

S2.01 – Développement d'une application

Chifoumi – Dossier d'Analyse et conception - Version 8

<https://github.com/samuelhentricks/chifoumi>

Sommaire :

Compléments de spécifications externes.	3
Diagramme des Cas d'Utilisation	3
Scénarios	3
Diagramme de classe (UML)	4
Version v0	8
Implémentation et tests	8
5.1 Implémentation	8
5.2 Test	8
Version v1	9
Classe Chifoumi : Diagramme états-transitions	9
Éléments d'interface	11
Implémentation et tests	12
8.1 Implémentation	12
8.2 Test	13
Version v2	17
Version v3	18
Fichiers .h modifiés	18
Implémentation et tests	18
10.1 Implémentation	18
10.2 Test	19
Version v4	20
Classe Chifoumi : Diagramme états-transitions	20
Nouveaux éléments d'interface	22
Liste des fichiers sources de cette version	22
Fichiers .h modifiés	23
Résultats des tests réalisés	23
Version 5	25
Classe Chifoumi : Diagramme états-transitions	25
Nouveaux éléments d'interface	27
Liste des fichiers sources de cette version	27
Fichiers .h modifiés	28
Résultats des tests réalisés	29
Version 6	34
Classe Chifoumi : Diagramme états-transitions	34
Nouveaux éléments d'interface	36

Liste des fichiers sources de cette version	36
Fichiers .h modifiés	37
Résultats des tests réalisés	38
Version 7	43
Classe Chifoumi : Diagramme états-transitions	43
Nouveaux éléments d'interface	46
Liste des fichiers sources de cette version	46
Fichiers .h modifiés	47
Résultats des tests réalisés	50
Version 8	53
Base de données	53
Liste des fichiers sources de cette version	54
Fichiers .h modifiés	55
Résultats des tests réalisés	56

1. Compléments de spécifications externes.

On précise **uniquement** les points qui vous ont semblé flous ou bien incomplets. Rien de plus à signaler dans cette étude.

2. Diagramme des Cas d'Utilisation

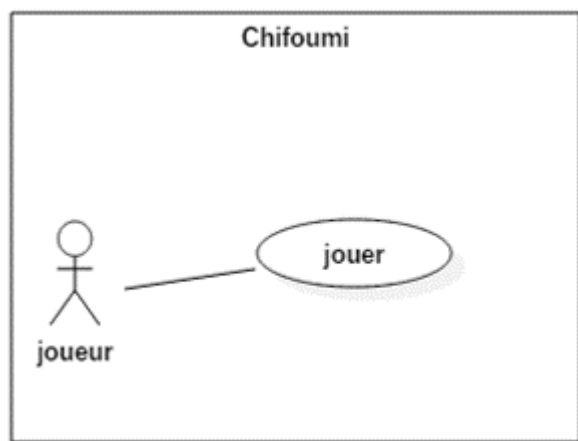


Figure 1 : Diagramme des Cas d'Utilisation du jeu Chifoumi

3. Scénarios

(a)Exemple Scénario

Cas d'utilisation	JOUER	
Résumé	Le joueur joue une partie.	
Acteur primaire	Joueur	
Système	Chifoumi	
Intervenants		
Niveau	Objectif utilisateur	
Préconditions	Le jeu est démarré et se trouve à l'état initial.	
Postconditions		
Date de création		
Date de mise à jour		
Créateur		
Opérations	Joueur	Système
1	Démarre une nouvelle partie.	
2		Rend les figures actives et les affiche actives.
3	Choisit une figure.	
4		Affiche la figure du joueur dans la zone d'affichage du dernier coup joueur.
5		Choisit une figure.
6		Affiche sa figure dans la zone d'affichage de son dernier coup.
7		Détermine le gagnant et met à jour les scores.
8		Affiche les scores. Retour à l'étape 3.
Extension		
3.A	Le joueur demande à jouer une nouvelle partie.	
3.A.1	Choisit une nouvelle partie	
3.A.2		Réinitialise les scores.
3.A.3		Réinitialise les zones d'affichage des derniers coups.
3.A.4		Retour à l'étape 3.

(b) Remarques :

- Le scénario est très simple.
- L'objectif est de mettre en évidence les actions de l'utilisateur, celles du système, sachant que ces actions sont candidates à devenir des méthodes du système

4. Diagramme de classe (UML)

- (a) Le diagramme de classes UML du jeu se focalise sur les classes **métier**, cad celles décrivant le jeu indépendamment des éléments d'interface que comportera le programme.

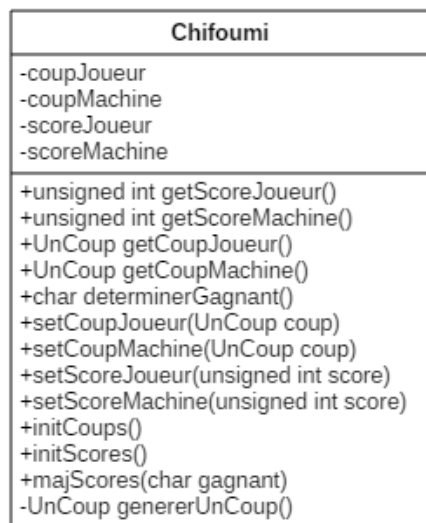


Figure 2 : Diagramme de Classes UML du jeu Chifoumi

(b) Dictionnaire des éléments de la **Classe Chifoumi**

Nom attribut	Signification	Type	Exemple
scoreJoueur	Nbre total de points acquis par le joueur durant la partie courante	unsigned int	1
scoreMachine	Nbre total de points acquis par la machine durant la partie courante	unsigned int	1
coupJoueur	Mémoire la dernière figure choisie par le joueur. Type énuméré enum unCoup {pierre, ciseau, papier, rien};	UnCoup	papier
coupMachine	Mémoire la dernière figure choisie par la machine.	UnCoup	Ciseau

Tableau 2 : Dictionnaire des éléments - Classe Chifoumi

(c) Dictionnaire des méthodes : intégrées dans l'interface de la classe : cf Figure 4

```

using namespace std;
class Chifoumi
{
    /// ---- PARTIE MODÈLE -----
    /// Une définition de type énuméré
public:
    enum UnCoup {pierre, papier, ciseau, rien};

    /// Méthodes publiques du Modèle
public:
    Chifoumi();
    virtual ~Chifoumi();

    // Getters
    UnCoup getCoupJoueur();
    /* retourne le dernier coup joué par le joueur */
    UnCoup getCoupMachine();
    /* retourne le dernier coup joué par le joueur */
    unsigned int getScoreJoueur();
    /* retourne le score du joueur */
    unsigned int getScoreMachine();
    /* retourne le score de la machine */
    char determinerGagnant();
    /* détermine le gagnant 'J' pour joueur, 'M' pour machine, 'N' pour match nul
    en fonction du dernier coup joué par chacun d'eux */

    /// Méthodes utilitaires du Modèle
private :
    UnCoup genererUnCoup();
    /* retourne une valeur aléatoire = pierre, papier ou ciseau.
    Utilisée pour faire jouer la machine */

    // Setters
public:
    void setCoupJoueur(UnCoup p_coup);
    /* initialise l'attribut coupJoueur avec la valeur
    du paramètre p_coup */
    void setCoupMachine(UnCoup p_coup);
    /* initialise l'attribut coupMachine avec la valeur
    du paramètre p_coup */
    void setScoreJoueur(unsigned int p_score);
    /* initialise l'attribut scoreJoueur avec la valeur
    du paramètre p_score */
    void setScoreMachine(unsigned int p_score);
    /* initialise l'attribut scoreMachine avec la valeur
    du paramètre p_score */

    // Autres modificateurs
    void majScores(char p_gagnant);
    /* met à jour le score du joueur ou de la machine ou aucun
    en fonction des règles de gestion du jeu */
    void initScores();
    /* initialise à 0 les attributs scoreJoueur et scoreMachine
    NON indispensable */
    void initCoups();
    /* initialise à rien les attributs coupJoueur et coupMachine
    NON indispensable */

    /// Attributs du Modèle
private:
    unsigned int scoreJoueur; // score actuel du joueur
    unsigned int scoreMachine; // score actuel de la Machine
    UnCoup coupJoueur; // dernier coup joué par le joueur
    UnCoup coupMachine; // dernier coup joué par la machine
};

```

Figure 4 : Schéma de classes = Une seule classe Chifoumi

(d) Remarques concernant le schéma de classes

1. On ne s'intéresse qu'aux attributs et méthodes métier. Notamment, on ne met pas, pour l'instant, ce qui relève de l'affichage car ce sont d'autres objets du programme (widgets) qui se chargeront de l'affichage. Par contre, on n'oublie pas les méthodes `getXXX()`, qui permettront aux objets métier de communiquer leur valeur aux objets graphiques pour que ceux-ci s'affichent.
2. On n'a mis ni le constructeur ni le destructeur, pour alléger le schéma.
3. D'autres attributs et méthodes viendront compléter cette vision ANALYTIQUE du jeu. Il s'agira des attributs et méthodes dits DE CONCEPTION nécessaires au développement de l'application.

Version v0

5. Implémentation et tests

5.1 Implémentation

Liste des fichiers de cette version :

- chifoumi.h : Déclaration de la classe Chifoumi
- chifoumi.cpp : Méthodes de la classe Chifoumi

Respectivement spécification et corps de la classe Chifoumi décrite au paragraphe 4.

5.2 Test

Test avec le programme fournit main.cpp

Testeur(s): Samuel HENTRICS LOISTINE
Ahmed FAKHFAKH
Cédric ETCHEPARE

Date: 14/04/2022

Élément testé : main.cpp

Version: 0

Classe	Description	Valeurs en entrée	Résultat(s) attendu(s)
Valide n°1	Le joueur joue pierre. La machine joue pierre.	coupJoueur=pierre coupMachine=pierre	Aucun gagnant
Valide n°2	Le joueur joue pierre. La machine joue feuille.	coupJoueur=pierre coupMachine=feuille	Machine
Valide n°3	Le joueur joue pierre. La machine joue ciseau.	coupJoueur=pierre coupMachine=ciseau	Joueur
Valide n°4	Le joueur joue feuille. La machine joue pierre.	coupJoueur=feuille coupMachine=pierre	Joueur
Valide n°5	Le joueur joue feuille. La machine joue feuille.	coupJoueur=feuille coupMachine=feuille	Aucun gagnant
Valide n°6	Le joueur joue feuille. La machine joue ciseau.	coupJoueur=feuille coupMachine=ciseau	Machine
Valide n°7	Le joueur joue ciseau. La machine joue pierre.	coupJoueur=ciseau coupMachine=pierre	Machine
Valide n°8	Le joueur joue ciseau. La machine joue feuille.	coupJoueur=ciseau coupMachine=feuille	Joueur
Valide n°9	Le joueur joue ciseau. La machine joue ciseau.	coupJoueur=ciseau coupMachine=ciseau	Aucun gagnant

Version v1

6. Classe Chifoumi : Diagramme états-transitions

(a) Diagramme états-transitions -actions du jeu

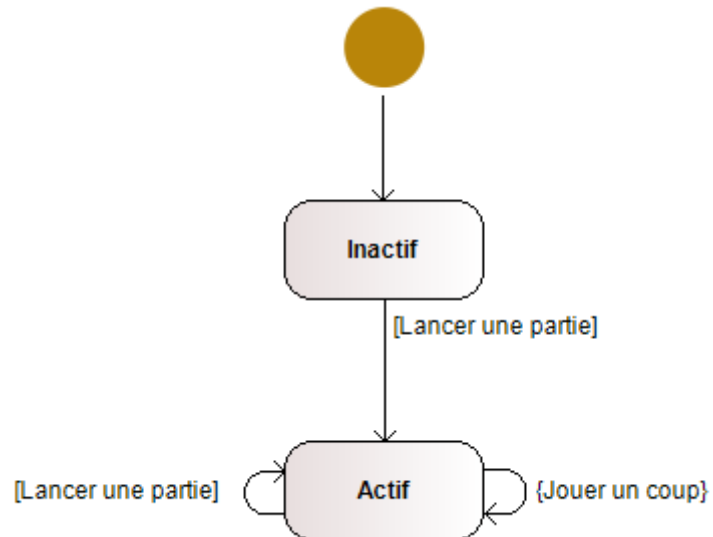


Figure 9 : Diagramme états-transitions

(b) Dictionnaires des états, événements et Actions

Dictionnaire des états du jeu

<i>nomEtat</i>	<i>Signification</i>
Inactif	Le jeu avant qu'on lance une partie
Actif	Le jeu pendant que la partie est en cours

Tableau 2 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

<i>nomÉvénement</i>	<i>Signification</i>
Lancer une partie	Met le jeu en état actif avec les scores des deux joueurs à zéro et leurs coups à "rien".
Jouer un coup	Fait tourner le jeu en déterminant à chaque partie qui est le gagnant en augmentant les scores.

