l'attention de ces joueurs et leurs permettre de jouer Vache&Taureau sans avoir besoin d'un stylo ou d'une feuille.

Ce travail nous a donné l'occasion de s'initier et de se familiarise à le Framework Qt de C++. Ceci nous a permis de maitriser ce Framework et de déceler ses avantages .Ce travail était pour nous une occasion exceptionnelle pour travailler en groupe .

Bibliographie

[1] Jasmin Blanchette & Mark Summerfield , *C++ GUI Programming with Qt4*,In association with Trolltech Press

Netographie

[N1] https://openclassrooms.com/

[N2] http://doc.qt.io/qt-5/classes.html