Cocheril Titouan
Salle Ivan
Kepa Eyherabide

Saé 2.01 – Développement d'une application

Chifoumi – Dossier d'Analyse et conception v4

Lien git hub : https://github.com/TitouCoch/Chifoumi-SAE-02

Sommaire :

Version Initiale :

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

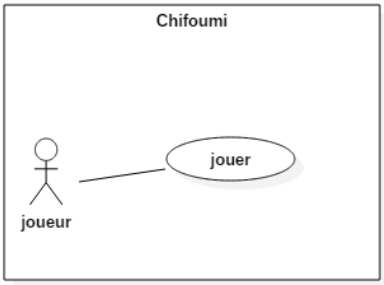


Figure 1 : Diagramme des Cas d'Utilisation du jeu Chifoumi

3-Scénarios

(a) Exemple Scénario

Titre : Jouer 2 coups
Résumé : Le joueur démarre une partie et joue 2 coups
Acteur : Utilisateur (acteur principal)
Pré-condition : Le jeu est démarré et se trouve à l'état initial.
Post-condition : néant
Date de création : 01/05/2021 **Date de mise à jour** : --
Créateur : …

Utilisateur	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.
9. Choisit une figure	10. Affiche la figure du joueur dans la zone d'affichage du dernier coup joueur.
	11. Choisit une figure.
	12. Affiche sa figure dans la zone d'affichage de son dernier coup.
	13. Détermine le gagnant et met à jour les scores.
	14. Affiche les scores.
15. Choisit une Nouvelle Partie	16. Réinitialise les scores.
	17. Réinitialise les zones d'affichage des derniers coups.
	19. Affiche les scores et les zones d'affichage des derniers coups.

Tableau 1 : Scénario nominal

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.

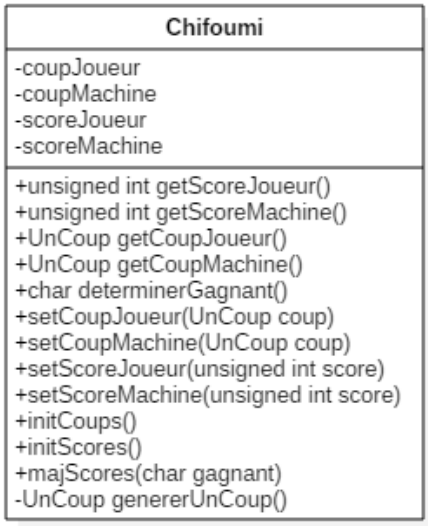


Figure 2 : Diagramme de Classes UML du jeu Chifoumi

(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`	Mémorise la dernière figure choisie par le joueur. Type énuméré `enum unCoup {pierre, ciseau, papier, rien};`	`UnCoup`	`papier`
`coupMachine`	Mémorise 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 3

```
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 coupMachine 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 3 : Schéma de classes = Une seule classe Chifoumi

Version 0 :

1-Liste des fichiers sources de cette version (et rôle de chacun)

Chifoumi.h : interface de l'application non graphique Chifoumi

```
    ///* Une définition de type énuméré
public:
    enum UnCoup {pierre, papier, ciseau, rien};
    ///* Méthodes du Modéle
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 coupMachine avec la valeur
           du paramètre p_score */

    // Autres modificateurs
     void majScores(char p_gagnant);
        /* Mise à jour des scores en fonction des règles de gestion actuelles :
            - 1 point pour le gagnant lorsqu'il y a un gagnant
            - 0 point en cas de match nul */
    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
```

Chifoumi.cpp : corps de l'application non graphique Chifoumi

Main.cpp : fichier source contenant la boucle principale qui exécute l'application non graphique et la boucle secondaire qui se met en attente des évènements

V0.pro : fichier contenant les chemins d'inclusions et l'arborescence des fichiers

Chifoumi_dossierAnalyseConception_V0 : dossier d'analyse & conception

2-Tests des méthodes :

Test	Résultat	Validation
Méthodes get() associées aux attributs 'score'	Score Joueur : 0 score Machine : 0	Ok
Méthodes get() associées aux attributs 'coup'	Coup Joueur : Rien Coup Machine : rien	Ok
Méthodes set() associées aux attributs 'score'	Score Joueur : 1 Score Machine : 2	Ok
Méthode initScores()	Score Joueur : 0 Score Machine : 0	Ok
Méthodes set() et get() associées aux attributs 'coup'/'choix'	Coup Joueur : pierre Coup Machine : ciseau	Ok
Quelques tours de jeu pour tester l'identification du gagnant et la maj des scores	Coup Joueur : pierre Coup Machine : ciseau Score Joueur : 1 Score Machine : 0	Ok
Appel du Constructeur	Scores à 0 CoupsJoueurs à rien	Ok

Version 1 :

1-Classe Chifoumi : Diagramme états-transitions

(a) Diagramme états-transitions -actions du jeu

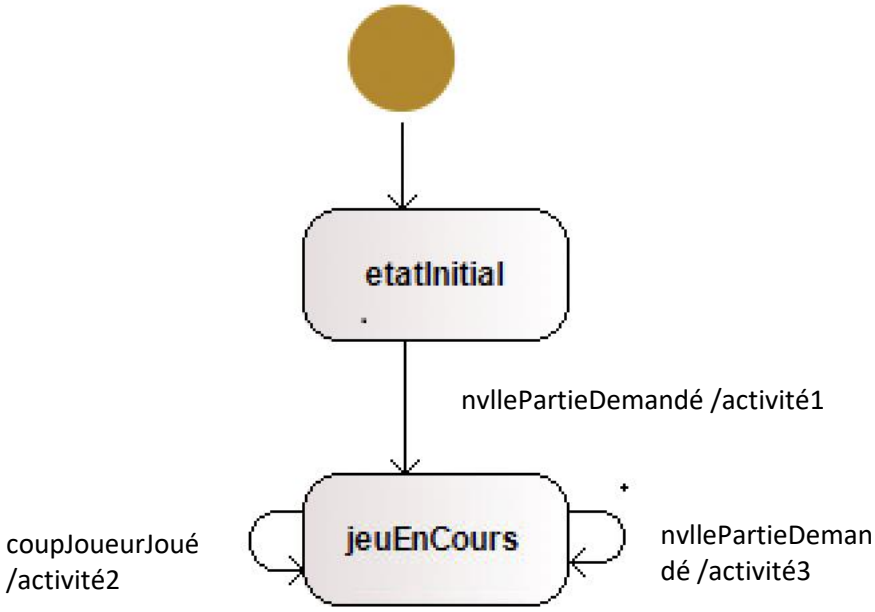


Figure 4 : Diagramme états-transitions

(b) Dictionnaires des états, événements et Actions

Dictionnaire des états du jeu

nomEtat	*Signification*
`etatInitial`	Le programme est lancé, les variables scores sont à 0 et la partie est prête à être joué. Les signes sont initialisés et l'affichage des boutons sont actifs
`jeuEnCours`	Le jeu est lancé, le joueur peut successivement avec la machine choisir son signe et les variables et fonctions seront misent à jour selon le résultat.

Tableau 3 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

nomEvénement	*Signification*
`nvllePartieDemandée`	Le joueur sélectionne nouvelle partie
`coupJoueurJoué`	Le joueur choisi un signe parmi les trois proposés (pierre, feuille, ciseau)

Tableau 4 : Evénements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

`Activité 1 :`	Système d'information : Le jeu est initialisé avec les variables coups et scores du joueur et de la machine prenant les valeurs de l'état initial Interface : Les boutons de signes sont accessibles pour le choix de l'utilisateur Les scores et nom des joueurs sont colorés en bleu pour indiquer que le jeu est en cours
`Activité 2 :`	Système d'Information : Le dernier coup du jour est mis à jour en fonction du coup joué. La machine joue (tirage aléatoire) et le dernier coup de la machine est mis à jour en conséquence. La machine détermine le gagnant et met à jour en conséquence les scores. Interface : Le signe choisi par le joueur s'affiche dans la case d'affichage prévu, de même pour celui de la machine Les scores sont mis à jour en fonction du gagnant
`Activité 3 :`	Système d'Information : Les scores Joueur/Machine sont mis à 0. Les derniers coups Joueur/Machine joués sont mis à rien. Interface : Les zones d'affichages sont misent en cohérence avec les propriétés précédentes.

Tableau 5 : 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

Événement → *nomEtatJeu*	coupJoueurJoué	nvllePartieDemandée
etatinitial		jeuEnCours/activité1
jeuEnCours	jeuEnCours / activité2	jeuEnCours/activité3

Tableau 6 : Matrice d'états-transitions du jeu chifoumi

Table `T_EtatsEvenementsJeu` avec **les éléments d'interface** prenant en charge les événements

	boutonPierre	boutonCiseau	boutonPierre	boutonNvllePartie

Événement → *nomEtatJeu*	coupJoueurJoué	nvllePartieDemandée
etatinitial		jeuEnCours/activité1
jeuEnCours	jeuEnCours / activité2	jeuEnCours/activité3

Tableau 6 : Matrice d'états-transitions du jeu chifoumi AVEC éléments d'interface

2-Liste des fichiers sources :

chifoumimodele.h : interface du modèle de l'application Chifoumi

