

## Saé 2.01 – Développement d'une application

### Chifoumi – Dossier d'Analyse et conception v2

## Sommaire :

### **Version Initiale :**

- 1-Compléments de spécifications externe
- 2-Diagramme des cas d'utilisations
- 3-Scénarios
- 4-Diagramme de classe UML

### **Version 0 :**

- 1-Liste des fichiers sources
- 2-Tests méthodes

### **Version 1 :**

- 1-Diagramme état transition
- 2-Liste des fichiers sources
- 3-Éléments d'interfaces
- 4-Test réalisés

### **Version 2 :**

- 1-Listes de fichiers sources
- 2-Présentation du modèle MVP
- 3-Test réalisés

## 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

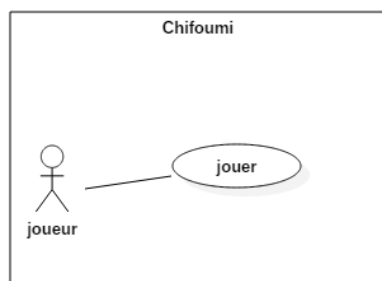


Figure 1 : Diagramme des Cas d'Utilisation du jeu Chifoumi

### 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.

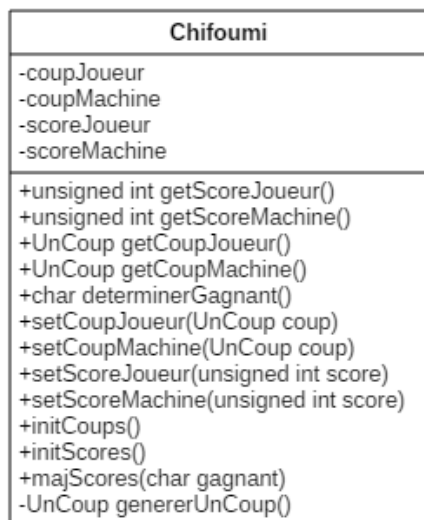


Figure 2 : Diagramme de Classes UML du jeu Chifoumi

(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	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 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 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);
        /* 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 de événements

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

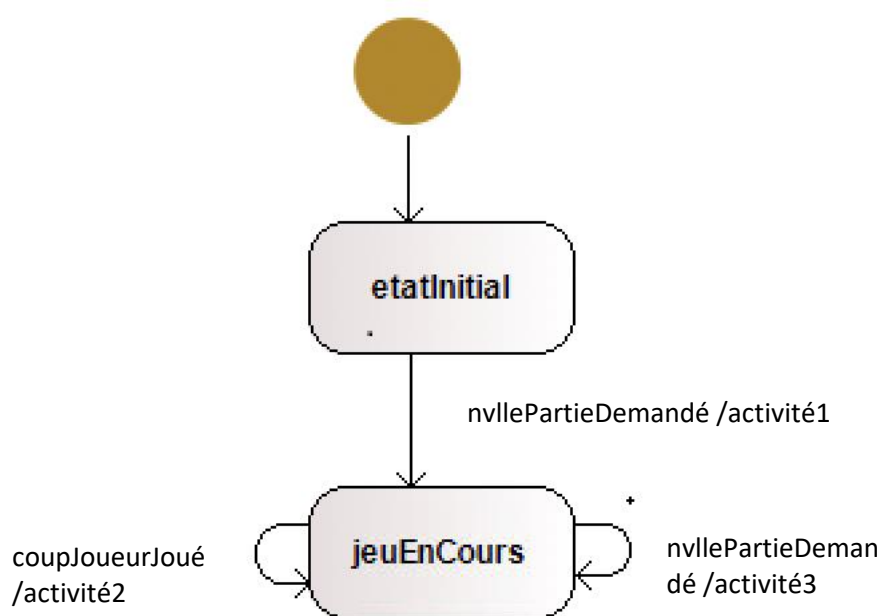


Figure 4 : Diagramme états-transitions



## (b) Dictionnaires des états, événements et Actions

### Dictionnaire des états du jeu

<i>nomEtat</i>	<i>Signification</i>
etatInitial	Le programme est lancé, les variables scores sont à 0 et les signes et la partie est prête à être jouée. Les signes sont initialisés et l’affichage des boutons aussi
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