Tableau 3 : Événements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

Inactif -> Actif (Lancer une partie)	Le joueur vient d'arriver sur l'application et souhaite lancer une partie
Actif (Jouer une partie)	Le joueur choisit un coup et joue contre la machine.
Actif (Lancer une partie)	Permet de mettre le jeu à zero et de recommencer une partie

Tableau 4 : Actions à réaliser lors des changements d'état

(c) Préparation au codage :

Table T_EtatsEvenementsJeu correspondant à la version matricielle du diagramme états-transitions du jeu :

- en ligne : les **événements** faisant changer le jeu d'état
- en colonne : les **états** du jeu

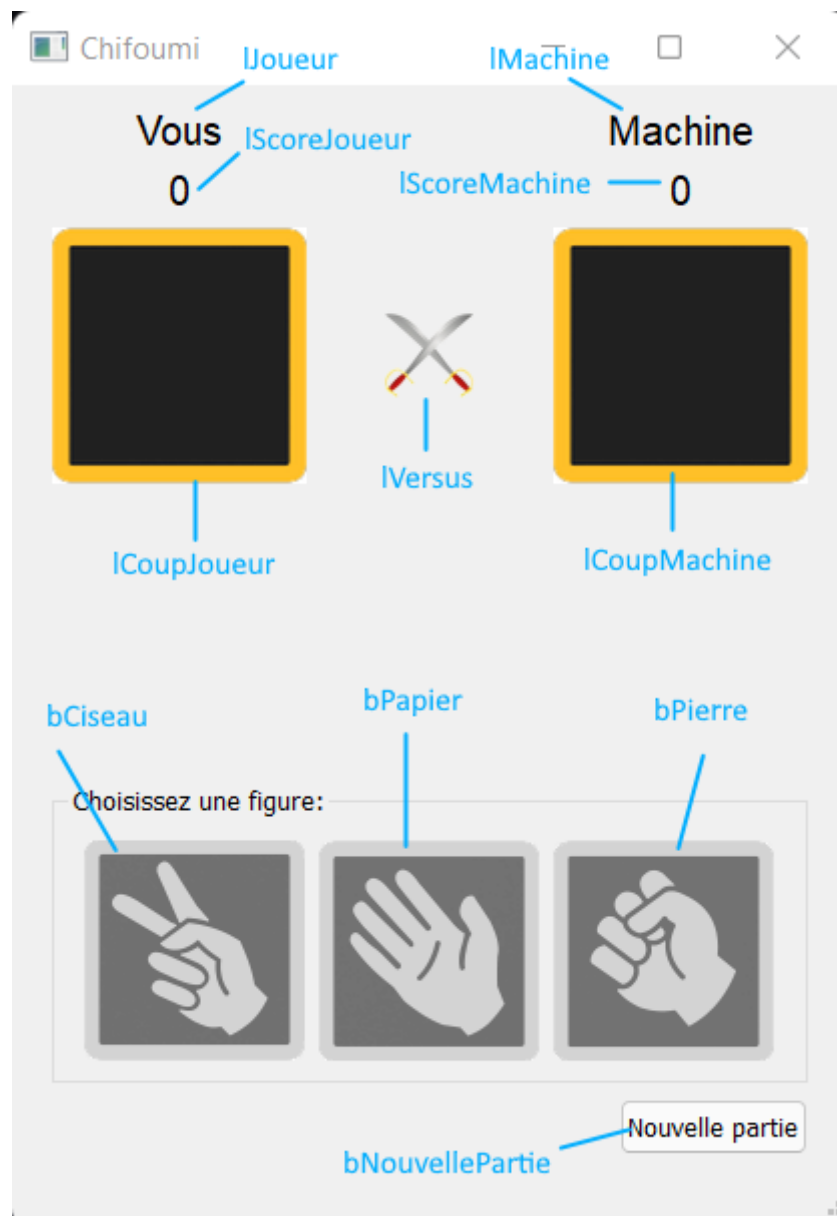
<i>Événement</i> <i>nomEtatJeu</i>	Jouer un coup	Lancer une partie
Inactif		Actif
Actif	Actif	Actif

Tableau 5 : Matrice d'états-transitions du jeu chifoumi

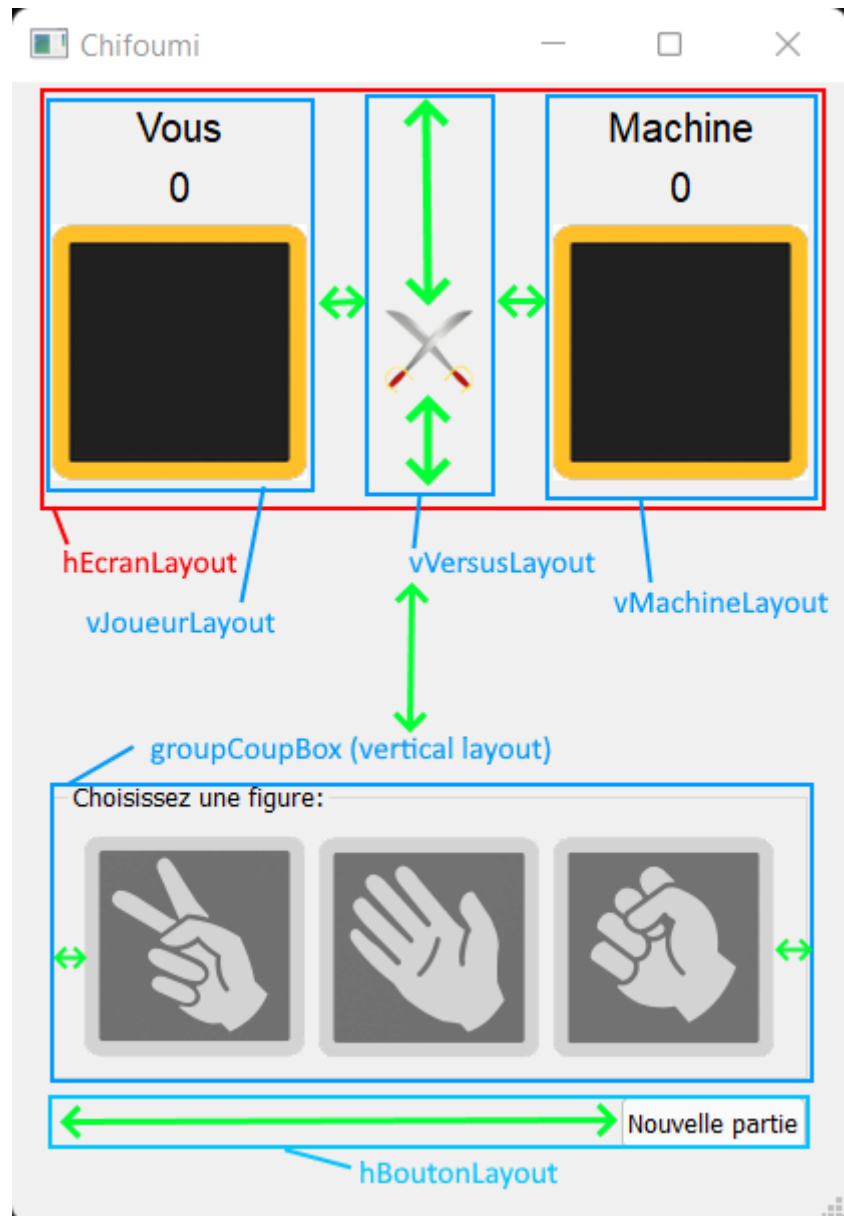
L'intérêt de cette vue matricielle est qu'elle permet une préparation naturelle et aisée de l'étape suivante de programmation.

7. Éléments d'interface

A faire ici : description sommaire des éléments de l'interface, par exemple, avec une copie d'écran sur laquelle sont nommés les variables/objets graphiques et où les layouts sont positionnés et nommés.



Objets graphiques



Les layouts (les "stretch" sont représentés en vert)

8. Implémentation et tests

8.1 Implémentation

A faire :

lister les fichiers impliqués dans cette version (répertoire, nom de fichier, rôle de chaque fichier)

Commenter brièvement les choix importants d'implémentation réalisés, comme par exemple, les signals/slots

chifoumi.pro : Fichier du projet Qt

main.cpp : Fichier permettant l'exécution du programme

chifoumi.cpp : Fichier contenant la définition des fonctions / procédures (disponibles dans le h) nécessaire à la bonne exécution du programme

chifoumi.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la

bonne exécution du programme

chifoumi.ui : Fichier permettant la réalisation et le placement des éléments graphiques du jeu.

chifoumires.qrc : Fichier de ressources Qt

images/ciseau.gif : Permet la sélection du coup "Ciseau" pour que le joueur joue.

images/ciseau_115.png : Permet l'affichage du coup "Ciseau" pour le coup joué de la machine ou du joueur.

images/papier.gif : Permet la sélection du coup "Papier" pour que le joueur joue.

images/papier_115.png : Permet l'affichage du coup "Papier" pour le coup joué de la machine ou du joueur.

images/pierre.gif : Permet la sélection du coup "Pierre" pour que le joueur joue.

images/pierre_115.png : Permet l'affichage du coup "Pierre" pour le coup joué de la machine ou du joueur.

images/rien_115.png : Permet l'affichage d'aucun coup à chaque lancement d'une partie pour la machine ou le joueur.

images/versus.gif : Permet l'affichage entre les deux coups.

8.2 Test

A faire :

Décrire les tests prévus / réalisés pour montrer :

- Le comportement fonctionnel du programme
- Le comportement de l'interface non lié aux aspects fonctionnels du programme

Le joueur joue ciseau
La machine joue ciseau

Egalité

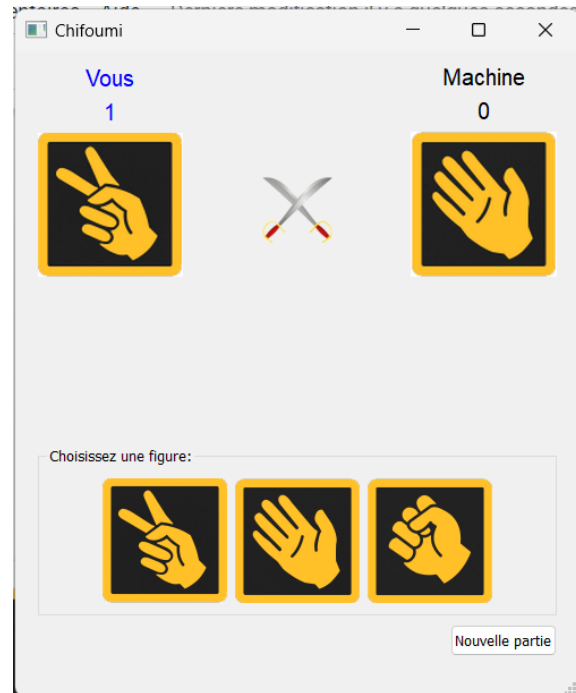
coupJoueur = ciseau
coupMachine = ciseau



Le joueur joue ciseau
La machine joue papier

Victoire du joueur

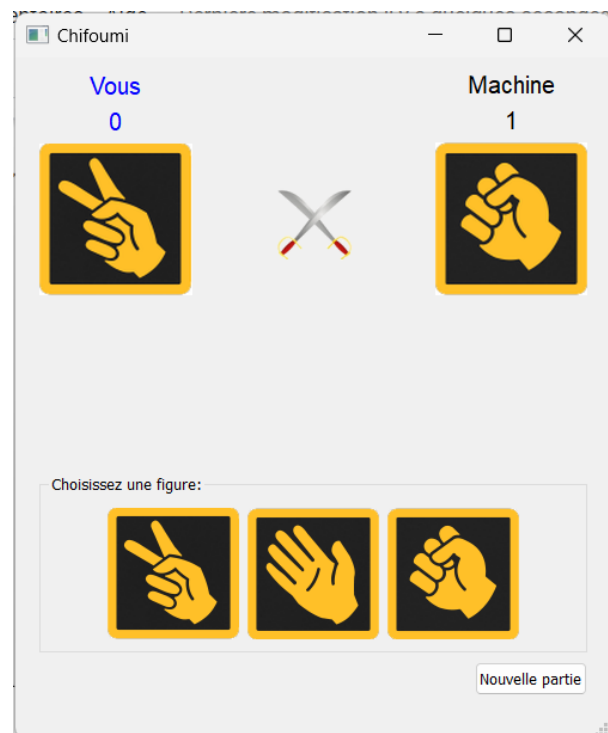
coupJoueur = ciseau
coupMachine = papier



Le joueur joue ciseau
La machine joue pierre

Victoire de la machine

coupJoueur = ciseau
coupMachine = pierre



Le joueur joue papier
La machine joue ciseau

Victoire de la machine

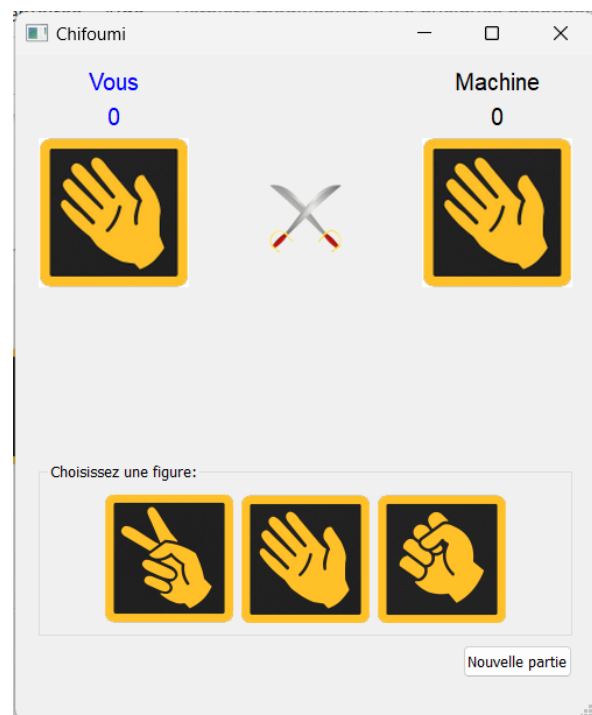
coupJoueur = papier
coupMachine = ciseau





Le joueur joue papier
La machine joue papier

Egalité

coupJoueur = papier
coupMachine = papier



<p>Le joueur joue papier La machine joue pierre</p> <p>Victoire du joueur</p> <p>coupJoueur= papier coupMachine = pierre</p>	
<p>Le joueur joue pierre La machine joue ciseau</p> <p>Victoire du joueur</p> <p>coupJoueur=pierre coupMachine=ciseau</p>	

<p>Le joueur joue pierre La machine joue papier</p> <p>Victoire de la machine</p> <p>coupJoueur = pierre coupMachine = papier</p>	
<p>Le joueur joue Pierre La machine joue Pierre</p> <p>Egalité</p> <p>coupJoueur = pierre coupMachine = pierre</p>	

Version v2

Pas encore implémenté

Version v3

9. Fichiers .h modifiés

```
[...]

public slots:
    void lancerPartie();
        /* Permet de lancer une partie entre le joueur et la machine
        */
    void jouerCiseau();
        /* Le joueur décide de jouer ciseau */
    void jouerPapier();
        /* Le joueur décide de jouer papier */
    void jouerPierre();
        /* Le joueur décide de jouer pierre */
    void jouerPartie(Chifoumi::UnCoup coup);
        /* Permet de déterminer le gagnant et met à jour l'interface
        à partir d'un coup (coup) donné par le joueur */
    void aProposDe();
        /* Permet l'affichage "A propos de..." pour l'utilisateur */

};

#endif // CHIFOUMI_H
```

Extrait du chifoumi.h modifié

Ajout du slot public “aProposDe()” dans chifoumi.h

10. Implémentation et tests

10.1 Implémentation

A faire :

- lister les fichiers impliqués dans cette version (répertoire, nom de fichier, rôle de chaque fichier)
- Commenter brièvement les choix importants d’implémentation réalisés, comme par exemple, les signals/slots

chifoumi.pro : Fichier du projet Qt

main.cpp : Fichier permettant l'exécution du programme

chifoumi.cpp : Fichier contenant la définition des fonctions / procédures (disponibles dans le h) nécessaire à la bonne exécution du programme

chifoumi.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du programme

chifoumi.ui : Fichier permettant la réalisation et le placement des éléments graphiques du jeu.

chifoumires.qrc : Fichier de ressources Qt

images/ciseau.gif : Permet la sélection du coup “Ciseau” pour que le joueur joue.

images/ciseau_115.png : Permet l'affichage du coup “Ciseau” pour le coup joué de la machine ou du joueur.

images/papier.gif : Permet la sélection du coup “Papier” pour que le joueur joue.

images/papier_115.png : Permet l'affichage du coup “Papier” pour le coup joué de la machine ou du joueur.

images/pierre.gif : Permet la sélection du coup “Pierre” pour que le joueur joue.

images/pierre_115.png : Permet l'affichage du coup “Pierre” pour le coup joué de la machine

ou du joueur.

images/rien_115.png : Permet l'affichage d'aucun coup à chaque lancement d'une partie pour la machine ou le joueur.

images/versus.gif : Permet l'affichage entre les deux coups.