```
class ChifoumiModele : public QObject
{
    Q_OBJECT
public:
    enum UnCoup {pierre,papier,ciseau,rien};

///* ---- PARTIE MODELE ---------------------------

    ///* Méthodes du Modéle
public:
    ChifoumiModele();

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

    // Setters
public slots:
    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 coupMachine avec la valeur
           du paramètre p_score */

    // Autres modificateurs
     void majScores(char p_gagnant);
        /* Mise à jour des scores en fonction des règles de gestion actuelles :
            - 1 point pour le gagnant lorsqu'il y a un gagnant
            - 0 point en cas de match nul
        */
    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
};
```

chifoumimodele.cpp : corp du modèle de l'application Chifoumi

chifoumivue.h : interface de la vue de l'application Chifoumi

```
class ChifoumiVue : public QMainWindow
{
    Q_OBJECT

public:
    ChifoumiVue(ChifoumiModele *m,QWidget *parent = nullptr);
    ~ChifoumiVue();

    ///* Méthodes du Modéle
public:
    void miseAJour(ChifoumiModele::UnCoup,ChifoumiModele::UnCoup);
    //Met à jour les scores des joueurs et affiche dans les cases prévus les images de signes

//Getter
    ChifoumiModele* getModele();
//Setter
    void setModele(ChifoumiModele *m);
//Slots
public slots:
    void choixPapier();
    //Procédure qui active les différentes fonctions lorsque le joueur choisi la feuille
    void choixPierre();
    //Procédure qui active les différentes fonctions lorsque le joueur choisi la pierre
    void choixCiseau();
    //Procédure qui active les différentes fonctions lorsque le joueur choisi la ciseau
    void creerNvllePartie();
    //Procédure qui initialise les scores et les coups du joueur et de la machine, donne accès
aux boutons de figures