<i>nomEvénement</i>	<i>Signification</i>
nvlllePartieDemandée	Le joueur demande à jouer une nouvelle partie : si on était dans l’état initial alors on passe dans l’état jeu en cours en mettant à disposition la possibilité pour le joueur de choisir un signe. Si on était dans l’état jeu en cours on reste dans cet état, les variables de coups et de scores sont remises respectivement à ‘rien’ et 0
coupJoueurJoué	Le joueur choisi un signe parmi les trois proposés (pierre, feuille, ciseau) et les variables de coups et de scores sont mis à jour en fonctions des choix du joueur et de la machine

Tableau 4 : Evénements faisant changer le jeu d’état

### Description des actions réalisées lors de la traversée des transitions

Activité 1 :	<u>Système d’information :</u> Le jeu est initialisé avec les variables coups et scores du joueur et de la machine prenant les valeurs de l’état initial <u>Interface :</u> Les boutons de signes sont accessibles pour le choix de l’utilisateur Les scores et nom des joueurs sont mis avec une couleur bleue pour indiquer que le jeu est en cours Le focus est sur le bouton ‘NvlllePartie’
Activité 2 :	<u>Système d’Information :</u> 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. <u>Interface :</u> 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 Le focus est sur le bouton ‘NvlllePartie’

Activité 3 :	<u>Système d'Information :</u> Les scores Joueur/Machine sont mis à 0. Les derniers coups Joueur/Machine joués sont mis à rien. <u>Interface :</u> Les zones d'affichage sont mises en cohérences avec les propriétés précédentes. Le focus est sur le bouton 'NvllePartie'
--------------	--

Tableau 5 : Actions à réaliser lors des changements d'état

(c) Préparation au codage :

**Table  $T_{\text{EtatsEvenementsJeu}}$**  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>	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
<i>Événement</i> → <i>nomEtatJeu</i>	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 Modèle */
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 Modèle
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

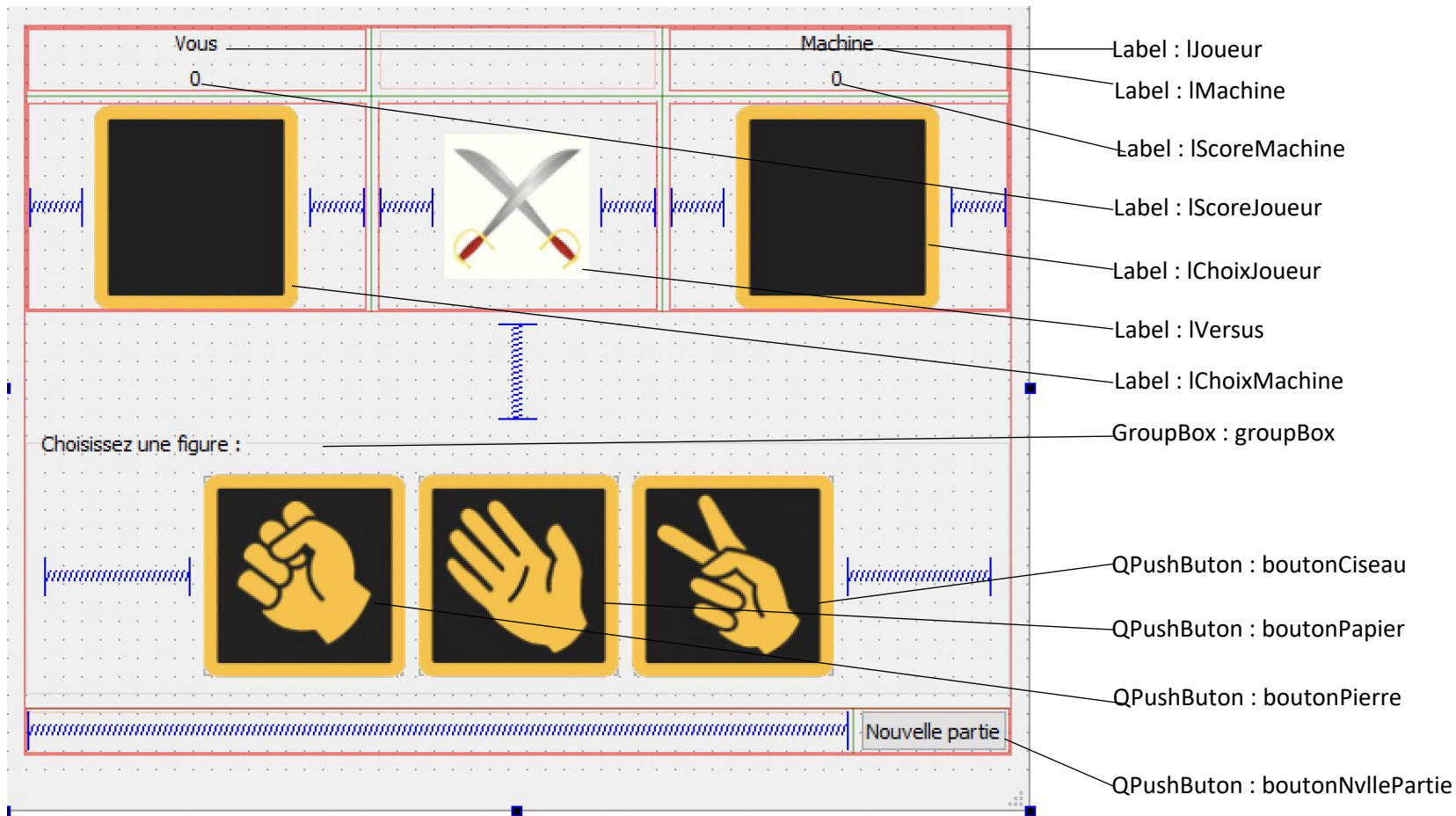
chifoumi.pro : fichier de construction du projet avec les chemins d'inclusion et l'arborescence des fichiers