10.2 Test

Testeur : Samuel HENTRICS LOISTINE

Date: 04/05/2022

Élément testé : Fermer l'application

Version : 3.0

Classe	Description	État
valide n°1	L'application se ferme depuis "Fichier > Quitter" (voir figure 1)	Validé ▾
valide n°2	L'application se ferme en faisant Alt + F3	Validé ▾
valide n°3	L'application se ferme en cliquant depuis le bouton de fermeture	Validé ▾

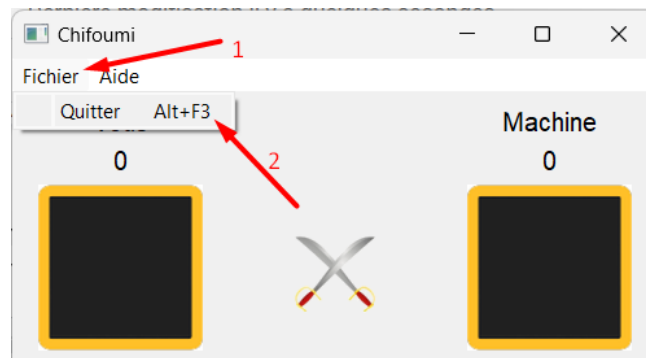


Figure 1. Fermer l'application depuis "Fichier > Quitter"

Testeur : Ahmed FAKHFAKH, Cédric ETCHEPARE

Date: 04/05/2022

Élément testé : Ouvrir la boîte de message (figure 3).

Version : 3.0

Classe	Description	État
valide n°1	La boîte de message s'ouvre depuis "Aide > A propos de..." (voir figure 2)	Validé ▾
valide n°2	La boîte de message s'ouvre en pressant la touche "F1"	Validé ▾



Figure 2. Ouvrir la boîte de message depuis “Aide > A propos de...”

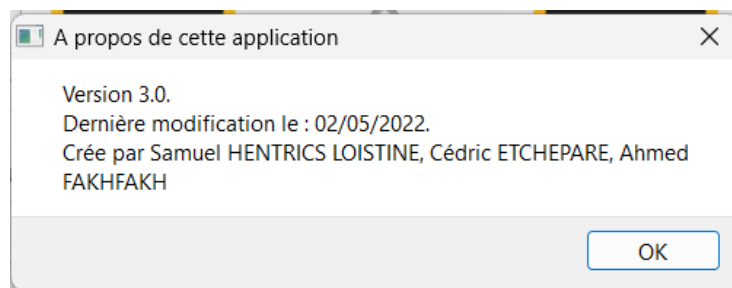


Figure 3. La boîte de messages “A propos de...”

Version v4

11. Classe Chifoumi : Diagramme états-transitions

(d) Diagramme états-transitions -actions du jeu

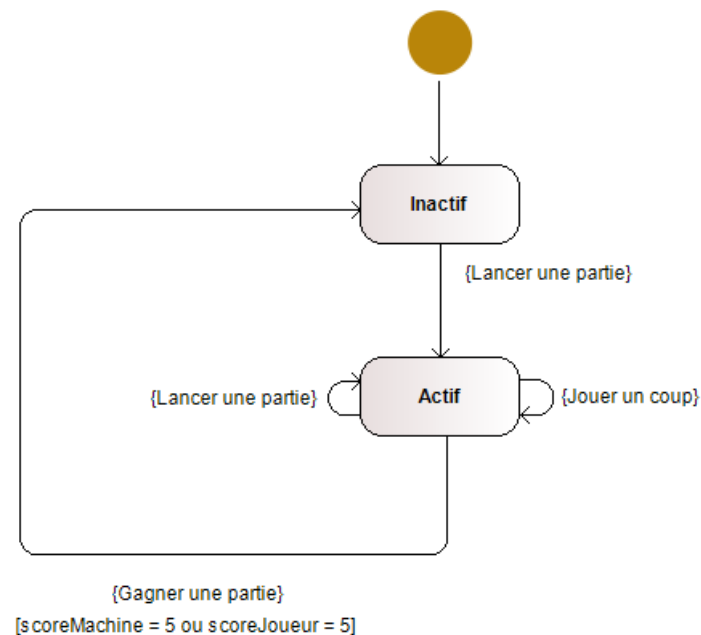


Figure 1 : Diagramme états-transitions

(e) **Dictionnaires des états, événements et Actions**

Dictionnaire des états du jeu

<i>nomEtat</i>	<i>Signification</i>
Inactif	Le jeu avant qu'on lance une partie
Actif	Le jeu pendant que la partie est en cours

Tableau 2 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

<i>nomÉvénement</i>	<i>Signification</i>
Lancer une partie	Met le jeu en état actif avec les scores des deux joueurs à zéro et leurs coups à "rien".
Jouer un coup	Fait tourner le jeu en déterminant à chaque partie qui est le gagnant en augmentant les scores.
Gagner une partie	Met le jeu en état inactif à condition que scoreMachine = 5 ou scoreJoueur = 5

Tableau 3 : Événements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

Inactif -> Actif (Lancer une partie)	Le joueur vient d'arriver sur l'application et souhaite lancer une partie
Actif (Jouer une partie)	Le joueur choisit un coup et joue contre la machine.
Actif (Lancer une partie)	Permet de mettre le jeu à zéro et de recommencer une partie
Actif -> Inactif (Gagner une partie)	La partie est terminée, la machine ou le joueur a atteint 5 points.

Tableau 4 : Actions à réaliser lors des changements d'état

(f) **Préparation au codage :**

Table T_EtatsEvenementsJeu correspondant à la version matricielle du diagramme états-transitions du jeu :

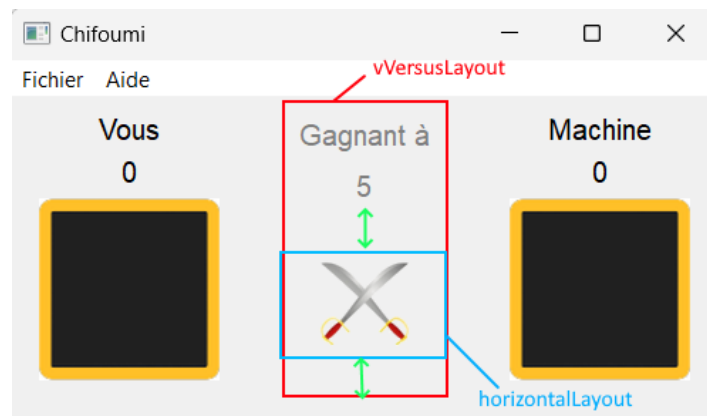
- en ligne : les **événements** faisant changer le jeu d'état
- en colonne : les **états** du jeu

<i>Événement</i>	Jouer un coup	Lancer une partie	Gagner une partie
------------------	---------------	-------------------	-------------------

<i>nomEtatJeu</i>			
Inactif		Actif	
Actif	Actif	Actif	Inactif

Tableau 5 : Matrice d'états-transitions du jeu chifoumi

12. Nouveaux éléments d'interface



Les layouts ajoutés (les "stretch" sont représentés en vert)

13. Liste des fichiers sources de cette version

chifoumi.pro : Fichier du projet Qt

main.cpp : Fichier permettant l'exécution du programme

chifoumi.cpp : Fichier contenant la définition des fonctions / procédures (disponibles dans le h) nécessaire à la bonne exécution du programme

chifoumi.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du programme

chifoumi.ui : Fichier permettant la réalisation et le placement des éléments graphiques du jeu.

chifoumires.qrc : Fichier de ressources Qt

images/ciseau.gif : Permet la sélection du coup "Ciseau" pour que le joueur joue.

images/ciseau_115.png : Permet l'affichage du coup "Ciseau" pour le coup joué de la machine ou du joueur.

images/papier.gif : Permet la sélection du coup "Papier" pour que le joueur joue.
images/papier_115.png : Permet l'affichage du coup "Papier" pour le coup joué de la machine ou du joueur.
images/pierre.gif : Permet la sélection du coup "Pierre" pour que le joueur joue.
images/pierre_115.png : Permet l'affichage du coup "Pierre" pour le coup joué de la machine ou du joueur.
images/rien_115.png : Permet l'affichage d'aucun coup à chaque lancement d'une partie pour la machine ou le joueur.
images/versus.gif : Permet l'affichage entre les deux coups.

14. Fichiers .h modifiés

```
...
public slots:
    void lancerPartie();
        /* Permet de lancer une partie entre le joueur et la machine
        */
    void jouerCiseau();
        /* Le joueur décide de jouer ciseau */
    void jouerPapier();
        /* Le joueur décide de jouer papier */
    void jouerPierre();
        /* Le joueur décide de jouer pierre */
    void jouerPartie(Chifoumi::UnCoup coup);
        /* Permet de déterminer le gagnant et met à jour l'interface
        à partir d'un coup (coup) donné par le joueur */
    void aProposDe();
        /* Permet l'affichage "A propos de..." pour l'utilisateur */
    void finirPartie();
        /* Permet de finir la partie lorsque un joueur a atteint 5 points */
...
```

Extrait du .h modifié

Ajout du slot public "finirPartie()" dans chifoumi.h

15. Résultats des tests réalisés

Testeur : Samuel HENTRICS LOISTINE
 Élément testé : finirPartie()

Date: 12/05/2022
 Version : 4.0

Classe	Description	État
valide n°1	Le programme s'arrête quand le score du joueur atteint 5 points (Figure 1)	Validé ▾
valide n°2	Le programme s'arrête quand le score de la machine atteint 5 points (Figure 2)	Validé ▾

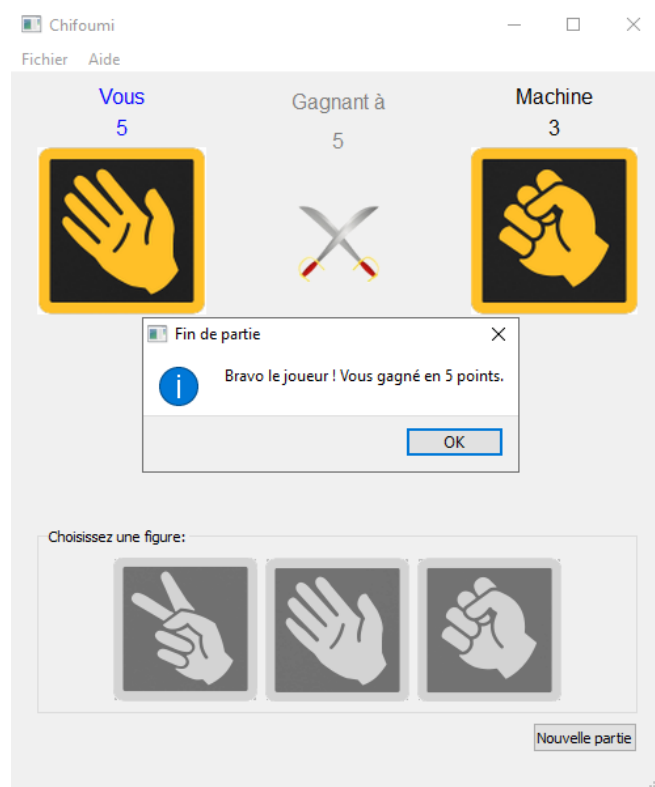


Figure 1. Le joueur a gagné la partie



Figure 2. La machine a gagné la partie

Version 5

16. Classe Chifoumi : Diagramme états-transitions

(g) Diagramme états-transitions -actions du jeu

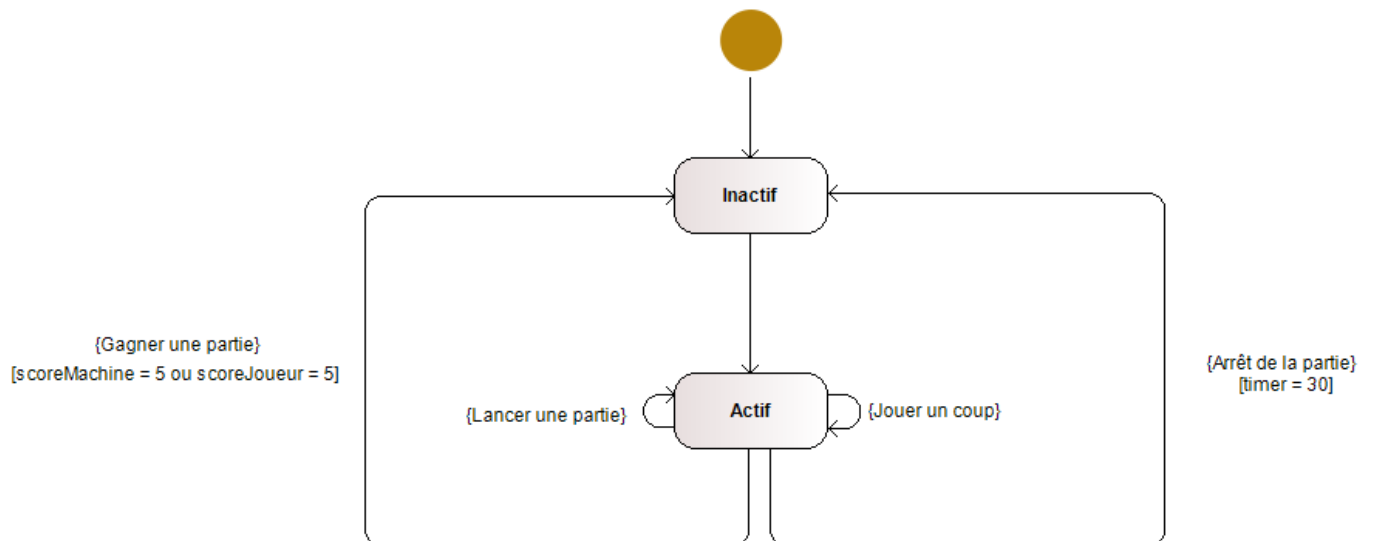


Figure 1 : Diagramme états-transitions

(h) Dictionnaires des états, événements et Actions

Dictionnaire des états du jeu

nomEtat	Signification
Inactif	Le jeu avant qu'on lance une partie
Actif	Le jeu pendant que la partie est en cours

Tableau 2 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

nomÉvénement	Signification
--------------	---------------

Lancer une partie	Met le jeu en état actif avec les scores des deux joueurs à zéro et leurs coups à "rien".
Jouer un coup	Fait tourner le jeu en déterminant à chaque partie qui est le gagnant en augmentant les scores.
Gagner une partie	Met le jeu en état inactif à condition que scoreMachine = 5 ou scoreJoueur = 5
Arrêt de la partie	Met le jeu en état inactif à condition que timer = 30