private:
    ChifoumiModele *_leModele;       // pteur vers le modèle
    Ui::ChifoumiVue *ui;
};
```

chifoumivue.cpp : corps de la vue de l'application Chifoumi

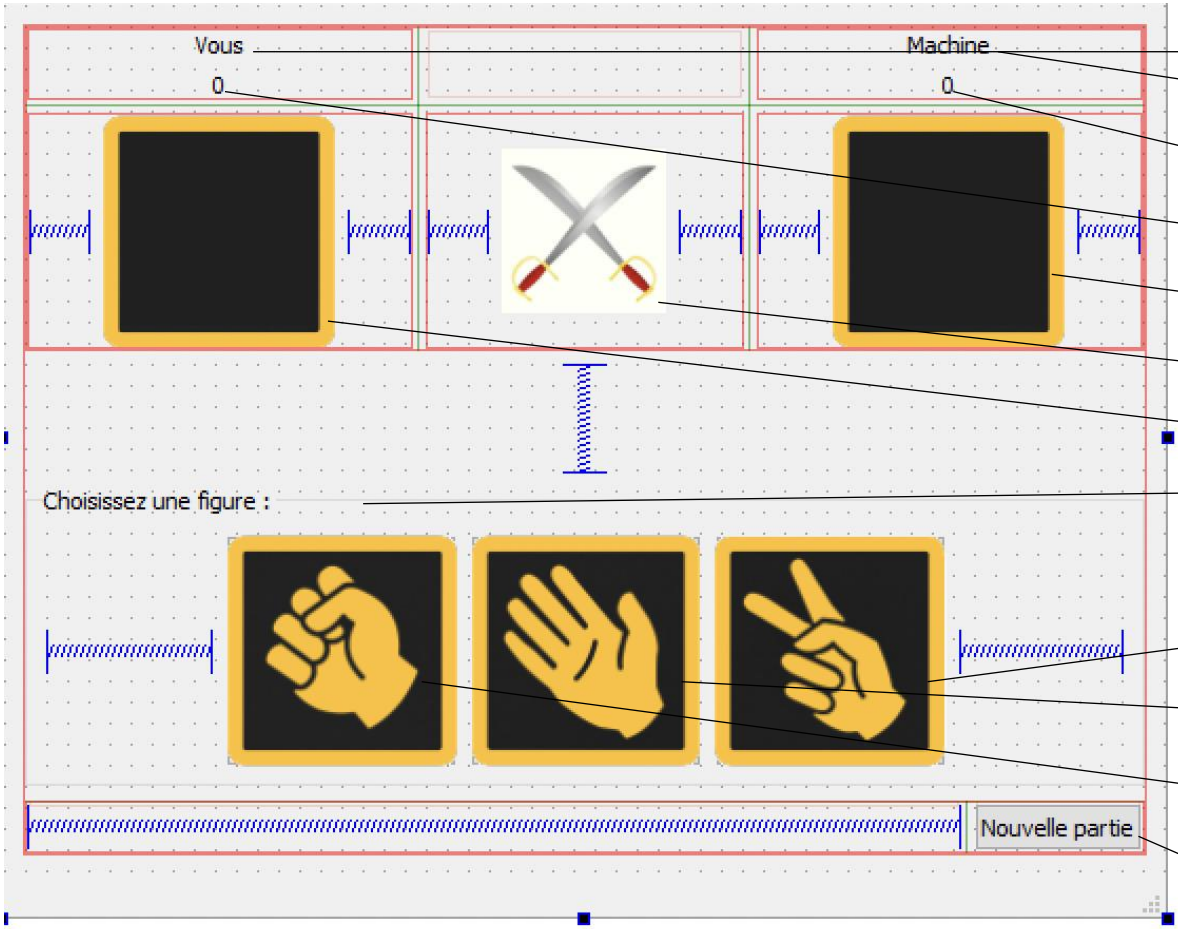
chifoumi.pro : fichier de construction du projet avec les chemins d'inclusion et l'arborescence des fichiers

chifoumievue.ui : fichier répertoriant tous les éléments d'interfaces et leurs dispositions fait avec QDesigner

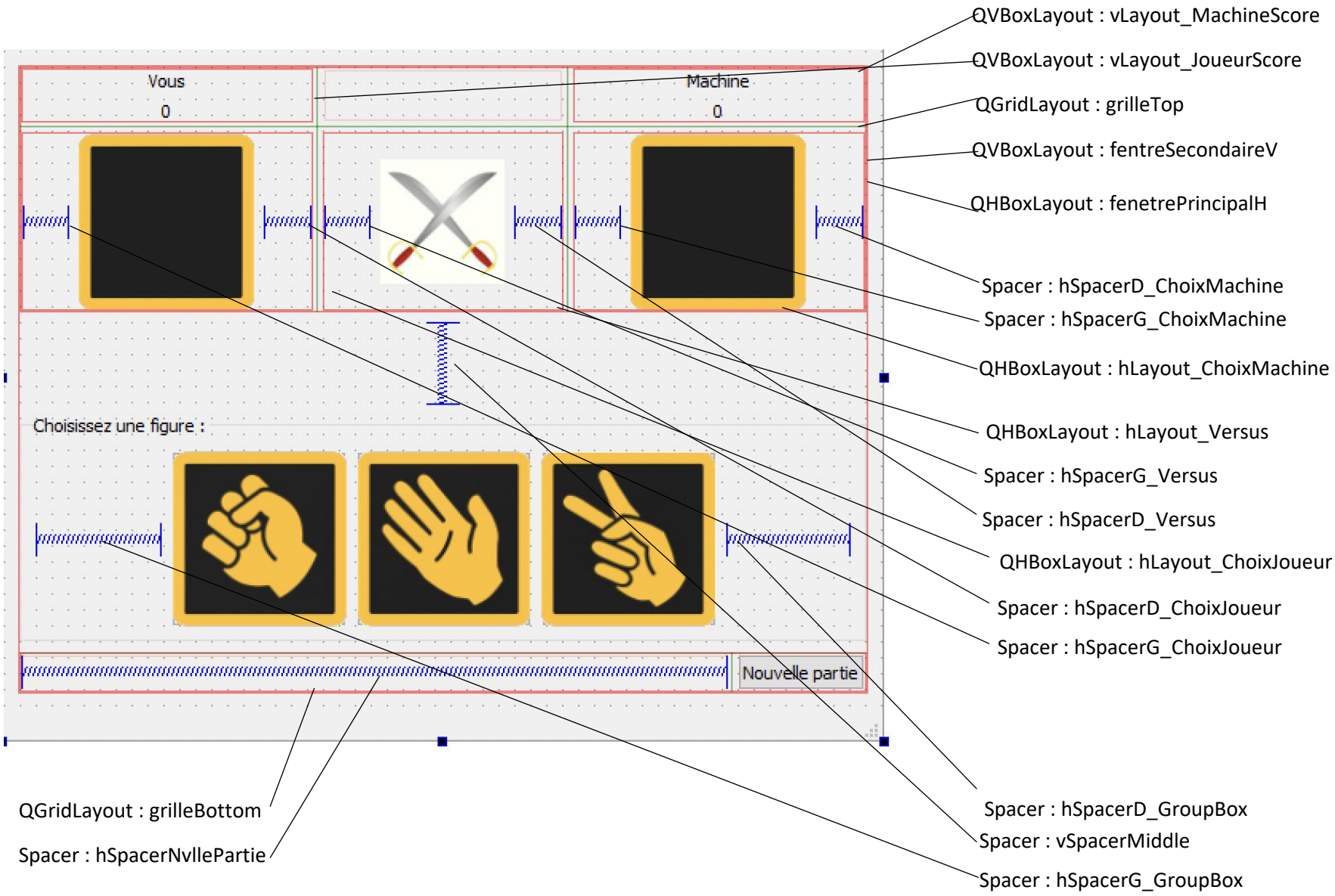
main.cpp : fichier source contenant la boucle principale d'exécution de l'application et secondaire d'attente des messages

ressourcesChifoumi.qrc : fichier qui répertorie toutes les sources de l'application et notamment des images

3-Element d’interfaces :

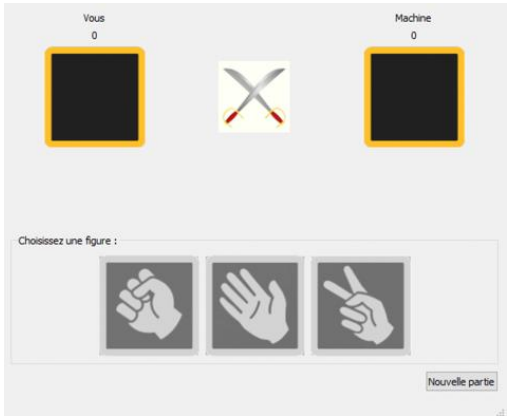


Label : lJoueur
Label : lMachine
Label : lScoreMachine
Label : lScoreJoueur
Label : lChoixJoueur
Label : lVersus
Label : lChoixMachine
GroupBox : groupBox
QPushButon : boutonCiseau
QPushButon : boutonPapier
QPushButon : boutonPierre
QPushButon : boutonNvllePartie

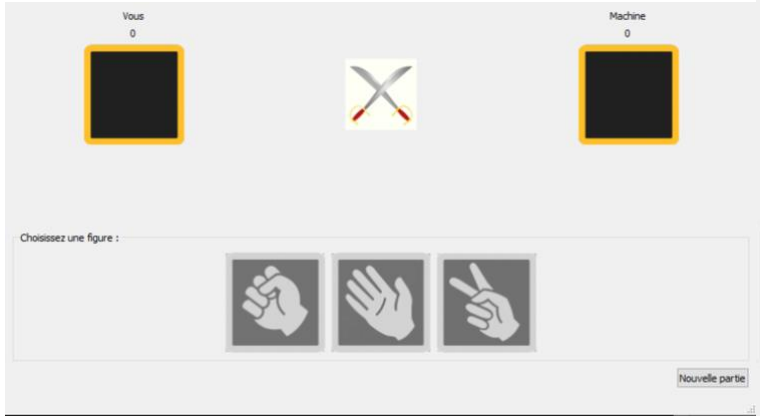

Vous
0
Machine
0
Choisissez une figure :
Nouvelle partie
QVBoxLayout : vLayout_MachineScore
QVBoxLayout : vLayout_JoueurScore
QGridLayout : grilleTop
QVBoxLayout : fentreSecondaireV
QHBoxLayout : fenetrePrincipalH
Spacer : hSpacerD_ChoixMachine
Spacer : hSpacerG_ChoixMachine
QHBoxLayout : hLayout_ChoixMachine
QHBoxLayout : hLayout_Versus
Spacer : hSpacerG_Versus
Spacer : hSpacerD_Versus
QHBoxLayout : hLayout_ChoixJoueur
Spacer : hSpacerD_ChoixJoueur
Spacer : hSpacerG_ChoixJoueur
QGridLayout : grilleBottom
Spacer : hSpacerNvllePartie
Spacer : hSpacerD_GroupBox
Spacer : vSpacerMiddle
Spacer : hSpacerG_GroupBox

4-Test de l’application Chifoumi :

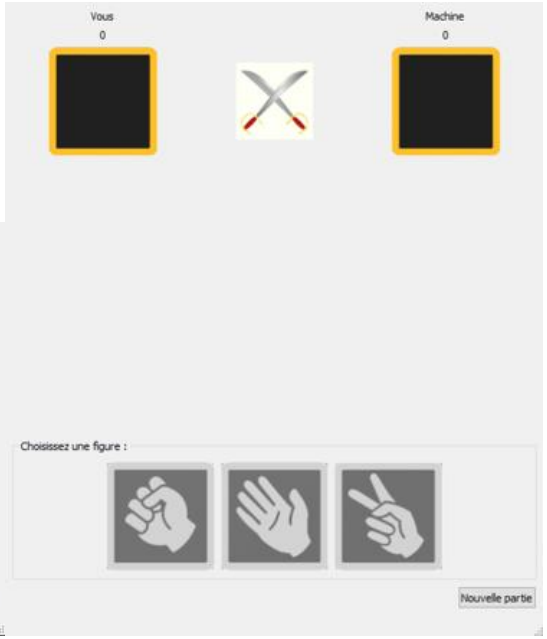
Tests de redimensionnement :



Taille normale



Étirement horizontal



Étirement vertical

Tests des activités :

Test	Résultat attendu	Validation
Activité 1	Système d'information : Le jeu est initialisé avec les variables coups et scores du joueur et de la machine prenant les valeurs de l'état initial Interface : Les boutons de signes sont accessibles pour le choix de l'utilisateur Les scores et noms des joueurs sont colorés en bleu pour indiquer que le jeu est en cours	Ok
Activité 2	Système d'Information : Le dernier coup du jour est mis à jour en fonction du coup joué. La machine joue (tirage aléatoire) et le dernier coup de la machine est mis à jour en conséquence. La machine détermine le gagnant et met à jour en conséquence les scores. Interface : Le signe choisi par le joueur s'affiche dans la case d'affichage prévu, de même pour celui de la machine Les scores sont mis à jour en fonction du gagnant	Ok
Activité 2.1 (Pierre)	//	Ok
Activité 2.2 (Papier)	//	Ok

Activité 2.3 (Ciseau)	//	Ok
Activité 3	Système d'Information : Les scores Joueur/Machine sont mis à 0. Les derniers coups Joueur/Machine joués sont mis à rien. Interface : Les zones d'affichage sont mises en cohérences avec les propriétés précédentes.	Ok

Version 2 :

1-Liste des fichiers de cette version :

chifoumiModele.h : interface du modèle de l'application Chifoumi
Rôle : Centré sur les informations et les actions métiers.
Déconnecté du dialogue avec l'utilisateur.
Indépendant des autres modules de l'application : ne connaît ni la vue ni la présentation.