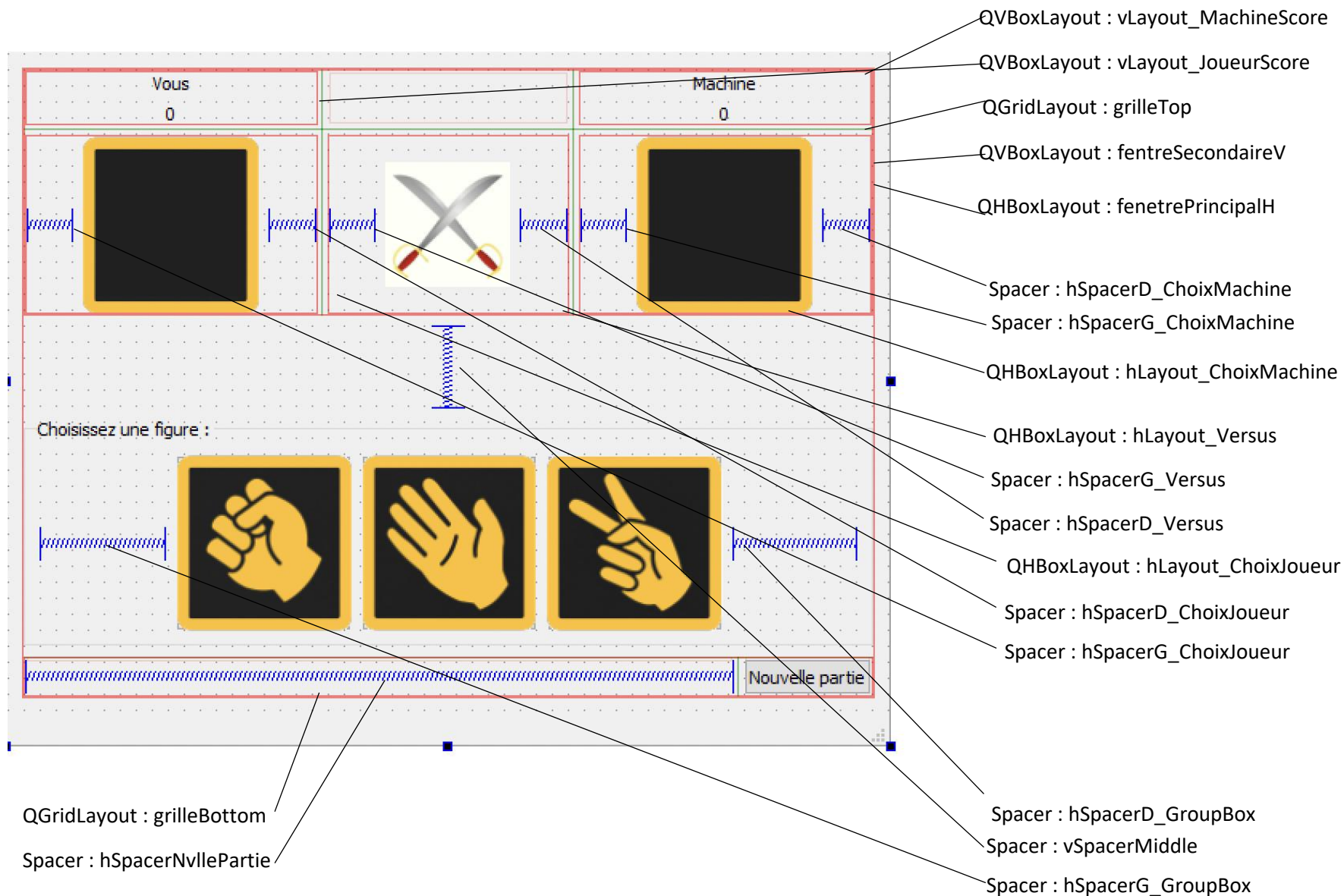
chifoumievue.ui : fichier répertoriant tous les éléments d'interfaces et leurs dispositions fait avec QDesigner