Tableau 3 : Evénements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

Inactif -> Actif (Lancer une partie)	Le joueur vient d'arriver sur l'application et souhaite lancer une partie
Actif (Jouer une partie)	Le joueur choisit un coup et joue contre la machine.
Actif (Lancer une partie)	Permet de mettre le jeu à zéro et de recommencer une partie
Actif -> Inactif (Gagner une partie)	La partie est terminée, la machine ou le joueur a atteint 5 points.
Actif -> Inactif (Arrêt de la partie)	La partie est terminée, le temps est arrivé à 30s

Tableau 4 : Actions à réaliser lors des changements d'état

(i) Préparation au codage :

Table T_EtatsEvenementsJeu correspondant à la version matricielle du diagramme états-transitions du jeu :

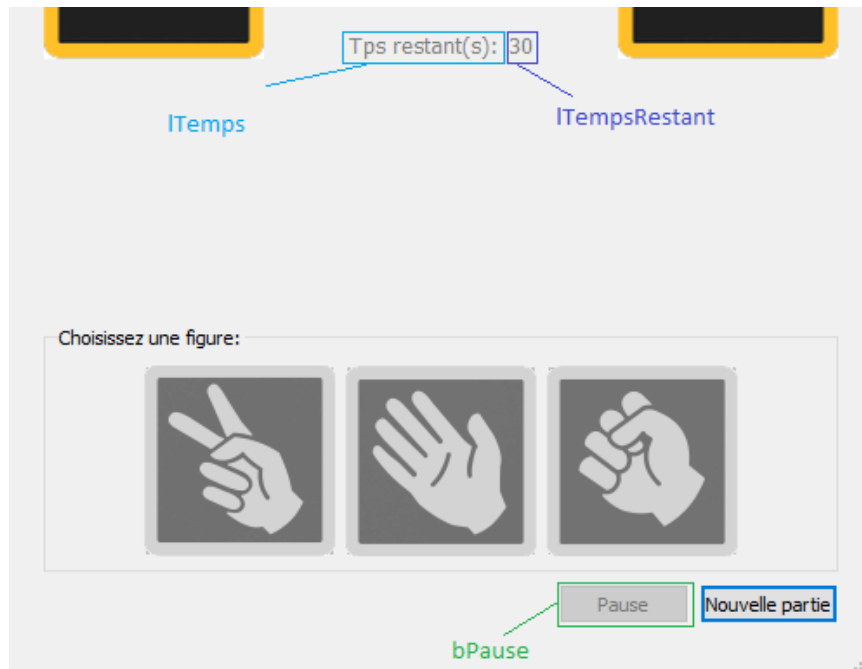
- en ligne : les **événements** faisant changer le jeu d'état

- en colonne : les **états** du jeu

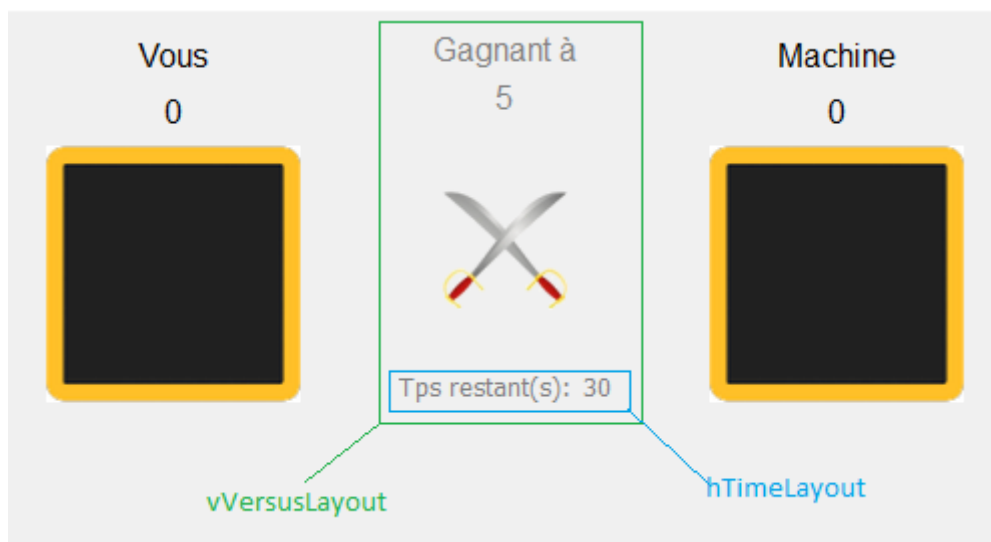
Événement nomEtatJeu	Jouer un coup	Lancer une partie	Gagner une partie	Arrêt d'une partie
Inactif		Actif		
Actif	Actif	Actif	Inactif	Inactif

Tableau 5 : Matrice d'états-transitions du jeu chifoumi

17. Nouveaux éléments d'interface



Objets graphiques



Les layouts ajoutés (les "stretch" sont représentés en vert)

18. Liste des fichiers sources de cette version

chifoumi.pro : Fichier du projet Qt

main.cpp : Fichier permettant l'exécution du programme

chifoumi.cpp : Fichier contenant la définition des fonctions / procédures (disponibles dans le h) nécessaire à la bonne exécution du programme

chifoumi.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du programme

chifoumi.ui : Fichier permettant la réalisation et le placement des éléments graphiques du jeu.

chifoumires.qrc : Fichier de ressources Qt

images/ciseau.gif : Permet la sélection du coup "Ciseau" pour que le joueur joue.

images/ciseau_115.png : Permet l'affichage du coup "Ciseau" pour le coup joué de la machine ou du joueur.

images/papier.gif : Permet la sélection du coup "Papier" pour que le joueur joue.

images/papier_115.png : Permet l'affichage du coup "Papier" pour le coup joué de la machine ou du joueur.

images/pierre.gif : Permet la sélection du coup "Pierre" pour que le joueur joue.

images/pierre_115.png : Permet l'affichage du coup "Pierre" pour le coup joué de la machine ou du joueur.

images/rien_115.png : Permet l'affichage d'aucun coup à chaque lancement d'une partie pour la machine ou le joueur.

images/versus.gif : Permet l'affichage entre les deux coups.

19. Fichiers .h modifiés

```
...

private:
    void initScores();
        /* initialise à 0 les attributs scoreJoueur et scoreMachine
        NON indispensable */
    void initCoups();
        /* initialise à rien les attributs coupJoueur et coupMachine
        NON indispensable */
    void desactiver();
        /* permet de désactiver les boutons, le timer
        */
...

private slots:
    void lancerPartie();
        /* Permet de lancer une partie entre le joueur et la machine
        */
    void jouerCiseau();
        /* Le joueur décide de jouer ciseau */
    void jouerPapier();
        /* Le joueur décide de jouer papier */
    void jouerPierre();
        /* Le joueur décide de jouer pierre */
    void jouerPartie(Chifoumi::UnCoup coup);
        /* Permet de déterminer le gagnant et met à jour l'interface
        à partir d'un coup (coup) donné par le joueur */
    void aProposDe();
        /* Permet l'affichage "A propos de..." pour l'utilisateur */
    void finirPartie();
        /* Permet de finir la partie lorsque un joueur a atteint 5 points */
    void majTemps();
        /* Met à jour le temps restant lors d'une partie et peut arreter la partie
        si le compteur est à zero */
    void majPause();
        /* Met le jeu en pause lorsque l'utilisateur demande à mettre le jeu en pause.
        * Ou alors reprend la partie si le timer est inactif */
...
```

Extrait du .h modifié

Modification de l'accès pour certaines procédures/fonctions en private.

Ajout de la procédure : majTemps(), majPause() et desactiver().

Modification des procédures : aProposDe(), finirPartie().

majTemps() : permet de mettre à jour l'interface toutes les secondes et détermine si le timer est arrivé à zero

majPause() : permet de mettre en pause ou de reprendre le jeu.

desactiver() : permet de désactiver les boutons et le timer.

aProposDe() : mise à jour vers la version V5 (message et date).

finirPartie() : ajout de l'option où le timer est arrivé à zero.

20. Résultats des tests réalisés

Testeur : Samuel HENTRICS LOISTINE

Date: 24/05/2022

Élément testé : finirPartie()

Version : 5.0

Classe	Description	État
valide n°1	Le programme s'arrête car il n'y a plus de temps restant et le joueur est désigné comme gagnant par défaut(Figure 1)	Validé ▾
valide n°2	Le programme s'arrête car il n'y a plus de temps restant et la machine est désignée comme gagnante par défaut (Figure 2)	Validé ▾
valide n°3	Le programme s'arrête car il n'y a plus de temps restant et le joueur et la machine ont le même score (Figure 3)	Validé ▾

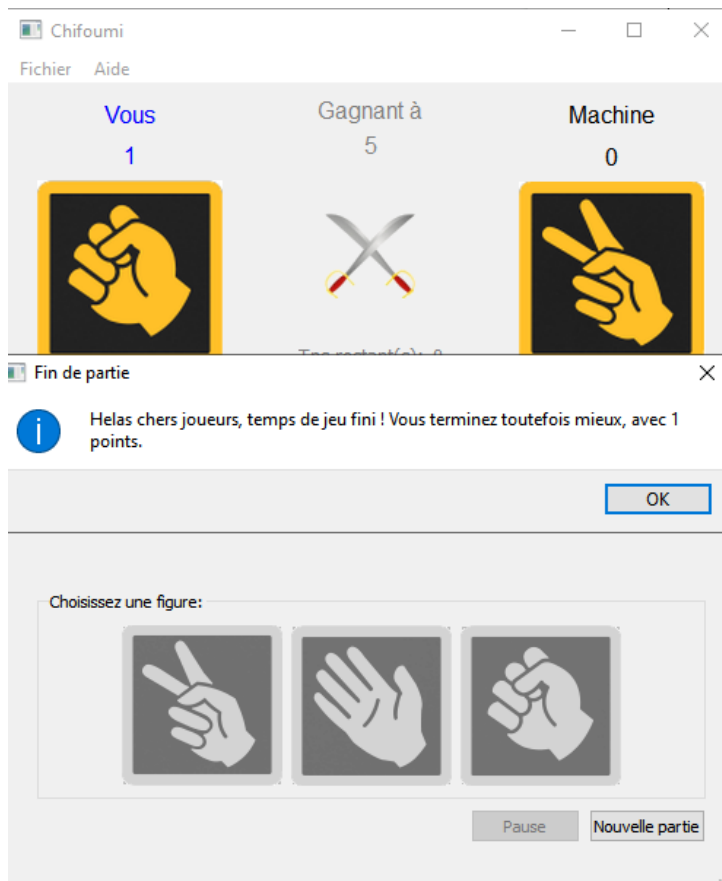


Figure 1. Temps fini et avantage pour le joueur

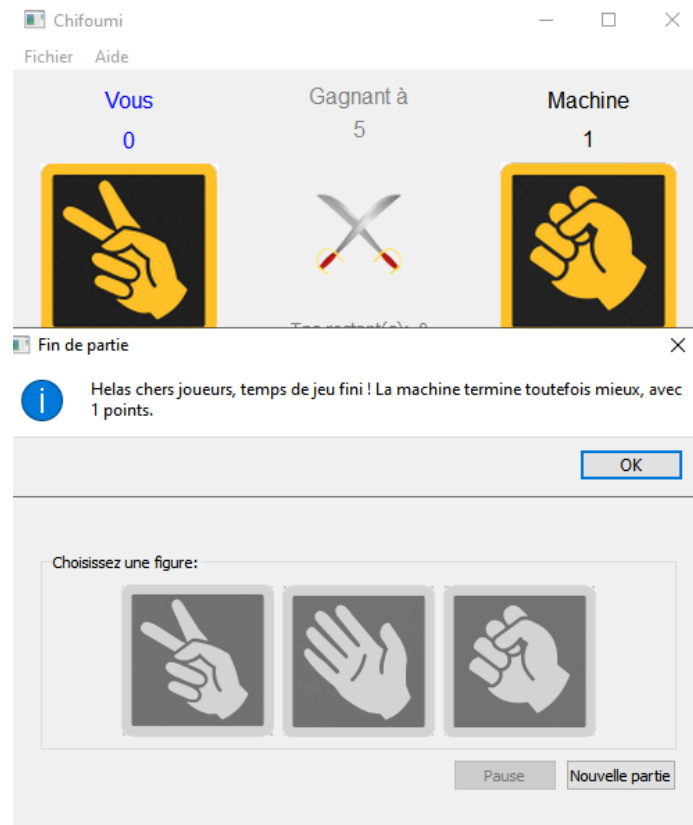


Figure 2. Temps fini et avantage pour la machine

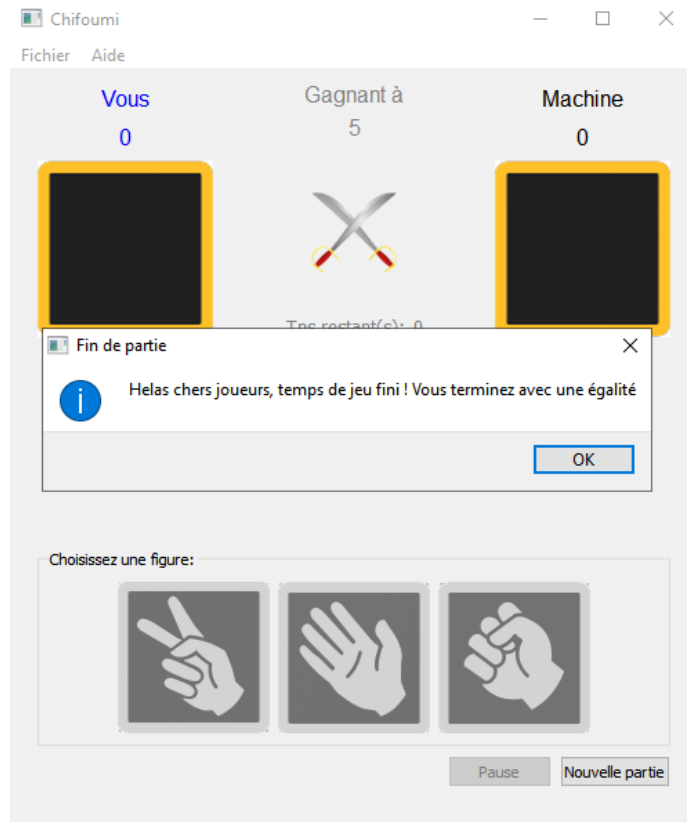


Figure 3. Temps fini et score égal

Testeur : Cédric ETCHEPARE
 Élément testé : majPause()

Date: 24/05/2022
 Version : 5.0

Classe	Description	État
valide n°1	Le joueur décide de mettre en pause le jeu (Figure 4).	Validé ▾
valide n°2	Le joueur décide de reprendre le jeu (Figure 5).	Validé ▾