```
class ChifoumiModele : public QObject
{
    Q_OBJECT
public:
    enum UnCoup {pierre, papier, ciseau, rien };
//    explicit ChifoumiModele(QObject *parent = nullptr);

///* ---- PARTIE MODELE ---------------------------

    ///* Méthodes du Modéle
public:
    ChifoumiModele();

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

    // Setters
public slots:
    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 coupMachine avec la valeur
           du paramètre p_score */

    // Autres modificateurs
     void majScores(char p_gagnant);
        /* Mise à jour des scores en fonction des règles de gestion actuelles :
            - 1 point pour le gagnant lorsqu'il y a un gagnant
            - 0 point en cas de match nul
        */
    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
```

chifoumiModele.cpp : corps du modèle de l'application Chifoumi

chifoumiPresentation.h : interface de la présentation de l'application Chifoumi

Rôle : Réagir aux événements de l'utilisateur, en modifiant les informations du modèle si nécessaire, puis en répercutant les effets sur la vue.
Indépendante des classes graphiques utilisées.
Les classes de la présentation font le lien entre les classes du modèle et les classes de la vue.

```
class ChifoumiPresentation : public QObject
{
    Q_OBJECT


public:
    //Type enuméré correspondant à l'état du jeu
    enum UnEtatJeu{etatInitial, jeuEnCours};
    //Constructeur
    explicit ChifoumiPresentation(ChifoumiModele *m,QObject *parent = nullptr);

    //Getter
    ChifoumiModele* getModele();
    ChifoumiVue* getVue();
    UnEtatJeu getEtatJeu();

    //Setter
    void setModele(ChifoumiModele *m);
    void setVue(ChifoumiVue *v);
    void setEtatJeu(UnEtatJeu e);

public slots :
    //Méthodes
    void choixPapier();
    //Procédure qui met en œuvre le diagramme états-transition lorsque le joueur
choisit le papier
    void choixPierre();
    //Procédure qui active les différentes fonctions lorsque le joueur choisi la
pierre
    void choixCiseau();
    //Procédure qui active les différentes fonctions lorsque le joueur choisi le
ciseau
    void creerNvllePartie();
    //Procédure qui initialise les scores et les coups du joueur et de la machine,
donne
    // accès aux boutons de figures

private :
    //Attribut de la présentation
    ChifoumiModele *_leModele;       // pteur vers le modèle
    ChifoumiVue *_laVue;             // pteur vers la vue
    UnEtatJeu _etatJeu;

};
```

chifoumiPresentation.cpp : corps de la présentation de l'application Chifoumi

chifoumiVue.h : interface de la vue de l'application Chifoumi
Rôle : C'est le code qui présente à l'utilisateur les informations du modèle, et qui interagit avec l'utilisateur.
Ne concerne que la gestion graphique.
Dépendante de la bibliothèque utilisée.

Indépendante du reste de l'application, en particulier, ne connaît rien de la logique de l'application.

```
class ChifoumiVue : public QMainWindow
{
    Q_OBJECT
public :
    //Constructeur
    explicit ChifoumiVue(ChifoumiPresentation *p, QWidget *parent = nullptr);
    ~ChifoumiVue();

    // getter et setter de la propriété pointant sur la Présentation
    ChifoumiPresentation* getPresentation();
    void setPresentation(ChifoumiPresentation *p);

/* Ici, toutes les méthodes qui correspondent à des ORDRES donnés par la présentation à
   l'interface
*/
    void miseAJour(ChifoumiPresentation::UnEtatJeu etat);
    // Met à jour les éléments d'interface AUTRES QUE les scores et derniers coups
    // joués,
    // en fonction de l'état du jeu.

    // ordres de mise à jour des scores et derniers coups joués
    void afficherScoreJoueur(QString scoreJ) ;
     // affiche la valeur du paramètre scoreJ dans la zone d'affichage du joueur
    void afficherScoreMachine(QString scoreM) ;
     // affiche la valeur du paramètre scoreM dans la zone d'affichage de la machine
    void afficherCoupJoueur(ChifoumiModele::UnCoup coupJ) ;
     // affiche la valeur du paramètre coupJ dans la zone d'affichage du joueur
    void afficherCoupMachine(ChifoumiModele::UnCoup coupM) ;
     // affiche la valeur du paramètre coupM dans la zone d'affichage de la machine

private:
    //Attribut de la vue
    Ui::ChifoumiVue *ui;
    ChifoumiPresentation* _laPresentation; //poiteur vers la présentation
};
```

chifoumiVue.cpp : corps de la vue de l'application Chifoumi

chifoumi.pro : fichier de construction du projet avec les chemins d'inclusion et l'arborescence des fichiers

chifoumieVue.ui : fichier répertoriant tous les éléments d'interfaces et leurs dispositions fait avec QDesigner

main.cpp : fichier source contenant la boucle principale d'exécution de l'application et secondaire d'attente des messages

Rôle : On retrouve les actions suivantes :

- -Création vue
- -Création modèle
- -Création présentation (initialisation membres privés _leModele et _laVue)
- -Il ordonne à la vue de s'afficher en cohérence avec l'état initial du modèle

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    // créer le modèle
    ChifoumiModele *m = new ChifoumiModele();
    // créer la présentation et lui associer le modèle
    ChifoumiPresentation *p = new ChifoumiPresentation(m);
    // créer la vue
```

```
    ChifoumiVue v(p);
    // associer la vue à la présentation
    p->setVue(&v);
    // initialiser la vue en conformité avec l'état initial du modèle
    p->getVue()->miseAJour(p->getEtatJeu());
    // afficher la vue et démarrer la boucle d'attente des messages
    v.show();
    return a.exec();
}
```

ressourcesChifoumi.qrc : fichier qui répertorie toutes les sources de l'application et notamment des images

2-Présentation du modèle MVP :

MVP – Une architecture basée sur 3 composants

MVP est un patron destiné aux applications à interface graphique qui préconise l'organisation du code en séparant les données, les traitements et l'interface.