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

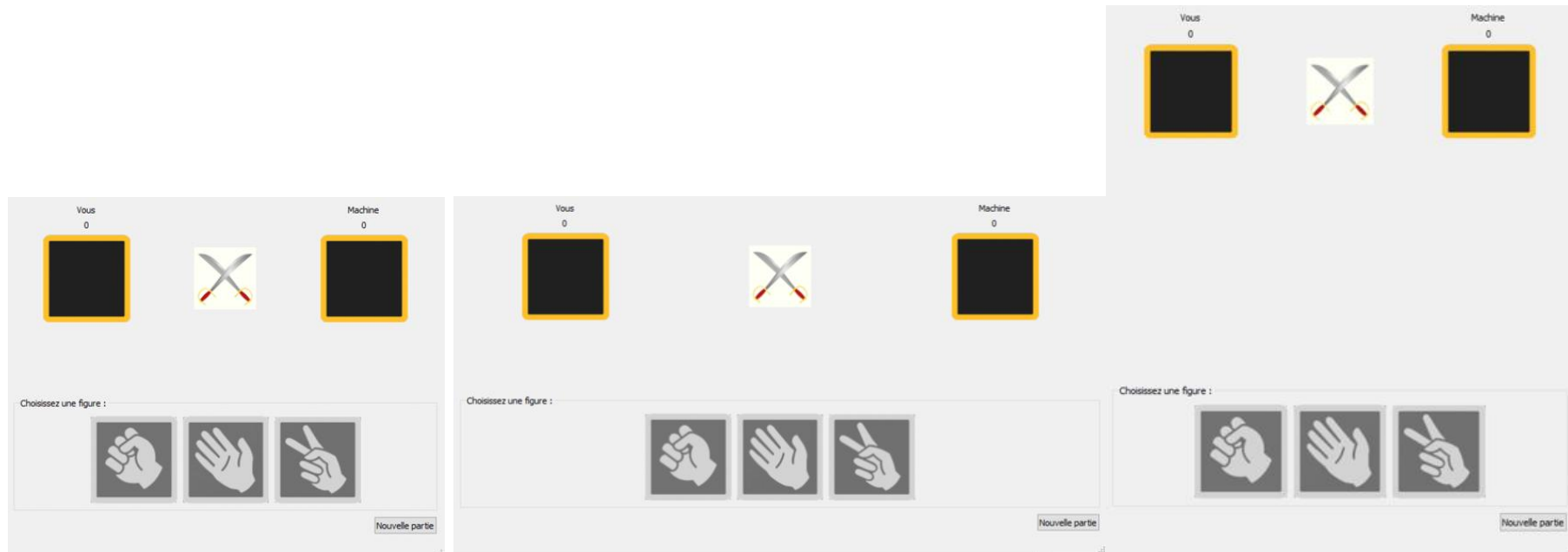
### 3-Element d'interfaces :