Figure 4. Mise en pause du jeu

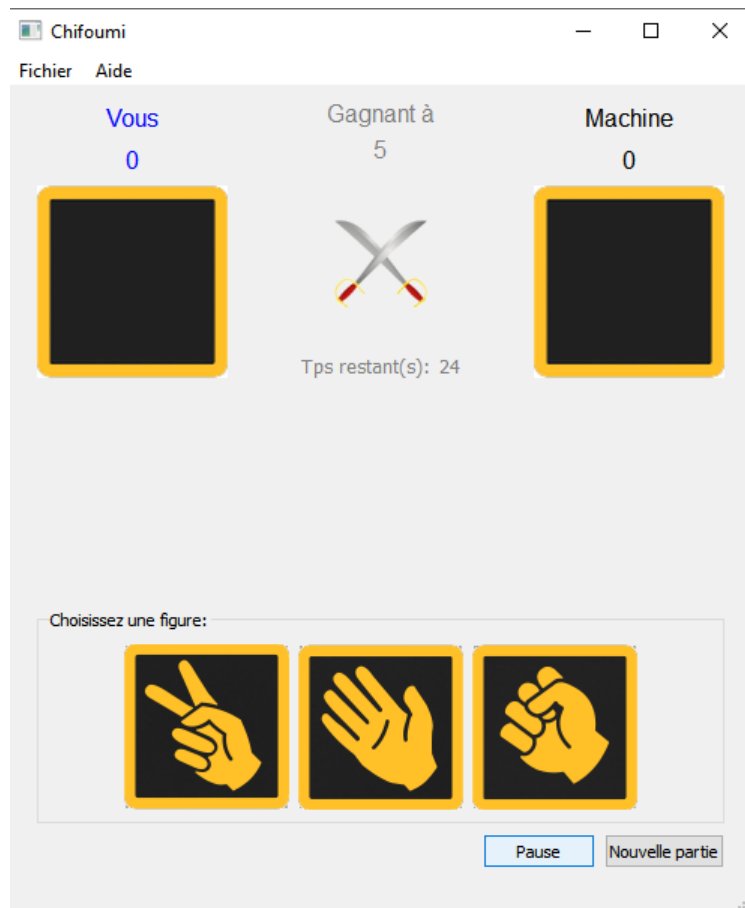


Figure 5. Reprise du jeu

Version 6

21. Classe Chifoumi : Diagramme états-transitions

(j) Diagramme états-transitions -actions du jeu

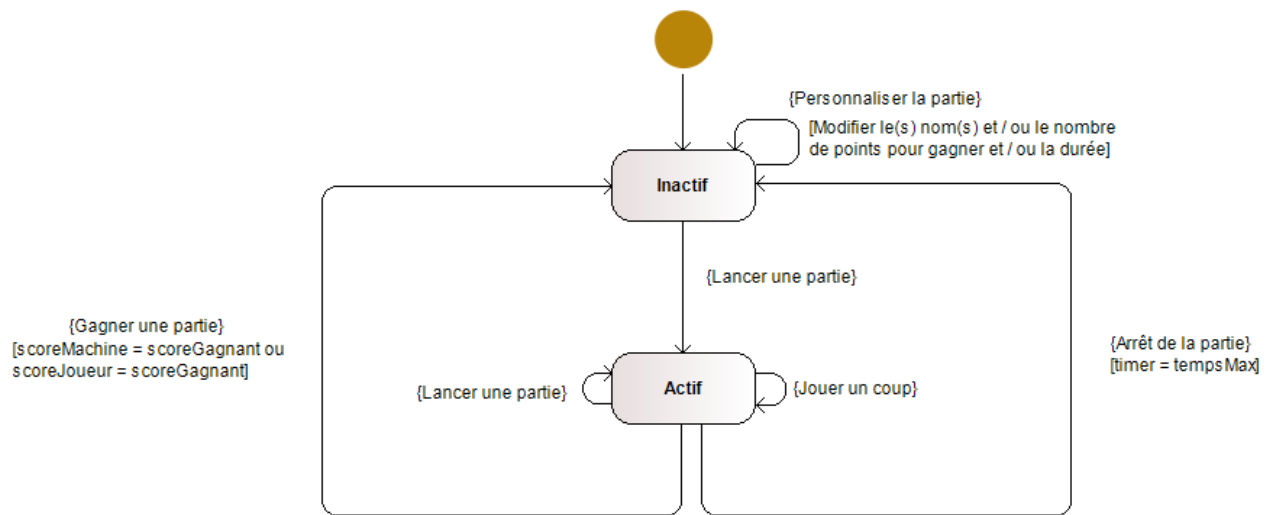


Figure 1 : Diagramme états-transitions

(k) Dictionnaires des états, événements et Actions

Dictionnaire des états du jeu

nomEtat	Signification
Inactif	Le jeu avant qu'on lance une partie
Actif	Le jeu pendant que la partie est en cours

Tableau 2 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

nomEvénement	Signification
Lancer une partie	Met le jeu en état actif avec les scores des deux joueurs à zéro et

	leurs coups à "rien".
Jouer un coup	Fait tourner le jeu en déterminant à chaque partie qui est le gagnant en augmentant les scores.
Gagner une partie	Met le jeu en état inactif à condition que scoreMachine = scoreGagnant ou scoreJoueur = scoreGagnant
Arrêt de la partie	Met le jeu en état inactif à condition que timer = tempsMax
Personnaliser la partie	Personnalisation du jeu (facultatif)

Tableau 3 : Événements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

Inactif -> Actif (Lancer une partie)	Le joueur vient d'arriver sur l'application et souhaite lancer une partie
Actif (Jouer une partie)	Le joueur choisit un coup et joue contre la machine.
Actif (Lancer une partie)	Permet de mettre le jeu à zéro et de recommencer une partie
Actif -> Inactif (Gagner une partie)	La partie est terminée, la machine ou le joueur a atteint X points.
Actif -> Inactif (Arrêt de la partie)	La partie est terminée, le temps est arrivé à X secondes
Inactif (Personnaliser la partie)	Le joueur souhaite personnaliser son jeu avant de lancer une partie

Tableau 4 : Actions à réaliser lors des changements d'état

(I) Préparation au codage :

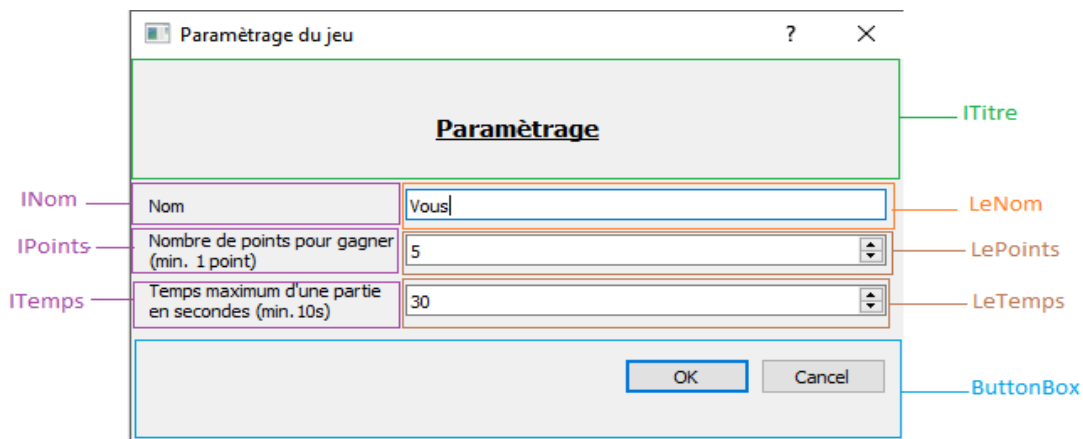
Table T_EtatsEvenementsJeu correspondant à la version matricielle du diagramme états-transitions du jeu :

- en ligne : les **événements** faisant changer le jeu d'état
- en colonne : les **états** du jeu

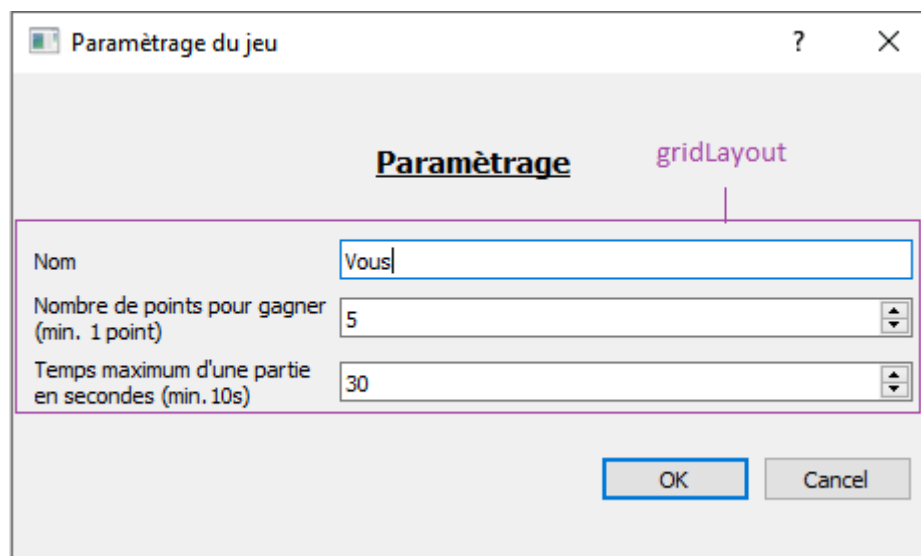
Événement nomEtatJeu	Jouer un coup	Lancer une partie	Gagner une partie	Arrêt d'une partie	Personnaliser la partie
Inactif		Actif			Inactif
Actif	Actif	Actif	Inactif	Inactif	

Tableau 5 : Matrice d'états-transitions du jeu chifoumi

22. Nouveaux éléments d'interface



Objets graphiques



Layouts

23. Liste des fichiers sources de cette version

chifoumi.pro : Fichier du projet Qt

main.cpp : Fichier permettant l'exécution du programme

chifoumi.cpp : Fichier contenant la définition des fonctions / procédures (disponibles dans le h) nécessaire à la bonne exécution du programme

chifoumi.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du programme

chifoumi.ui : Fichier permettant la réalisation et le placement des éléments graphiques du jeu.

chifoumires.qrc : Fichier de ressources Qt

chifoumivue.cpp: Fichier contenant la définition des fonctions/procédures nécessaire à la bonne exécution du programme (partie vue du chifoumi).

chifoumivue.h: Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du programme

parametrage.cpp: Fichier contenant la définition des fonctions / procédures (disponibles dans le h) nécessaire à la bonne exécution du dialogue (paramétrage).

parametrage.h: Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du dialogue (paramétrage).

parametrage.ui: Fichier permettant la réalisation et le placement des éléments graphiques du dialogue (paramétrage).

images/ciseau.gif: Permet la sélection du coup "Ciseau" pour que le joueur joue.

images/ciseau_115.png: Permet l'affichage du coup "Ciseau" pour le coup joué de la machine ou du joueur.

images/papier.gif: Permet la sélection du coup "Papier" pour que le joueur joue.

images/papier_115.png: Permet l'affichage du coup "Papier" pour le coup joué de la machine ou du joueur.

images/pierre.gif: Permet la sélection du coup "Pierre" pour que le joueur joue.

images/pierre_115.png: Permet l'affichage du coup "Pierre" pour le coup joué de la machine ou du joueur.

images/rien_115.png: Permet l'affichage d'aucun coup à chaque lancement d'une partie pour la machine ou le joueur.

images/versus.gif: Permet l'affichage entre les deux coups.

24. Fichiers .h modifiés

```

/*****
* Name:    parametrage.h
* Author:  Samuel HENTRICS LOISTINE, Ahmed FAKHFAKH, Cédric ETCHEPARE
* Created: 2022-05-27
* Description: Permet le paramétrage du jeu
*****/

#ifndef PARAMETRAGE_H
#define PARAMETRAGE_H

#include <QDialog>

namespace Ui {
class Parametrage;
}

class Parametrage : public QDialog
{
    Q_OBJECT

    /*** Méthodes de la vue
public:
    explicit Parametrage(QWidget *parent = nullptr);
    ~Parametrage();

    // Setters
public:
    void setNom(QString nom);
    /* Permet de mettre dans le champ de saisie, le nom actuel du
    * joueur de la partie*/
    void setPoints(unsigned int points);
    /* Permet de mettre dans le champ de saisie, les points actuels
    * pour gagner une partie*/
    void setTemps(unsigned int temps);
    /* Permet de mettre dans le champ de saisie, le temps maximum
    * actuel pour gagner une partie*/

```

```

    /** Slots
public:
    QString getNom();
    // Retourne le nom saisi par l'utilisateur
    unsigned int getPoints();
    // Retourne le nombre max. de points saisi par l'utilisateur
    unsigned int getTemps();
    // Retourne le temps max. à jouer saisi par l'utilisateur

private slots:
    void verifierNom();
    /* Permet de vérifier que le nom n'est pas vide*/

    /** Attributs de la vue
private:
    Ui::Parametrage *ui;
};

#endif // PARAMETRAGE_H

```

Extrait du parametrage.h

setNom() : Permet de modifier le contenu de la variable nom.
 setPoints() : Permet de modifier le contenu de la variable points.
 setTemps() : Permet de modifier le contenu de la variable temps.
 getNom() : Permet de retourner le contenu de la variable nom.
 getPoints() : Permet de retourner le contenu de la variable points.
 getTemps() : Permet de retourner le contenu de la variable temps.
 verifierNom() : Permet de vérifier que le contenu de la variable nom est non vide.

Nous avons séparé l'ancien fichier chifoumi.cpp et chifoumi.h en chifoumiVue.h/.cpp (partie Vue) et chifoumi.cpp/.h (partie Métier).