Modèle Vue Présentation :

Donnée→ Le Modèle : C'est le code qui structure les informations gérées par l'application, ainsi que les opérations 'métier' de l'application. C'est le système d'information sous-jacent à l'application.
Interface→ La Vue : C'est le code qui présente à l'utilisateur les informations du modèle, et qui interagit avec l'utilisateur.

Traitements→ La Présentation : C'est le code qui valide les entrées de l'utilisateur, et qui détermine ce qu'elles signifient pour le modèle. Tous les échanges entre la Vue et le Modèle passent par la Présentation.

3-Tests réalisés :

Présentation :

Test	Résultat attendu	Validation
ChifoumiModele* getModele();	Système d'information : Retourne la donnée membre correspondant au pointeur du modèle	Ok
ChifoumiVue* getVue();	Système d'information : Retourne la donnée membre correspondant au pointeur de la vue	Ok
UnEtatJeu getEtatJeu();	Système d'information : Retourne l'état du jeu	Ok
void setModele(ChifoumiModele	Système d'information :	Ok

*m);	Affecte la valeur du paramètre dans l'attribut qui pointe vers le modèle	
void setVue(ChifoumiVue *v);	Système d'information : Affecte la valeur du paramètre dans l'attribut qui pointe vers la vue	Ok
void setEtatJeu(UnEtatJeu e);	Système d'information : Affecte la valeur du paramètre comme état du jeu	Ok
Void choixSigne() ;	Système d'information : Appelle de la méthodes CoupJoueur dans le modèle pour affecter un coup au joueur Appelle des méthode GenererCoup et setCoupMachine dans le modèle pour affecter un coup à la machine Appelle de la méthode DétermineGagnant dans le modèle Interface : Appelle des méthodes de la vue pour afficher les coups et les scores	Ok
Void choixPapier	//juste le signe qui change	Ok
void choixPierre();	// juste le signe qui change	Ok
void choixCiseau();	// juste le signe qui change	Ok
creerNvllePartie();	Si état est etatInitial Système d'information : Affecte l'état du jeu à « jeuEnCour » Appelle des méthodes pour initialiser les scores et les coups dans le modèle Interface : Appelle de la méthode pour mettre à jour la vue Si état est jeuEnCour Système d'information : Appelle des méthodes pour initialiser les scores et les coups dans le modèle Interface : Appelle de la méthode pour mettre à jour la vue	Ok

Vue :

Test	Résultat attendu	Validation
ChifoumiPresentation* getPresentation();	Système d'information : Retourne la donnée	Ok

	membre correspondant au pointeur de la présentation	
void setPresentation(ChifoumiPresentation *p);	Système d'information : Affecte la valeur du paramètre dans l'attribut qui pointe vers la présentation	Ok
void miseAJour(ChifoumiPresentation::UnEtatJeu etat);	Si état est etatInitial Interface : Désactivation des boutons de signes Et focus sur le boutons nouvelle partie Si état est jeuEnCour Interface : Affichage des scores du joueur et de la machine sur 0 Affichage des coups du joueur et de la machine sur rien Activation des boutons de signes	Ok
void afficherScoreJoueur(unsigned int scoreJ) ;	Interface : Affiche le score du joueur	Ok
void afficherScoreMachine(unsigned int scoreM) ;	Interface : Affiche le score de la machine	Ok
void afficherCoupJoueur(ChifoumiModele::UnCoup coupJ) ;	Interface : Affiche le coup du joueur passé en paramètre	Ok
void afficherCoupMachine(ChifoumiModele::UnCoup coupM) ;	Interface : Affiche le coup de la machine passé en paramètre	Ok

Version 3 :

1-Liste des fichiers de cette version :

chifoumiModele.h : cf Version 2

chifoumiModele.cpp : cf Version 2

chifoumiPresentation.h : cf Version 2

chifoumiPresentation.cpp : cf Version 2

chifoumiVue.h : interface de la vue de l'application Chifoumi

```
class ChifoumiVue : public QMainWindow
{
    Q_OBJECT
public :
    //Constructeur
    explicit ChifoumiVue(ChifoumiPresentation *p, QWidget *parent = nullptr);
    ~ChifoumiVue();

    // getter et setter de la propriété pointant sur la Présentation
    ChifoumiPresentation* getPresentation();
    void setPresentation(ChifoumiPresentation *p);

/* Ici, toutes les méthodes qui correspondent à des ORDRES donnés par la présentation à
   l'interface
*/
    void miseAJour(ChifoumiPresentation::UnEtatJeu etat);
    // Met à jour les éléments d'interface AUTRES QUE les scores et derniers coups
    // joués,
    // en fonction de l'état du jeu.

    // ordres de mise à jour des scores et derniers coups joués
    void afficherScoreJoueur(QString scoreJ) ;
     // affiche la valeur du paramètre scoreJ dans la zone d'affichage du joueur
    void afficherScoreMachine(QString scoreM) ;
     // affiche la valeur du paramètre scoreM dans la zone d'affichage de la machine
    void afficherCoupJoueur(ChifoumiModele::UnCoup coupJ) ;
     // affiche la valeur du paramètre coupJ dans la zone d'affichage du joueur
    void afficherCoupMachine(ChifoumiModele::UnCoup coupM) ;
     // affiche la valeur du paramètre coupM dans la zone d'affichage de la machine
public slots:
    void aProposDe();
    //Procédure qui affiche une message box contenant des informations sur l'application
(version,date,auteurs)

private:
    //Attribut de la vue
    Ui::ChifoumiVue *ui;
    ChifoumiPresentation* _laPresentation; //poiteur vers la présentation
};
```

chifoumiVue.cpp : corps de la vue de l'application Chifoumi

chifoumi.pro : cf Version 2

main.cpp : cf version 2

chifoumieVue.ui : fichier répertoriant tous les éléments d'interfaces et leurs dispositions fait avec QDesigner

ressourcesChifoumi.qrc : fichier qui répertorie toutes les sources de l'application et notamment des images

2-Modifications apportées :

ChifoumiVue.ui :

-Rajout dans le menu bar :
- -menu fichier contenant actionQuitter (CTRL+Q)
- -menu aide contenant actionApropos (F1)

ChifoumiVue.cpp :

-Connexion entre les boutons d'action de la barre de menu et leurs méthodes

```
connect(ui->actionApropos, SIGNAL(triggered()), this, SLOT(aProposDe()));
connect(ui->actionQuitter, SIGNAL(triggered()), this, SLOT(close()));
```

-Rajout de la méthode aProposDe() qui lance une message box contenant des informations sur l'application

```
void ChifoumiVue::aProposDe()
{
    //qDebug() << "A propos de cliqué" << Qt::endl;
    QMessageBox::information(this, "A propos de cette application", "Version : 3 \n
Date de création : 29/04/2022 \n Auteurs : Titouan Cocheril, Ivan Salle, Kepa
Eyherabide ");
}
```