#### 4-Test de l'application Chifoumi :

Tests de redimensionnement :



Taille normale

Étirement horizontal

Étirement vertical



## Tests des activités :

Test	Résultat attendu	Validation
Activité 1	<p><u>Système d'information :</u> Le jeu est initialisé avec les variables coups et scores du joueur et de la machine prenant les valeurs de l'état initial</p> <p><u>Interface :</u> Les boutons de signes sont accessibles pour le choix de l'utilisateur Les scores et nom des joueurs sont mis avec une couleur bleue pour indiquer que le jeu est en cours Le focus est sur le bouton 'NvlePartie'</p>	Ok
Activité 2	<p><u>Système d'Information :</u> 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.</p> <p><u>Interface :</u> 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 Le focus est sur le bouton 'NvlePartie'</p>	
Activité 2.1 (Pierre)	//	Ok
Activité 2.2 (Papier)	//	Ok

Activité 2.3 (Ciseau)	//	Ok
Activité 3	<u>Système d'Information :</u> Les scores Joueur/Machine sont mis à 0. Les derniers coups Joueur/Machine joués sont mis à rien. <u>Interface :</u> Les zones d'affichage sont mises en cohérences avec les propriétés précédentes. Le focus est sur le bouton 'NvlePartie'	OK

## Version 2 :

### 1-Liste des fichiers de cette version :

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

Centré sur les informations et les actions métier.

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 Modèle */
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

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 énuméré 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

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

chifoumiVue.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

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->getView()->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();	Retourne l'attribut du modèle	Ok
ChifoumiVue* getView();	Retourne l'attribut de la vue	Ok
UnEtatJeu getEtatJeu();	Retourne l'état de jeu	Ok
void setModele(ChifoumiModele *m);	Affecte la valeur du paramètre dans l'attribut qui pointe vers le modèle	Ok
void setVue(ChifoumiVue *v);	Affecte la valeur du paramètre dans l'attribut qui pointe vers la	Ok

	vue	
void setEtatJeu(UnEtatJeu e);	Affecte la valeur du paramètre comme état du jeu	Ok
Void choixSigne() ;	Si état est etatInitial Rien ne se passe  Si l'état est jeuEnCours 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 Appelle des méthodes de la vue pour afficher les coups et les scores	
Void choixPapier	//juste le signe qui change	Ok
void choixPierre();	// juste le signe qui change	Ok
void choixCiseau();	// juste le signe qui change	Ok
creerNvlePartie();	Si état est etatInitial Affecte l'état du jeu à « jeuEnCour » Appelle des méthodes pour initialiser les scores et les coups dans le modèle Appelle de la méthode pour mettre à jour la vue  Si état est jeuEnCour Appelle des méthodes pour initialiser les scores et les coups dans le modèle Appelle de la méthode pour mettre à jour la vue	Ok

Vue :

Test	Résultat attendu	Validation
ChifoumiPresentation* getPresentation();	Retourne l'attribut de la présentation	Ok
void setPresentation(ChifoumiPresentation *p);	Affecte la valeur du paramètre dans	Ok

	l'attribut qui pointe vers la présentation	
void miseAJour(ChifoumiPresentation::UnEtatJeu etat);	<p>Si état est etatInitial Désactivation des boutons de signes Et focus sur le boutons nouvelle partie</p> <p>Si état est jeuEnCour 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</p>	Ok
void afficherScoreJoueur(unsigned int scoreJ) ;	Affiche le score du joueur	Ok
void afficherScoreMachine(unsigned int scoreM) ;	Affiche le score de la machine	Ok
void afficherCoupJoueur(ChifoumiModele::UnCoup coupJ) ;	Affiche le coup du joueur passé en paramètre	Ok
void afficherCoupMachine(ChifoumiModele::UnCoup coupM) ;	Affiche le coup de la machine passé en paramètre	Ok