25. Résultats des tests réalisés

Testeur : Ahmed FAKHFAKH
 Élément testé : paramettrerJeu()

Date: 30/05/2022
 Version : 6.0

Classe	Description	État
valide n°1	Ouvrir la fenêtre de dialogue et rien changer (figure 1).	Validé ▾
valide n°2	Ouvrir la fenêtre de dialogue et modifier qu'un seul paramètre (le nom par exemple) (figure 2).	Validé ▾
valide n°3	Ouvrir la fenêtre de dialogue et modifier tous les paramètres (figure 3).	Validé ▾
valide n°4	Le bouton OK de la fenêtre de dialogue est bloqué tant que le champ nom est vide (figure 4).	Validé ▾
valide n°5	Le paramétrage est impossible à faire lorsqu'une partie est lancée.	Validé ▾

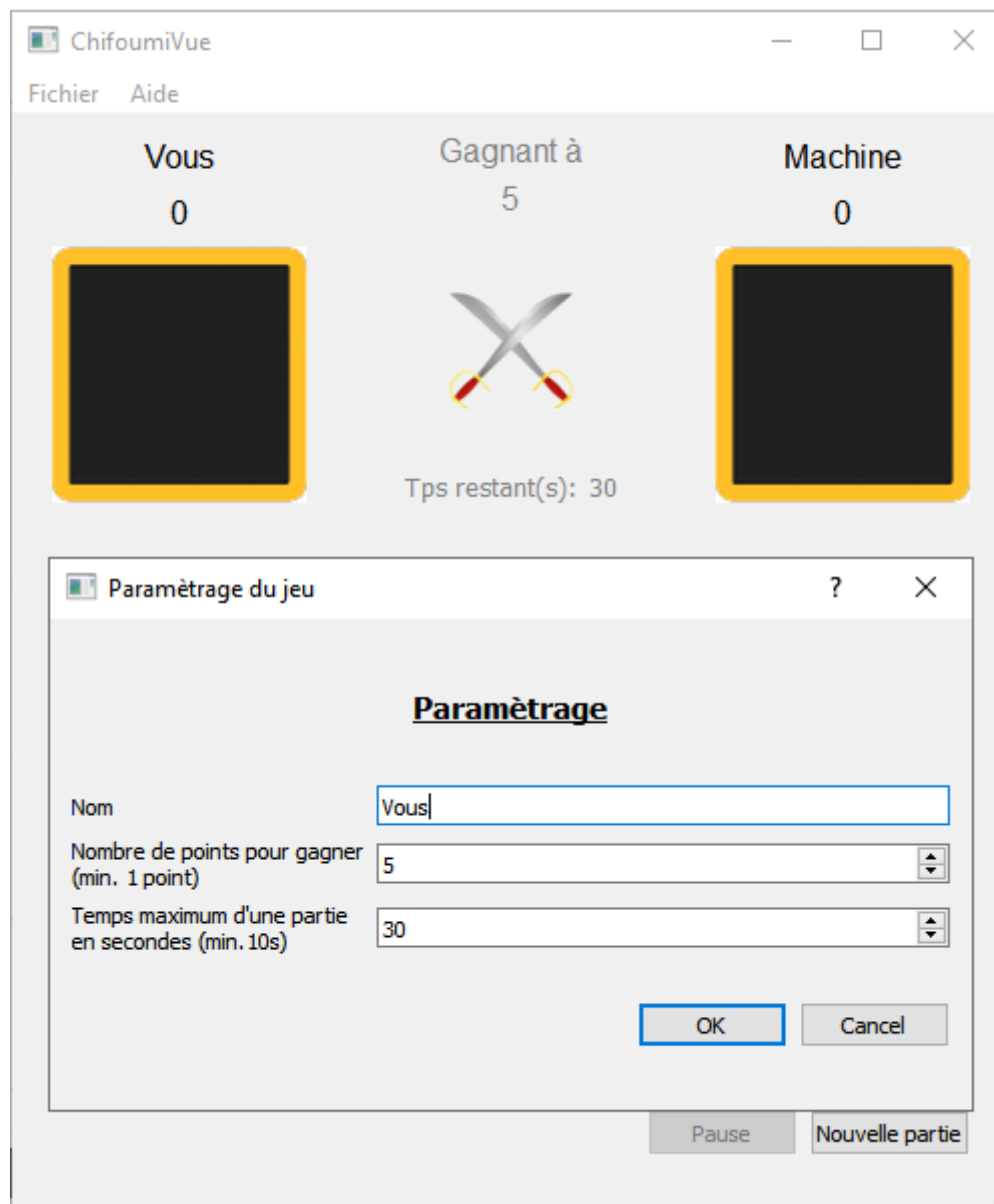


Figure 1. Rien modifier

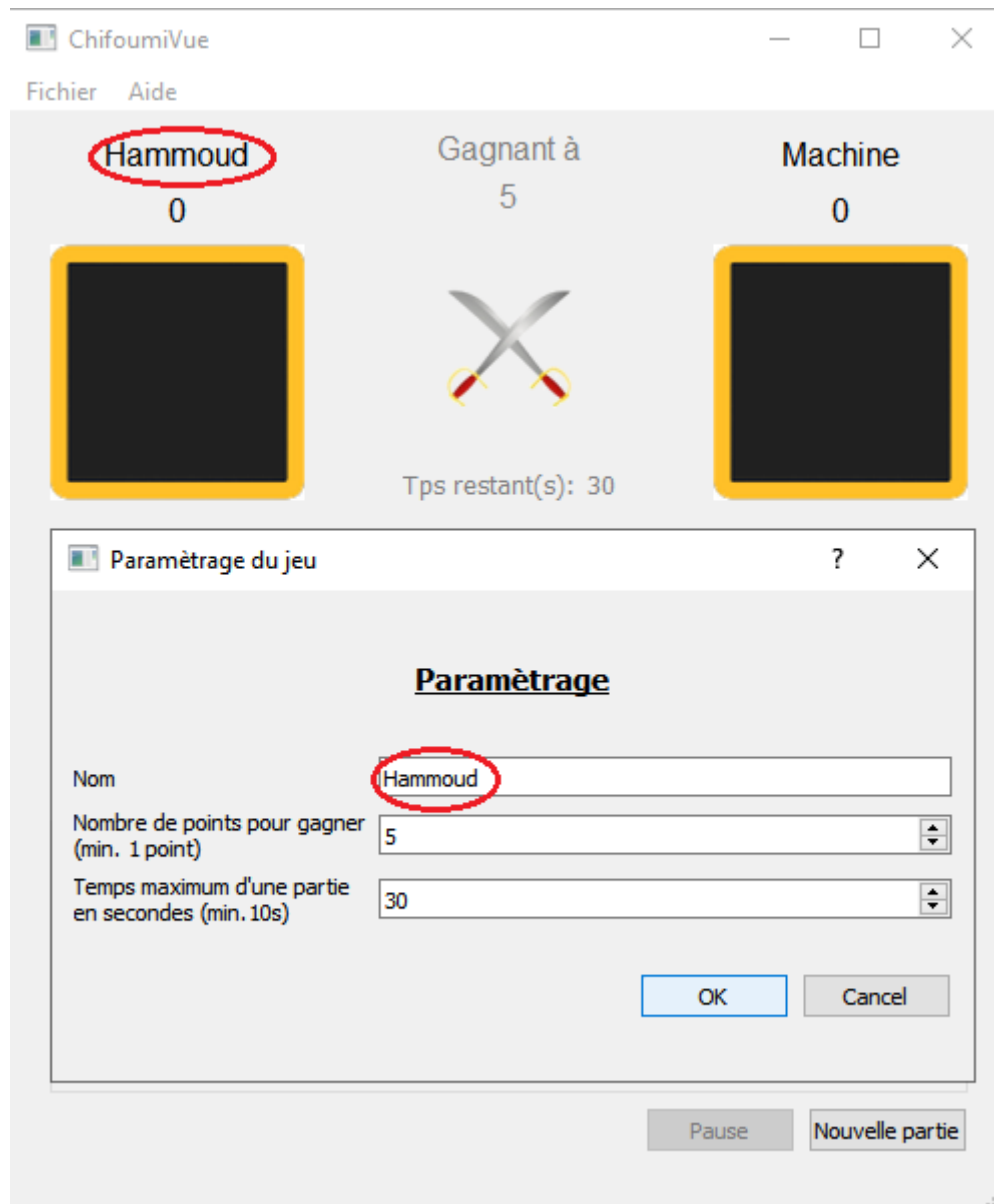


Figure 2. Modifier qu'un seul paramètre (nom)

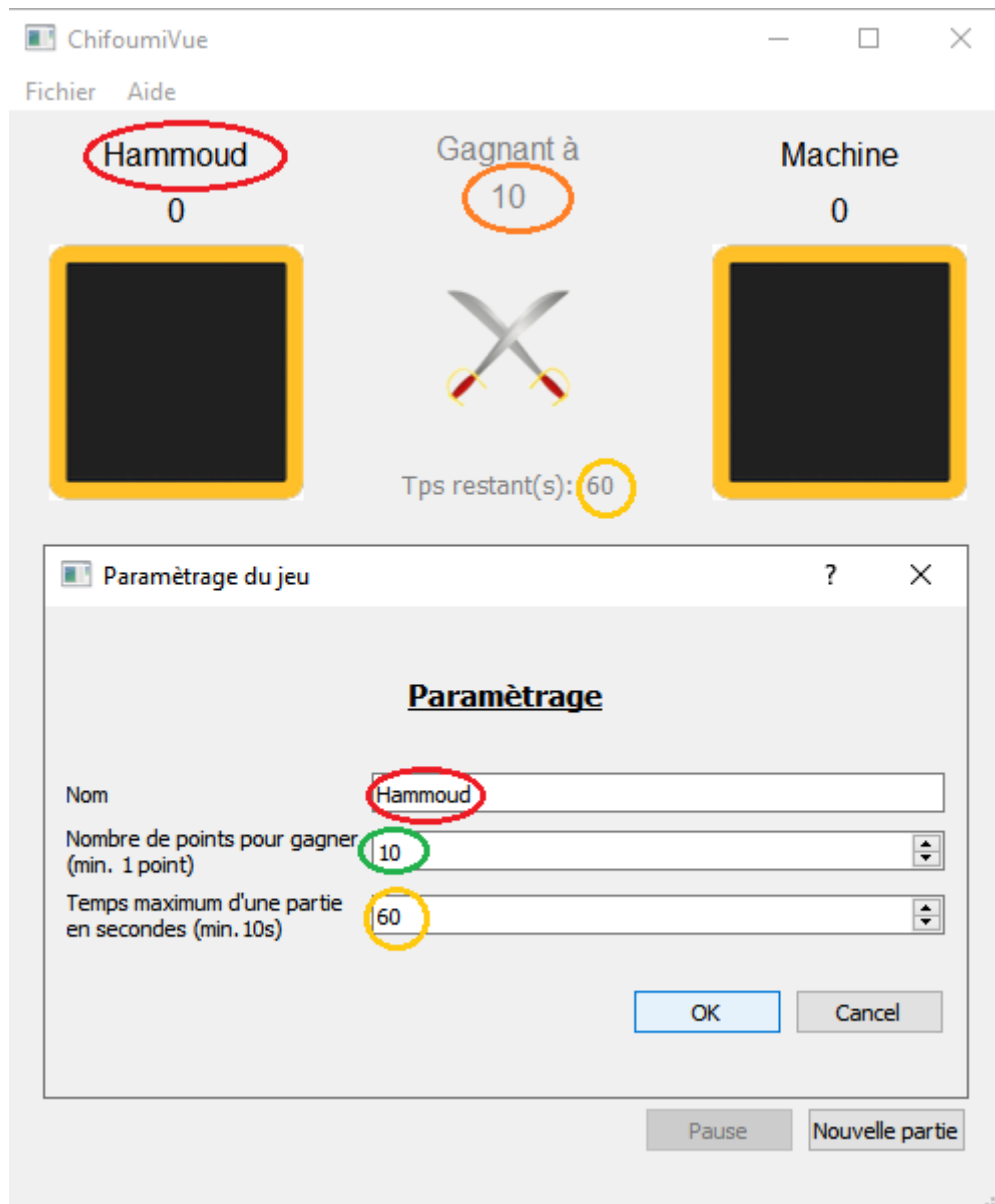


Figure 3. Modifier tous les paramètres

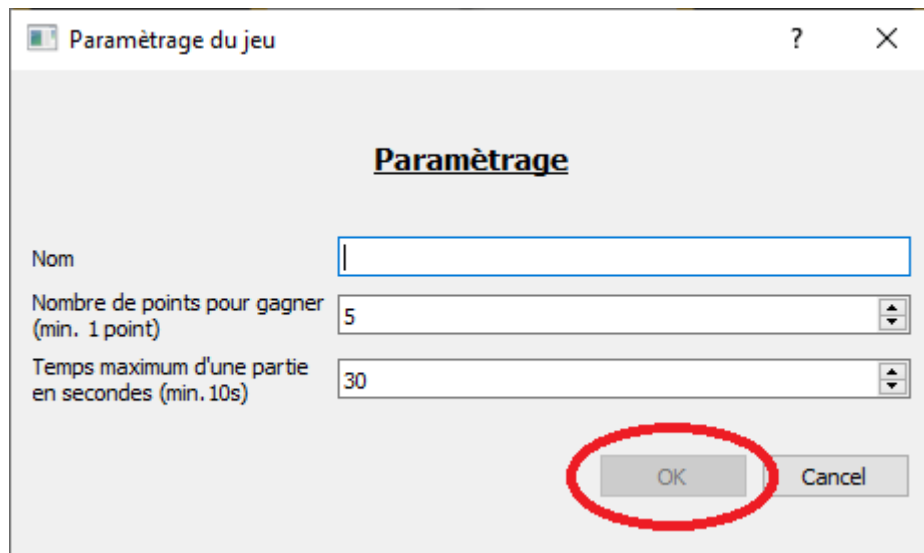


Figure 4. Bouton OK bloqué

Version 7

26. Classe Chifoumi : Diagramme états-transitions

(m) Diagramme états-transitions -actions du jeu

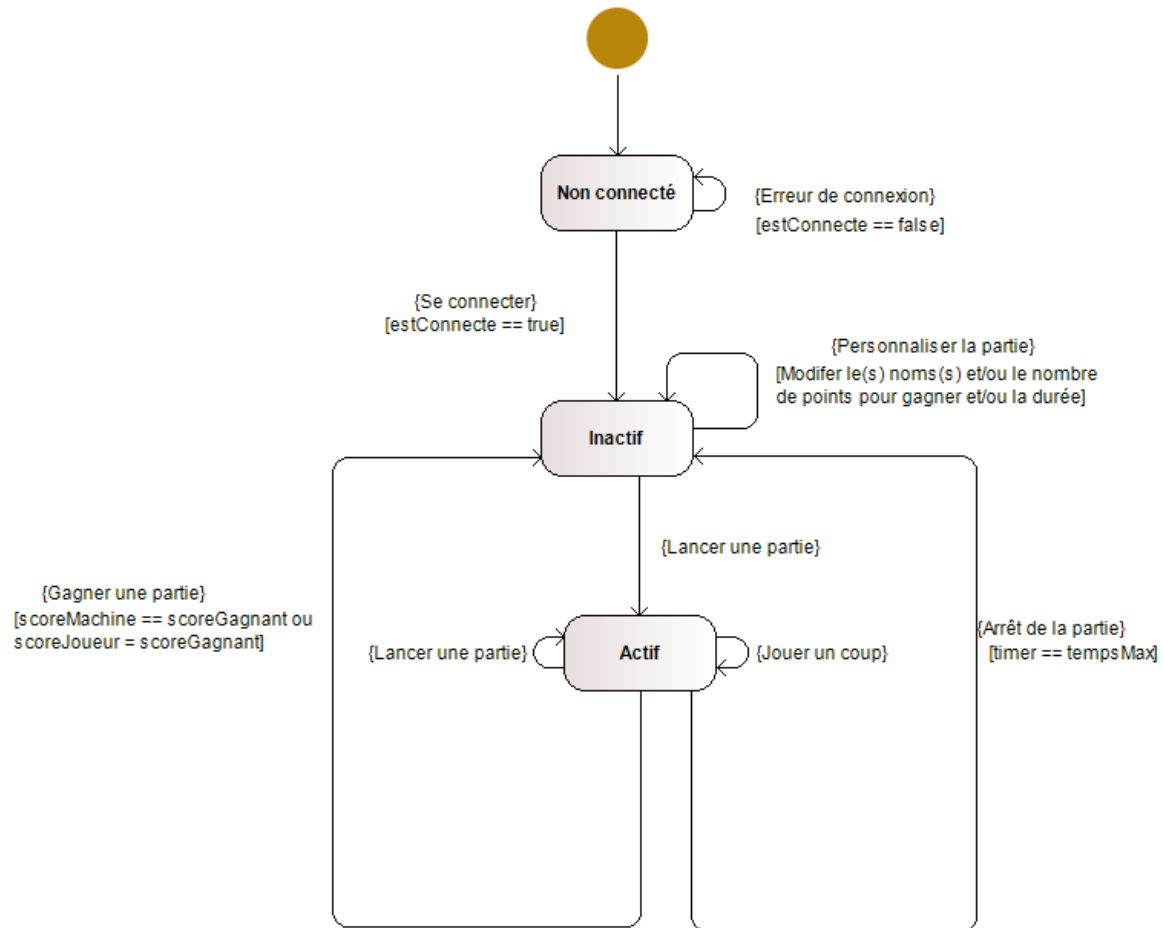


Figure 1 : Diagramme états-transitions

(n) Dictionnaires des états, événements et Actions

Dictionnaire des états du jeu

nomEtat	Signification
Non connecté	Le jeu avant vérification de l'existence de l'utilisateur dans la base de données
Inactif	Le jeu après vérification de l'existence de l'utilisateur dans la base de données et avant qu'on lance une partie.