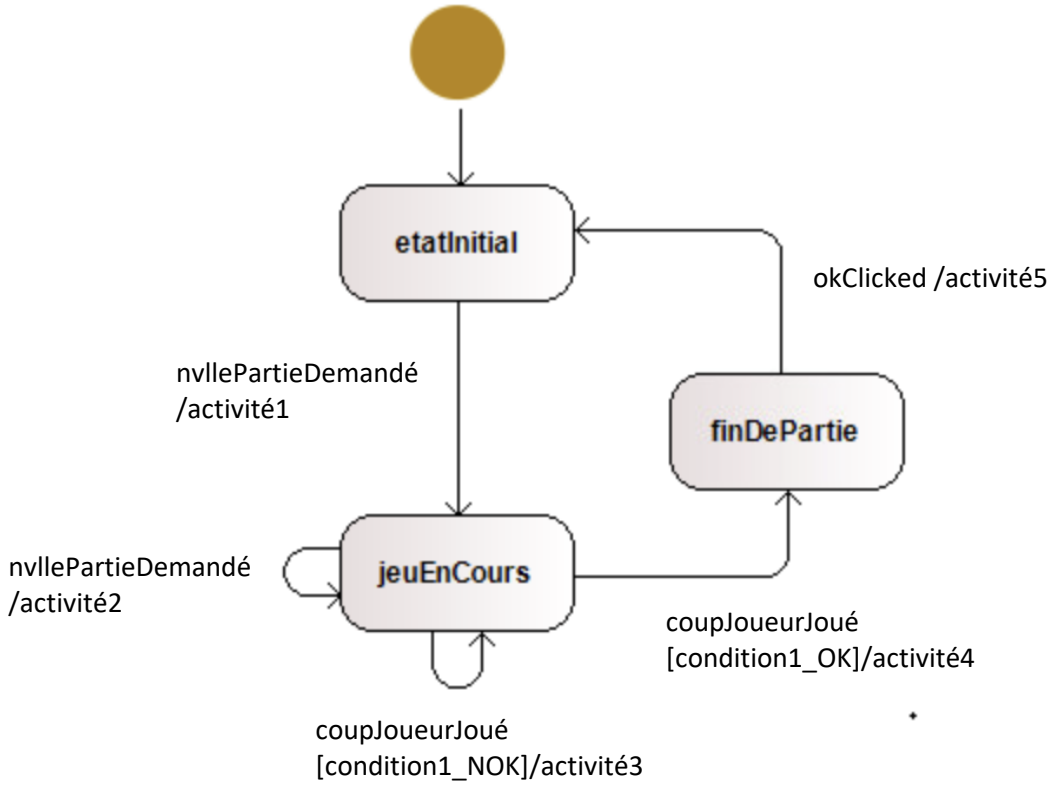
3-Tests réalisés :

Test	Résultat attendu	Validation
aProposDe()	Interface : Affiche une Message Box contenant des informations sur les auteurs, la date de création et la version	Ok
Close()	Interface : Ferme la fenêtre	Ok

Version 4 :

1-Classe Chifoumi : Diagramme états-transitions :

(a) Diagramme états-transitions -actions du jeu



Condition1 : score du joueur ou score machine = 5

Figure 5 : Diagramme états-transitions

(b) Dictionnaires des états, événements et Actions

Dictionnaire des états du jeu

nomEtat	*Signification*
`etatInitial`	Le programme est lancé, les variables scores sont à 0 et la partie est prête à être joué. Les signes sont initialisés et l'affichage des boutons sont actifs
`jeuEnCours`	Le jeu est lancé, le joueur peut successivement avec la machine choisir son signe et les variables et fonctions seront misent à jour selon le résultat.
`finPartie`	La partie est terminée, une message box affiche les informations sur la partie.

Tableau 7 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

nomEvénement	*Signification*
`nvllePartieDemandée`	Le joueur sélectionne nouvelle partie
`coupJoueurJoué`	Le joueur choisi un signe parmi les trois proposés (pierre, feuille, ciseau)
`okCliqued`	Le joueur valide le message de fin de partie

Tableau 8 : Evénements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

`Activité 1 :`	Système d'information : Le jeu est initialisé avec les variables coups et scores du joueur et de la machine prenant les valeurs de l'état initial Interface : Les boutons de signes sont accessibles pour le choix de l'utilisateur Les scores et nom des joueurs sont colorés en bleu pour indiquer que le jeu est en cours
`Activité 2 :`	Système d'Information : Les scores Joueur/Machine sont mis à 0. Les derniers coups Joueur/Machine joués sont mis à rien. Interface : Les zones d'affichage sont mises en cohérences avec les propriétés précédentes.

Activité 3 :	Système d'Information : Le dernier coup du jour est mis à jour en fonction du coup joué. La machine joue (tirage aléatoire) et le dernier coup de la machine est mis à jour en conséquence. La machine détermine le gagnant et met à jour en conséquence les scores. Interface : Le signe choisi par le joueur s'affiche dans la case d'affichage prévu, de même pour celui de la machine Les scores sont mis à jour en fonction du gagnant
Activité 4 :	Système d'Information : L'état du jeu est mis sur l'état finPartie Demande une mise à jour de l'interface Interface : Une boite de message s'ouvre pour indiquer la fin de partie et son gagnant
Activité 5 :	Système d'Information : Initialisation des scores et des coups dans le modèle L'état du jeu est placé sur l'état initial Interface : On met à jour l'interface en fonction de l'état Les zones d'affichage sont mises en cohérences avec les propriétés précédentes.

Tableau 9 : 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

Événement → *nomEtatJeu*	coupJoueurJoué	nvllePartieDemandée	OkClicked
etatinitial	--	jeuEnCours/activité1	--

jeuEnCours	[condition1_NOK] jeuEnCours / activité3 --- [condition1_OK] FinPartie / activité4	jeuEnCours/activité2	--
finPartie	--	--	etatInitial/activité5