Actif	Le jeu pendant que la partie est en cours
-------	---

Tableau 2 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

<i>nomÉvénement</i>	<i>Signification</i>
Lancer une partie	Met le jeu en état actif avec les scores des deux joueurs à zéro et leurs coups à "rien".
Jouer un coup	Fait tourner le jeu en déterminant à chaque partie qui est le gagnant en augmentant les scores.
Gagner une partie	Met le jeu en état inactif à condition que scoreMachine = scoreGagnant ou scoreJoueur = scoreGagnant
Arrêt de la partie	Met le jeu en état inactif à condition que timer = tempsMax
Personnaliser la partie	Personnalisation du jeu (facultatif)
Se connecter	Met le jeu en état inactif à condition que que estConnecte == true
Erreur dans la connexion	Laisse le jeu en non connecté et redemande le nom d'utilisateur / mot de passe à condition que estConnecte == false

Tableau 3 : Événements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

Inactif -> Actif (Lancer une partie)	Le joueur vient d'arriver sur l'application et souhaite lancer une partie
Actif (Jouer une partie)	Le joueur choisit un coup et joue contre la machine.
Actif (Lancer une partie)	Permet de mettre le jeu à zéro et de recommencer une partie
Actif -> Inactif (Gagner une partie)	La partie est terminée, la machine ou le joueur a atteint X points.
Actif -> Inactif (Arrêt de la partie)	La partie est terminée, le temps est arrivé à X secondes
Inactif (Personnaliser la partie)	Le joueur souhaite personnaliser son jeu avant de lancer une partie
Non connecté -> Inactif (Se connecté)	Le joueur se connecte afin d'accéder au jeu
Non connecté -> Non	La connexion a échoué car le nom d'utilisateur / mot de passe est

connecté (Erreur dans la connexion)	incorrect ou alors il y a problème de connexion à la base de données
-------------------------------------	--

Tableau 4 : Actions à réaliser lors des changements d'état

(o) **Préparation au codage :**

Table T_EtatsEvenementsJeu correspondant à la version matricielle du diagramme états-transitions du jeu :

- en ligne : les **événements** faisant changer le jeu d'état
- en colonne : les **états** du jeu

Événement <i>nomEtatJeu</i>	Jouer un coup	Lancer une partie	Gagner une partie	Arrêt d'une partie	Personnaliser la partie	Connexion	Erreur dans la connexion
Non connecté						Inactif	Non-connecté
Inactif		Actif			Inactif		
Actif	Actif	Actif	Inactif	Inactif			

Tableau 5 : Matrice d'états-transitions du jeu chifoumi

(p) **Schéma Relationnel de la base de données**

UTILISATEURS(nom, mdp);

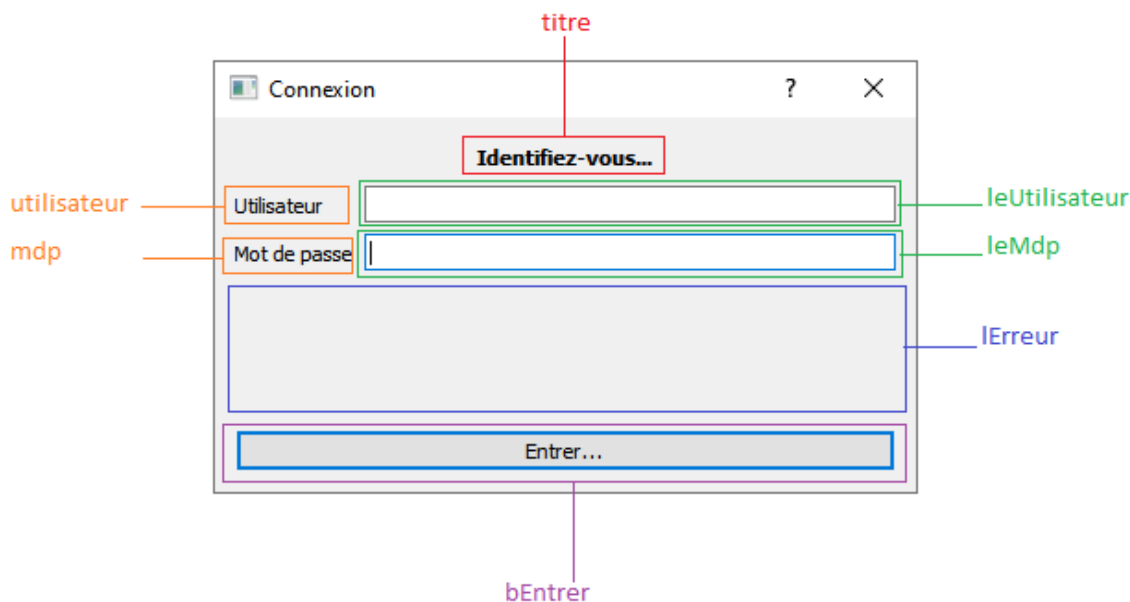
Légende

NOMDETABLEENMAJUSCULE(listedeschamps séparés par des virgules) Clés primaires : soulignées Clés étrangères : précédées de #

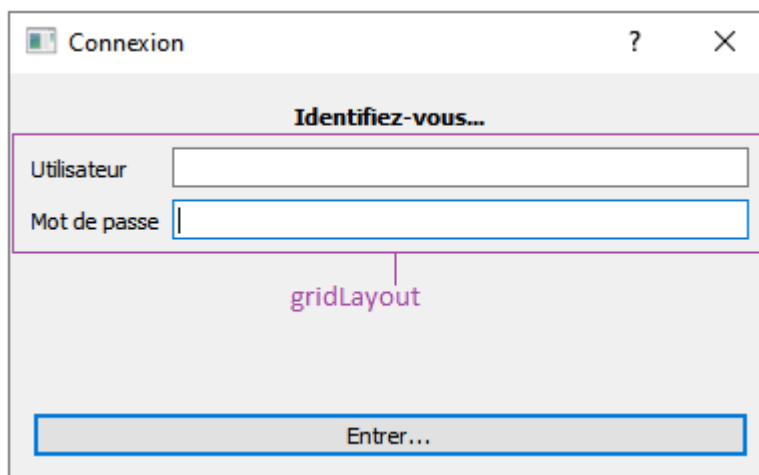
(q) **Dictionnaire des données de la base de données**

Libellé	Signification	Domaine			Taille	Contraintes	Exemple
		Niveau conceptuel	Niveau logique	Niveau physique (MySQL)			
Utilisateurs							
nom	Nom de l'utilisateur	alphanumérique	chaîne de caractères	varchar2	25	Identifiant unique	Eliott
mdp	Mot de passe de l'utilisateur	alphanumérique	chaîne de caractères	varchar2	25	Obligatoirement renseigné	test

27. Nouveaux éléments d'interface



Objets graphiques



Layouts

28. Liste des fichiers sources de cette version

chifoumi.pro : Fichier du projet Qt

main.cpp : Fichier permettant l'exécution du programme

chifoumi.cpp : Fichier contenant la définition des fonctions / procédures (disponibles dans le

h) nécessaire à la bonne exécution du programme
chifoumi.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du programme
chifoumi.ui : Fichier permettant la réalisation et le placement des éléments graphiques du jeu.
chifoumires.qrc : Fichier de ressources Qt
chifoumivue.cpp : Fichier contenant la définition des fonctions/procédures nécessaire à la bonne exécution du programme (partie vue du chifoumi).
chifoumivue.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du programme
parametrage.cpp : Fichier contenant la définition des fonctions / procédures (disponibles dans le h) nécessaire à la bonne exécution du dialogue (paramétrage).
parametrage.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du dialogue (paramétrage).
parametrage.ui : Fichier permettant la réalisation et le placement des éléments graphiques du dialogue (paramétrage).
connexion.cpp : Fichier contenant la définition des fonctions / procédures (disponible dans le h) nécessaire à la bonne exécution de la fenêtre connexion.
connexion.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution de la fenêtre connexion.
connexion.ui : Fichier permettant la réalisation et le placement des éléments graphiques du dialogue (paramétrage).
database.cpp : Fichier contenant la définition des fonctions / procédures (disponible dans le h) nécessaire à la bonne exécution de la base de données.
database.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution de la base de données.
images/ciseau.gif : Permet la sélection du coup "Ciseau" pour que le joueur joue.
images/ciseau_115.png : Permet l'affichage du coup "Ciseau" pour le coup joué de la machine ou du joueur.
images/papier.gif : Permet la sélection du coup "Papier" pour que le joueur joue.
images/papier_115.png : Permet l'affichage du coup "Papier" pour le coup joué de la machine ou du joueur.
images/pierre.gif : Permet la sélection du coup "Pierre" pour que le joueur joue.
images/pierre_115.png : Permet l'affichage du coup "Pierre" pour le coup joué de la machine ou du joueur.
images/rien_115.png : Permet l'affichage d'aucun coup à chaque lancement d'une partie pour la machine ou le joueur.
images/versus.gif : Permet l'affichage entre les deux coups.

29. Fichiers .h modifiés

```

/*****
* Name:    connexion.h
* Author:   Samuel HENTRICS LOISTINE, Ahmed FAKHFAKH, Cédric ETCHEPARE
* Created:  2022-05-30
* Description : Chifoumi v7
*****/

#ifndef CONNEXION_H
#define CONNEXION_H

#include <QDialog>
#include "database.h"

namespace Ui {

```

```

class Connexion;
}

class Connexion : public QDialog
{
    Q_OBJECT

public:
    explicit Connexion(QWidget *parent = nullptr);
    ~Connexion();

    /** Procédures publiques de la vue
public:
    QString getNom();
    /* Retourne le nom d'utilisateur saisi dans le formulaire */
    bool infoConnexion();
    /* Retourne vrai si le joueur est connecté, faux si celui-ci ne l'est pas */

    /** Attributs privés de la vue
private:
    Ui::Connexion *ui;
    bool estConnecte;
    /* Statut du joueur si celui-ci est connecté */
    Database *db = new Database();
    /* Base de données */

    /** Slots privés de la vue
private slots:
    void demanderConnexion();
    // Permet de vérifier si l'utilisateur a renseigné les bonnes informations de connexion
};

#endif // CONNEXION_H

```

connexion.h

```

/*****
* Name:    database.h
* Author:   Samuel HENTRICS LOISTINE, Ahmed FAKHFAKH, Cédric ETCHEPARE
* Created:  2022-05-30
* Description : Chifoumi v7
*****/

#ifndef DATABASE_H
#define DATABASE_H

#include <QSqlDatabase>
#include <QSqlQuery>
#include <QDebug>
#include <QSqlError>
#include <QVariantList>

#define DATABASE_NAME "bdd_chifoumi"
#define CONNECT_TYPE "QODBC"

class Database
{
public:
    Database();
    bool openDatabase();
    // renvoie vrai si la base s'ouvre, faux sinon
    void closeDatabase();
    // Permet de fermer la base de donnée
    bool restoreDatabase();
    // Retourne faux si impossible d'accéder la bdd et vrai sinon
    bool verifierMotDePasse(QString nom, QString mdp);

```



```

/* Retourne vrai si le mot de passe est celui de l'utilisateur, faux sinon. */
bool updateNomUtilisateur(QString ancienNom, QString nouveauNom);
/* Essaie de modifier le nom d'un utilisateur, retourne vrai si cela a marché
* faux, si le nom d'utilisateur existe déjà*/

```

```
private:
```

```
    QSqlDatabase mydb;
```

```
private:
```

```
    bool createTable();
```

```
    // permet de créer le table Identifiants
```

```
    bool insertTable(const QVariantList &);
```

```
    // permet de peupler la table Identifiants
```

```
};
```

```
#endif // DATABASE_H
```

database.h

```
....
```

```
    ///* Attributs privés de la vue
```

```
private:
```

```
    unsigned int scoreGagnant; // score à atteindre pour gagner
```

```
    unsigned int tempsPartie; // Temps par défaut avant la fin d'une partie
```

```
    unsigned int tempsRestant; // Temps restant pour la partie
```

```
    Chifoumi *leJeu = new Chifoumi();
```

```
    QString nomJoueur; // nom du joueur
```

```
    QTimer *timer = new QTimer(this); // timer qui s'enclenche toutes les 1 secondes
```

```
    Connexion *conn = new Connexion(this);
```

```
    Parametrage *param = new Parametrage(this); // Boite de dialogue pour paramétrer le jeu
```

```
    Database *db = new Database(); // Base de données du jeu
```

```
    QMessageBox* mbox = new QMessageBox(); // Permet l'affichage de message d'information (gagnant par exemple)
```

```
...
```

```
    ///* Slots privés de la vue
```

```
private slots:
```

```
    void lancerPartie();
```

```
    /* Permet de lancer une partie entre le joueur et la machine
```

```
    */
```

```
    void jouerCiseau();
```

```
    /* Le joueur décide de jouer ciseau */
```

```
    void jouerPapier();
```

```
    /* Le joueur décide de jouer papier */
```

```
    void jouerPierre();
```

```
    /* Le joueur décide de jouer pierre */
```

```
    void jouerPartie(Chifoumi::UnCoup coup);
```

```
    /* Permet de déterminer le gagnant et met à jour l'interface
```

```
        à partir d'un coup (coup) donné par le joueur */
```

```
    void aProposDe();
```

```
    /* Permet l'affichage "A propos de..." pour l'utilisateur */
```

```
    void finirPartie();
```

```
    /* Permet de finir la partie lorsque un joueur a atteint 5 points */
```

```
    void majTemps();
```

```
    /* Met à jour le temps restant lors d'une partie et peut arreter la partie
```

```
        si le compteur est à zero*/
```

```
    void majPause();
```

```
    /* Met le jeu en pause lorsque l'utilisateur demande à mettre le jeu en pause.
```

```
        * Ou alors reprend la partie si le timer est inactif*/
```

```
    void paramétrerJeu();
```

```
    /* Procédure permettant à l'utilisateur de paramétrer le jeu
```

```
        * (son nom, la durée d'une partie, le nombre de points pour gagner) */
```

```
};
```

```
#endif // CHIFOUMIVUE_H
```

chifoumiVue.h

30. Résultats des tests réalisés

Testeur : Samuel HENTRICS LOISTINE

Date: 05/06/2022

Élément testé : restoreDatabase()