Tableau 10 : Matrice d'états-transitions du jeu chifoumi

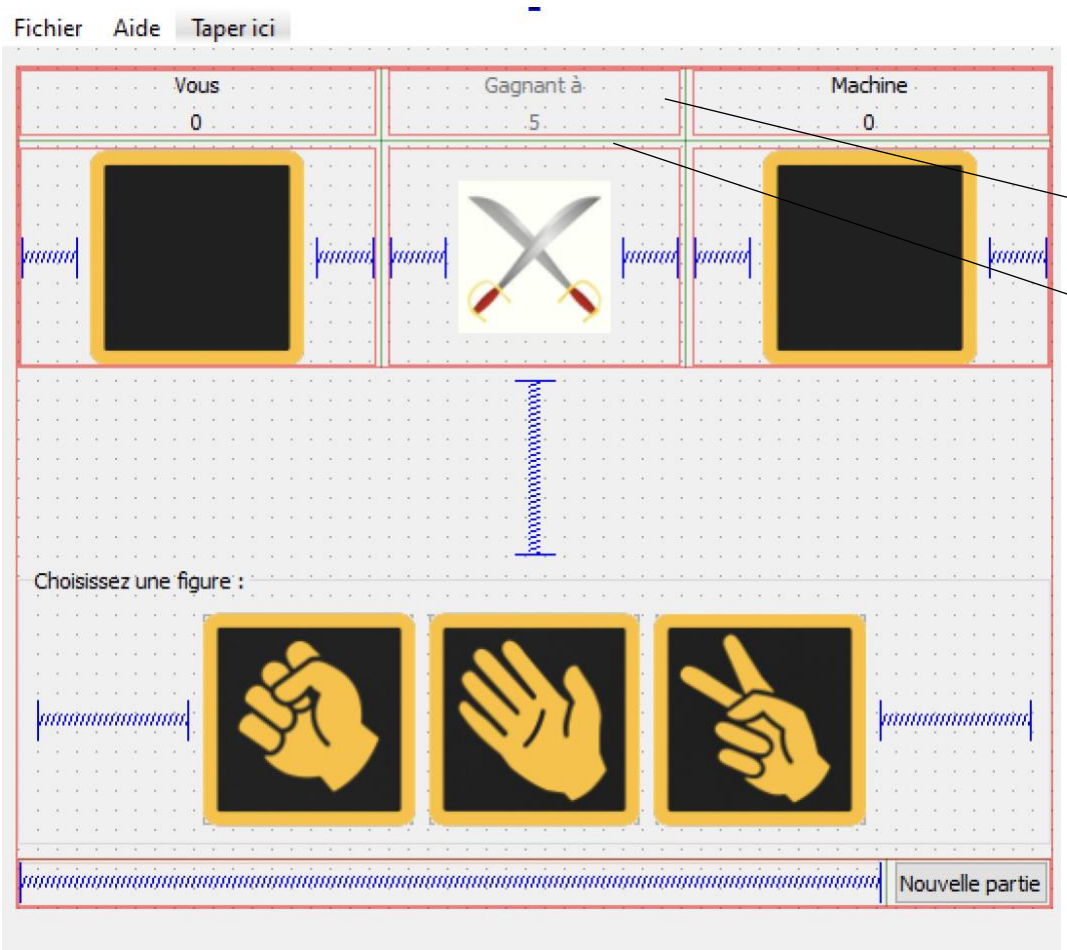
Table `T_EtatsEvenementsJeu` avec **les éléments d'interface** prenant en charge les événements

	boutonPierre	boutonCiseau	boutonPierre	boutonNvlle Partie	
Événement → *nomEtatJeu*	coupJoueurJoué			NvllePartie Demandée	okClicked
etatinitial	--			jeuEnCours/ activité1	--
jeuEnCours	[condition1_NOK] jeuEnCours / activité3 --- [condition1_OK] FinPartie / activité4			jeuEnCours/ activité2	--

finPartie	--	--	etatInitial /activité5

Tableau 6 : Matrice d'états-transitions du jeu chifoumi AVEC éléments d'interface

2-Nouveaux éléments d'interface :



Label : lGagnant_a

Label : lScoreGagnant

3-Liste des fichiers sources :

chifoumiModele.h : cf Version 2

chifoumiModele.cpp : cf Version 2

chifoumiPresentation.h : interface de la présentation de l'application Chifoumi

```
class ChifoumiPresentation : public QObject
{
    Q_OBJECT
public:
    //Type enuméré correspondant à l'état du jeu
    enum UnEtatJeu{etatInitial, jeuEnCours, finPartie};
    //Constructeur
    explicit ChifoumiPresentation(ChifoumiModele *m,QObject *parent = nullptr);

    //Getter
    ChifoumiModele* getModele();
    /* retourne le pointeur du modèle */
    ChifoumiVue* getVue();
    /* retourne le pointeur de la vue */
    UnEtatJeu getEtatJeu();
    /* retourne l'état du jeu */

    //Setter
    void setModele(ChifoumiModele *m);
    /* initialise le pointeur du modèle avec la valeur
       du paramètre m */
    void setVue(ChifoumiVue *v);
    /* initialise le pointeur de la vue avec la valeur
       du paramètre v */
    void setEtatJeu(UnEtatJeu e);
    /* initialise l'état du jeu avec la valeur
       du paramètre e */
    //Méthodes
    QString getScoreMax();
    //Fonction qui renvoie le score le plus élevé entre les deux joueurs
    QString recupererNomGagnant();
    //Procédure qui convertie le char retourner par la méthode determineGagnant en QString
    bool maxScoreAtteind();
    //Fonction qui retourne vrai si le score maximal de la partie est atteind sinon
retourne faux
    void okClicked();
    //Procédure qui initialise les coups et scores, place l'état du jeu sur l'état initial
et met
    //à jour la vue lorsque l'utilisateur clique sur le bouton OK de la message box

public slots :
    //Méthodes SLOTS
    void choixPapier();
    //Procédure qui met en œuvre le diagramme états-transition lorsque le joueur choisit le
papier
    void choixPierre();
    //Procédure qui active les différentes fonctions lorsque le joueur choisi la pierre
    void choixCiseau();
    //Procédure qui active les différentes fonctions lorsque le joueur choisi le ciseau
    void creerNvllePartie();
    //Procédure qui initialise les scores et les coups du joueur et de la machine, donne
    // accès aux boutons de figures

private :
    //Attribut de la présentation
    ChifoumiModele *_leModele;       // pteur vers le modèle
    ChifoumiVue *_laVue;             // pteur vers la vue
    UnEtatJeu _etatJeu;         // données membre de l'état du jeu

};
```

chifoumiPresentation.cpp : cf Version 2

chifoumiVue.h :

```
class ChifoumiVue : public QMainWindow
{
    Q_OBJECT
public :
    //Constructeur
    explicit ChifoumiVue(ChifoumiPresentation *p, QWidget *parent = nullptr);
    ~ChifoumiVue();


    // getter et setter de la propriété pointant sur la Présentation
    ChifoumiPresentation* getPresentation();
    void setPresentation(ChifoumiPresentation *p);

/* Ici, toutes les méthodes qui correspondent à des ORDRES donnés par la
présentation à
   l'interface
*/
    void miseAJour(ChifoumiPresentation::UnEtatJeu etat);
    // Met à jour les éléments d'interface AUTRES QUE les scores et derniers coups
    // joués,
    // en fonction de l'état du jeu.

    // ordres de mise à jour des scores et derniers coups joués
    void afficherScoreJoueur(QString scoreJ) ;
     // affiche la valeur du paramètre scoreJ dans la zone d'affichage du joueur
    void afficherScoreMachine(QString scoreM) ;
     // affiche la valeur du paramètre scoreM dans la zone d'affichage de la
machine
    void afficherCoupJoueur(ChifoumiModele::UnCoup coupJ) ;
     // affiche la valeur du paramètre coupJ dans la zone d'affichage du joueur
    void afficherCoupMachine(ChifoumiModele::UnCoup coupM) ;
     // affiche la valeur du paramètre coupM dans la zone d'affichage de la machine

    void notifierGagnant(QString, QString);
    //Procédure qui affiche dans une Message Box information le gagnant de la
partie et son score

public slots:
    //Méthodes SLOTS
    void aProposDe();
    //Procédure qui affiche une Message Box contenant des informations sur
l'application (version,date,auteurs)

private:
    //Attribut de la vue
    Ui::ChifoumiVue *ui;
    ChifoumiPresentation* _laPresentation; //poiteur vers la présentation
};
```

chifoumiVue.cpp : cf Modification apportées

chifoumi.pro : cf Version 2

chifoumieVue.ui : cf Modification apportées

main.cpp : cf Version 2

ressourcesChifoumi.qrc : cf Version 2

4-Modifications apportés :

ChifoumiVue.ui :

-Rajout de deux labels :
- -label lGagnant_a
- -label lScoreGagnant

ChifoumiVue.cpp :