Version : 7.0

Classe	Description	État
valide n°1	La base a été créée et l'utilisateur Elliot a été ajouté (Figure 1)	Validé ▾



Figure 1. La base de données est créée

Testeur : Samuel HENTRICS LOISTINE

Date: 05/06/2022

Élément testé : demanderConnexion()

Version : 7.0

Classe	Description	État
valide n°1	Le DSN est inexistant (Figure 2).	Validé ▾
valide n°2	Le joueur a saisi un identifiant incorrect (figure 3).	Validé ▾
valide n°3	Le joueur a saisi un identifiant correct (Figure 4)	Validé ▾



Figure 2. DSN Inexistant

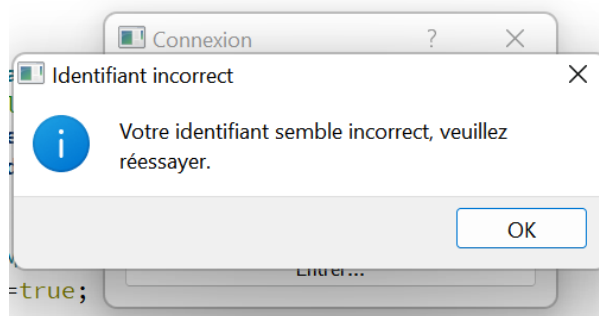


Figure 3. L'identifiant est incorrect



Figure 4. Le joueur est connecté

Classe	Description	État
valide n°1	Le joueur change son nom et celui-ci n'est pas déjà pris dans la base de données (Figure 5).	Validé ▾
valide n°2	Le joueur a saisi un identifiant déjà pris en modifiant son nom d'utilisateur (figure 6).	Validé ▾



Figure 5: L'utilisateur a modifié son nom d'utilisateur



Figure 6 : L'utilisateur a tenté de modifier son nom d'utilisateur mais le nom est déjà pris

Version 8

31. Base de données

(r) Schéma Relationnel de la base de données

UTILISATEURS(id, nom, mdp);

RESULTATS(id, horodatage, id_joueur, scoreJoueur, scoreMachine)

Légende

NOMDETABLEENMAJUSCULE(listedeschampsséparéspardesvirgules)
Clés primaires : soulignées Clés étrangères : précédées de #

(s) Dictionnaire des données de la base de données

Libellé	Signification	Domaine			Taille	Contraintes	Exemple
		Niveau conceptuel	Niveau logique	Niveau physique			

				(MySQL)			
Utilisateurs							
id	Code unique du joueur	entier	entier	int	4	Identifiant unique S'incrémente automatiquement	1
nom	Nom de l'utilisateur	alphanumérique	chaîne de caractères	varchar2	25	Obligatoirement renseigné	Eliott
mdp	Mot de passe de l'utilisateur	alphanumérique	chaîne de caractères	varchar2	25	Obligatoirement renseigné	test
Resultats							
id	Code unique du résultat	entier	entier	int	4	Identifiant unique S'incrémente automatiquement	1
horodatage	Heure à laquelle le jeu démarre	date	date	datetime		Valeur par défaut : current_timestamp()	2022-06-06 22:56:02
joueur_id	Code unique du joueur	entier	entier	int	4	Clé étrangère de la clé primaire utilisateur Obligatoirement renseigné	1
scoreJoueur	Score du joueur	entier	entier	int	4	Obligatoirement renseigné	4
scoreMachine	Score de la machine	entier	entier	int	4	Obligatoirement renseigné	3

32. Liste des fichiers sources de cette version

chifoumi.pro : Fichier du projet Qt

main.cpp : Fichier permettant l'exécution du programme

chifoumi.cpp : Fichier contenant la définition des fonctions / procédures (disponibles dans le h) nécessaire à la bonne exécution du programme

chifoumi.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du programme

chifoumi.ui : Fichier permettant la réalisation et le placement des éléments graphiques du jeu.

chifoumires.qrc : Fichier de ressources Qt

chifoumivue.cpp : Fichier contenant la définition des fonctions/procédures nécessaire à la bonne exécution du programme (partie vue du chifoumi).

chifoumivue.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du programme

parametrage.cpp : Fichier contenant la définition des fonctions / procédures (disponibles dans le h) nécessaire à la bonne exécution du dialogue (paramétrage).

parametrage.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution du dialogue (paramétrage).

parametrage.ui : Fichier permettant la réalisation et le placement des éléments graphiques du dialogue (paramétrage).

connexion.cpp : Fichier contenant la définition des fonctions / procédures (disponible dans le h) nécessaire à la bonne exécution de la fenêtre connexion.

connexion.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution de la fenêtre connexion.

connexion.ui : Fichier permettant la réalisation et le placement des éléments graphiques du dialogue (paramétrage).

database.cpp : Fichier contenant la définition des fonctions / procédures (disponible dans le h) nécessaire à la bonne exécution de la base de données.

database.h : Fichier contenant la déclaration des fonctions / procédures nécessaire à la bonne exécution de la base de données.

images/ciseau.gif : Permet la sélection du coup "Ciseau" pour que le joueur joue.

images/ciseau_115.png : Permet l'affichage du coup "Ciseau" pour le coup joué de la machine ou du joueur.

images/papier.gif : Permet la sélection du coup "Papier" pour que le joueur joue.

images/papier_115.png : Permet l'affichage du coup "Papier" pour le coup joué de la machine ou du joueur.

images/pierre.gif : Permet la sélection du coup "Pierre" pour que le joueur joue.

images/pierre_115.png : Permet l'affichage du coup "Pierre" pour le coup joué de la machine ou du joueur.

images/rien_115.png : Permet l'affichage d'aucun coup à chaque lancement d'une partie pour la machine ou le joueur.

images/versus.gif : Permet l'affichage entre les deux coups.

33. Fichiers .h modifiés

```
...

public:
    Database();
    bool openDatabase();
    // renvoie vrai si la base s'ouvre, faux sinon
    void closeDatabase();
    // Permet de fermer la base de donnée
    bool restoreDatabase();
    // Retourne faux si impossible d'accéder la bdd et vrai sinon
    bool verifierMotDePasse(QString nom, QString mdp);
    /* Retourne vrai si le mot de passe est celui de l'utilisateur, faux sinon. */
    bool updateNomUtilisateur(QString ancienNom, QString nouveauNom);
    /* Essaye de modifier le nom d'un utilisateur, retourne vrai si cela a marché
    * faux, si le nom d'utilisateur existe déjà */
    bool insertResultat(QString nom, unsigned int scoreJoueur, unsigned int scoreMachine);
    /* Permet d'insérer les résultats d'une partie finie */
```

database.h

```
...

///* Slots privés de la vue
private slots:
    void lancerPartie();
    /* Permet de lancer une partie entre le joueur et la machine
    */
```


<pre> void jouerCiseau(); /* Le joueur décide de jouer ciseau */ void jouerPapier(); /* Le joueur décide de jouer papier */ void jouerPierre(); /* Le joueur décide de jouer pierre */ void jouerPartie(Chifoumi::UnCoup coup); /* Permet de déterminer le gagnant et met à jour l'interface à partir d'un coup (coup) donné par le joueur */ void aProposDe(); /* Permet l'affichage "A propos de..." pour l'utilisateur */ void finirPartie(); /* Permet de finir la partie lorsque un joueur a atteint 5 points */ void majTemps(); /* Met à jour le temps restant lors d'une partie et peut arreter la partie si le compteur est à zero*/ void majPause(); /* Met le jeu en pause lorsque l'utilisateur demande à mettre le jeu en pause. * Ou alors reprend la partie si le timer est inactif*/ void paramettrerJeu(); /* Procédure permettant à l'utilisateur de paramètrer le jeu * (son nom, la durée d'une partie, le nombre de points pour gagner) */ }; #endif // CHIFOUMIVUE_H </pre>
chifoumiVue.h

Nous avons modifié notre code pour le `restoreDatabase()`. En effet, pour éviter que le joueur Elliot se recrée à chaque partie si toutefois on avait changé le nom d'utilisateur de celui-ci, nous avons mis une condition que si toutefois il n'y a aucun utilisateur, on rajoute celui-ci par défaut.

34. Résultats des tests réalisés

Testeur : Samuel HENTRICS LOISTINE
Élément testé : `restoreDatabase()`

Date: 07/06/2022
Version : 8.0

Classe	Description	État
valide n°1	La base de données Resultats a été créée (Figure 1)	Validé 

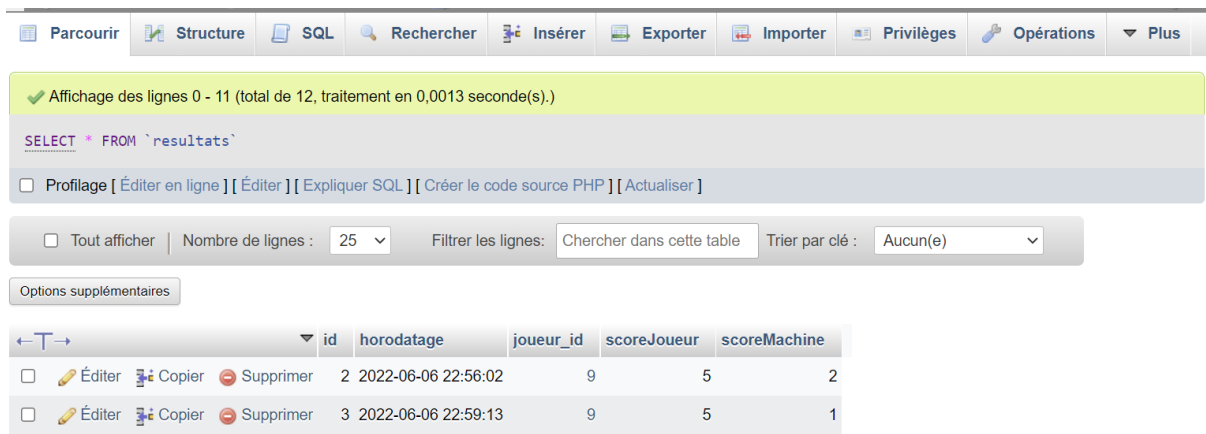


Figure 1. La base de données est créée

Testeur : Samuel HENTRICS LOISTINE
Élément testé : finirPartie()

Date: 07/06/2022
Version : 8.0

Classe	Description	État
valide n°1	Le joueur gagne et le résultat est mis dans la base de données (Figure 2)	Validé
valide n°2	La machine gagne et le résultat est mis dans la base de données (Figure 3)	Validé
valide n°3	Le temps est écoulé (Figure 4)	Validé

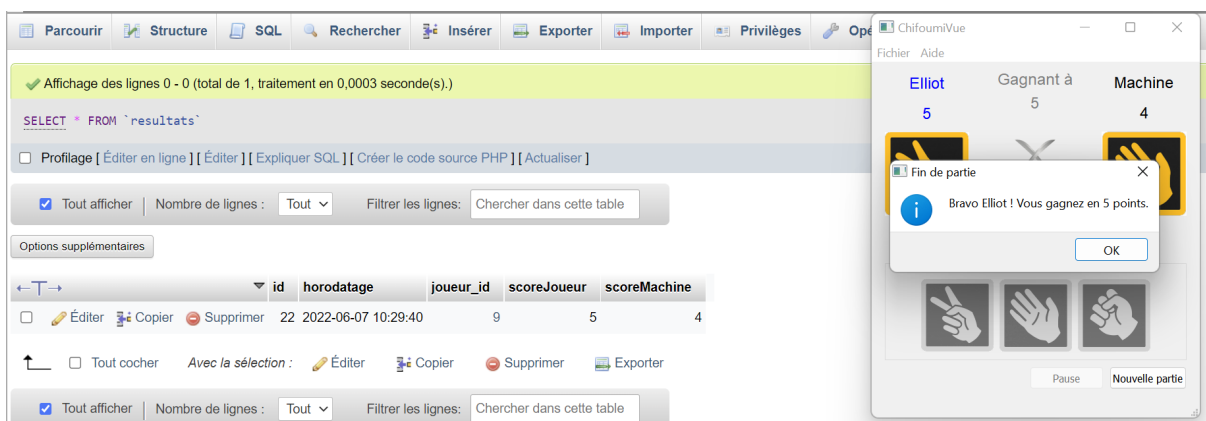


Figure 2. Le joueur gagne et le résultat est mis dans la base de données



Figure 3. La machine gagne et le résultat est mis dans la base de données

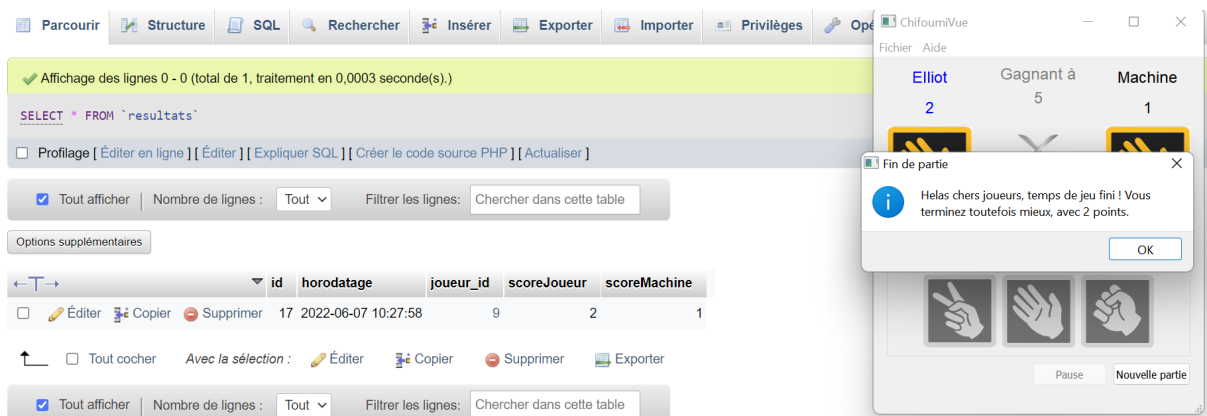


Figure 4. Le temps est écoulé