-Prise en compte dans la méthode mise à jour de l'état finPartie

-Appel de l'affichage fin partie et notifie la présentation quand bouton ok cliqué

```
void ChifoumiVue::miseAJour(ChifoumiPresentation::UnEtatJeu etat)
{
    switch(etat){
    case ChifoumiPresentation::etatInitial:
        {
        //Désactivation des boutons figures
        ui->boutonPierre->setEnabled(false);
        ui->boutonPapier->setEnabled(false);
        ui->boutonCiseau->setEnabled(false);
        //Mise en couleur des labels de noms et scores
        ui->lScoreJoueur->setStyleSheet("color: black;");
        ui->lScoreMachine->setStyleSheet("color: black;");
        ui->label_Vous->setStyleSheet("color: black;");
        ui->lMachine->setStyleSheet("color: black;");
        ui->boutonNvllePartie->setFocus();  //Placement du focus sur le bouton
nouvelle partie
    }break;
    case ChifoumiPresentation::jeuEnCours:
    {
        //Initialisation des scores et des coups
        ui->lScoreJoueur->setText("0");
        ui->lScoreMachine->setText("0");
        ui->lChoixJoueur->setPixmap(QPixmap(":/chifoumi/images/rien.gif"));
        ui->lChoixMachine->setPixmap(QPixmap(":/chifoumi/images/rien.gif"));
        //Activation des boutons figures
        ui->boutonPierre->setEnabled(true);
        ui->boutonPapier->setEnabled(true);
        ui->boutonCiseau->setEnabled(true);
        //Mise en couleur des labels de noms et scores
        ui->lScoreJoueur->setStyleSheet("color: blue;");
        ui->lScoreMachine->setStyleSheet("color: blue;");
        ui->label_Vous->setStyleSheet("color: blue;");
        ui->lMachine->setStyleSheet("color: blue;");
        ui->boutonNvllePartie->setFocus();  //Placement du focus sur le bouton
nouvelle partie
    }break;
    case ChifoumiPresentation::finPartie:
    {
        //Appel de la fonction qui affiche la fin de partie dans une message box
        notifierGagnant(_laPresentation->recupererNomGagnant(),_laPresentation-
>getScoreMax());
       //Notifie la présentation que l'utilisateur a cliqué sur le bouton OK de la
Message Box
        _laPresentation->okClicked();
        }break;
    default :
            break;
    }
}
```

-Affiche la fin de partie dans une message box

```
void ChifoumiVue::notifierGagnant(QString nom,QString pts)
{
    //Affichage d'une box message indiquant le gagnant et son score
    QMessageBox::information(this, "Fin de partie", "Bravo à " + nom + ". Vous
gagnez la partie avec "+pts+" points !");
}
```

ChifoumiPresentation.cpp :

- Prise en compte dans les méthodes de l'état finPartie et actions associées

-Modification des méthodes de choixSigne (choixPierre (), choixPapier(), choixCiseau()) :

A chaque nouvelle manche, on vérifie si le joueur ou la machine n'ont pas atteint le score maximal et si c'est le cas on place l'état du jeu sur finPartie et on met à jour la vue

```
void ChifoumiPresentation::choixPierre()
{
    UnEtatJeu g = getEtatJeu();
    switch(g){
    //Etat initial
    case etatInitial:{} //Cas impossible
    break;
    //Etat jeu en cours
    case jeuEnCours:
    {
    _leModele->setCoupJoueur(ChifoumiModele::pierre);   //Place le coup du
joueur sur ciseau
    _leModele->setCoupMachine(_leModele->genererUnCoup());  //Génère un coup
pour la machine
    _leModele->majScores(_leModele->determinerGagnant());   //Détermine le
gagnant et on met à jours les scores
    _laVue->afficherCoupJoueur(_leModele->getCoupJoueur()); //Affiche le
coup du joueur
    _laVue->afficherCoupMachine(_leModele->getCoupMachine());   //Affiche le
coup de la machine
    _laVue->afficherScoreJoueur(QString::number(_leModele-
>getScoreJoueur()));  //Affiche le score du joueur
    _laVue->afficherScoreMachine(QString::number(_leModele-
>getScoreMachine()));    //Affiche le score de la machine
    if(maxScoreAtteind())   //Vérifie si un des joueurs à le score maximal
    {
        setEtatJeu(finPartie);  //Place l'état du jeu sur finPartie
        _laVue->miseAJour(getEtatJeu());    //Mise à jour de l'affichage
avec l'état du jeu
    }
    }break;
    case finPartie:{} //Cas impossible
    break;
    default :
            break;
    }
}
```

-Méthode qui retourne vrai si le score maximal est atteint sinon faux

```
bool ChifoumiPresentation::maxScoreAtteind()
{
    //Si le score maximal (5) est atteint retourne vrai sinon retourne faux
    if(_leModele->getScoreMachine()==5 || _leModele->getScoreJoueur()==5)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

-Méthode qui initialise les variables de scores et de coups, change l'état du jeu sur etatInitial, met à jour l'interface quand le bouton OK est cliqué

```
void ChifoumiPresentation::okClicked()
{
     UnEtatJeu g = ChifoumiPresentation::getEtatJeu();
     switch (g) {
     //Etat initial
     case etatInitial:
     {}break;//Cas impossible
     //Etat jeuEnCours
     case jeuEnCours:
     {}break;//Cas impossible
     //Etat finPartie
     case finPartie:
     {
         _leModele->initCoups();
         _leModele->initScores();
         setEtatJeu(etatInitial);
         _laVue->miseAJour(getEtatJeu());
         _laVue->afficherCoupJoueur(_leModele->getCoupJoueur());
         _laVue->afficherCoupMachine(_leModele->getCoupMachine());
         _laVue->afficherScoreJoueur(QString::number(_leModele-
>getScoreJoueur()));
         _laVue->afficherScoreMachine(QString::number(_leModele-
>getScoreMachine()));
     }break;
     }
}
```

-Méthode qui retourne le nom du gagnant sous forme de string en prenant le char de la méthode déterminéeGagnant() du modèle

```
QString ChifoumiPresentation::recupererNomGagnant()
{
    //Affectation du nom du gagnant en fonction du résultat de la partie
    char g = _leModele->determinerGagnant();
    QString gagnant;
    switch(g){
    //Cas ou le joueur gagne la partie
    case 'J':
        {
        gagnant="vous Joueur";
        }break;
        //Cas ou la machine gagne la partie
    case 'M':
        {
        gagnant="La Machine";
    }break;
    //Cas ou il n'y a pas de gagnant
    case 'N':{}break;   //Cas impossible
    }
    return gagnant;
}
```

-Méthode qui retourne le score du joueur ayant le score le plus élevé

```
QString ChifoumiPresentation::getScoreMax()
{
    //Si le joueur à un score supérieur
    if(_leModele->getScoreJoueur()>_leModele->getScoreMachine())
    {
      return QString::number(_leModele->getScoreJoueur());
    }
    //Si la machine à un score supérieur
    else if(_leModele->getScoreJoueur()<_leModele->getScoreMachine())
    {
      return  QString::number(_leModele->getScoreMachine());
    }
    //Si la machine à un score égale au score du joueur
    else{
        return 0; //Cas impossible
    }
}
```

5-Tests réalisés

Test	Résultat attendu	Validation
scoreJoueur=5	Système d'information : Met l'état sur finPartie Interface : Affiche la fin de partie et le joueur gagnant	Ok
scoreMachine=5	Système d'information : Met l'état sur finPartie Interface : Affiche la fin de partie et la machine gagnante	Ok
okClicked()	Système d'information : Initialise les variables score et coup Met l'état sur etatInital Interface : Mise à jour de l'interface en fonction de l'état et des variables	Ok