

## Saé 2.01 – Développement d'une application

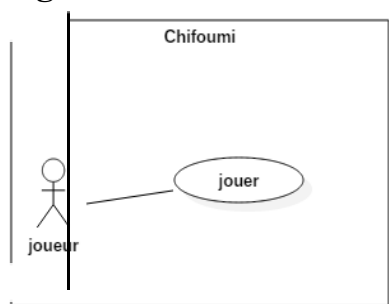
### Chifoumi – Dossier d'Analyse et conception

#### 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.

1.1

#### 2. Diagramme des Cas d'Utilisation



1.2

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.

Tableau 1 :  
Scénario  
nominal

(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*

1.3

## 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.

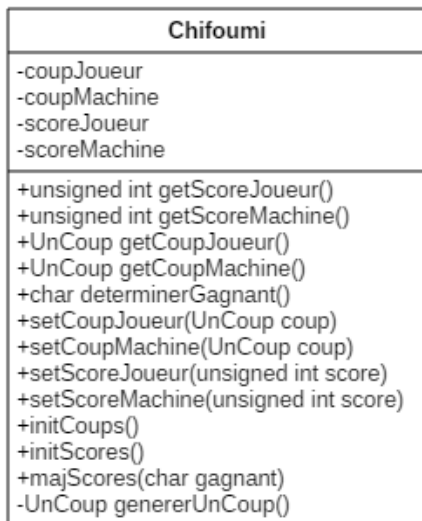


Figure 2 : Diagramme de Classes UML du jeu Chifoumi

(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 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 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.

**1.3.1**

## Version v0

### 5. Implémentation et tests

#### 5.1 Implémentation

Liste des fichiers de cette version :

- chifoumi.h : liste des sous programmes et des variables composants la classe chifoumi
  - chifoumi.cpp : programmation des sous programmes de la classe chifoumi. La classe chifoumi permet de programmer le jeu
  - main.cpp : Programme le jeu a l'aide du chifoumi.cpp et fait les tests.
- Respectivement spécification et corps de la classe Chifoumi décrite au paragraphe 4.

#### 5.2 Test

Test avec le programme fourni main.c(pp

*Valeurs fournies / attendues... comme montré dans la ressource R2.03 (partie tests)*

```
appel du constructeur : construction d'un chifoumi : scores a 0, et coupsJoueurs a RIEN'

teste les methodes get() associees aux attributs 'score'
score Joueur : 0      score Machine : 0

teste les methodes get() associees aux attributs 'coup'
coup Joueur : rien    coup Machine : rien

teste les methodes set() associees aux attributs 'score'
score Joueur : 1      score Machine : 2

teste initScores()
score Joueur : 0      score Machine : 0

teste les méthodes set() et get() associees aux attributs 'coup'/'choix'
coup Joueur : pierre  coup Machine : ciseau

quelques tours de jeu pour tester l'identification du gagnant et la maj des scores
coup Joueur : ciseau  coup Machine : ciseau
score Joueur : 0      score Machine : 0

Quitter ? (o/n)
coup Joueur : papier  coup Machine : papier
score Joueur : 0      score Machine : 0

Quitter ? (o/n)
coup Joueur : ciseau  coup Machine : papier
score Joueur : 1      score Machine : 0
```

Méthodes testées	fichier	valeurs attendues	valeurs obtenues	commentaires
initScores	chifoumi.cpp	0	0	ok
getScoreJoueur()	chifoumi.cpp	0	0	ok
getScoreMachine()	chifoumi.cpp	0	0	ok
initCoups()	chifoumi.cpp	rien	rien	ok

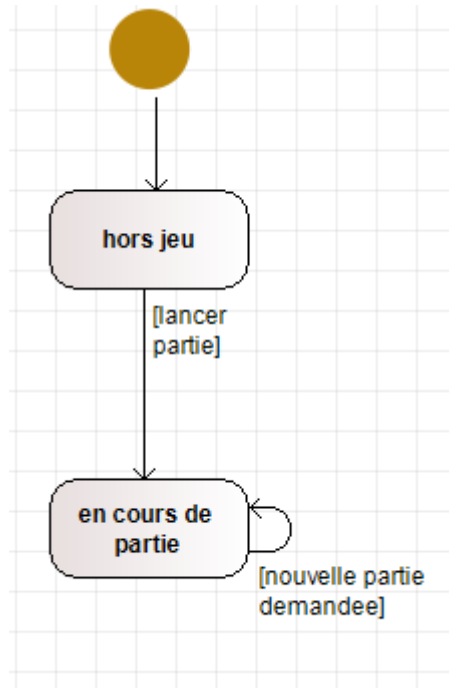
getCoupJoueur()	chifoumi.cpp	rien	rien	ok
getCoupMachine()	chifoumi.cpp	rien	rien	ok
genereUnCoup()	chifoumi.cpp	Pierre Papier ou Ciseau	Pierre Papier ou Ciseau	ok
determinerGagnant ( )	chifoumi.cpp	J M ou N	J M ou N	ok
setScoreJoueur()	chifoumi.cpp	1	1	
setScoreMachine()	chifoumi.cpp			
setCoupJoueur()	chifoumi.cpp			
setCoupMachine()	chifoumi.cpp			

## 6. Classe Chifoumi : Diagramme états-transitions

### (a) Diagramme états-transitions -actions du jeu

*A faire*

Figure 9 : Diagramme états-transitions



(b) Dictionnaires des états, événements et Actions

Dictionnaire des états du jeu

<i>nomEtat</i>	<i>Signification</i>
horsJeu	Le joueur ne joue pas et le logiciel est en attente d'interaction du joueur.
enCoursDePartie	Le joueur est en train de jouer, un score est alors calculé lorsque qqn gagne.

Tableau 2 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

<i>nomÉvénement</i>	<i>Signification</i>
nouvelle Partie	Lorsque le joueur clique sur le bouton "nouvelle partie", le logiciel est alors en cours de partie.
fin de partie	Aucun événement ne permet de quitter la partie.

Tableau 3 : Événements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

activite 1	Pour cela, il suffit de cliquer sur "nouvelle partie"
activite 2	pas de score max dans la v1
activite 3	pas de limite de temps dans la v1

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>	nouvelle partie	fin de la partie
nouvelle partie demandee	cliquer sur "nouvelle partie"	/
En cours de partie	/	Aucun événement

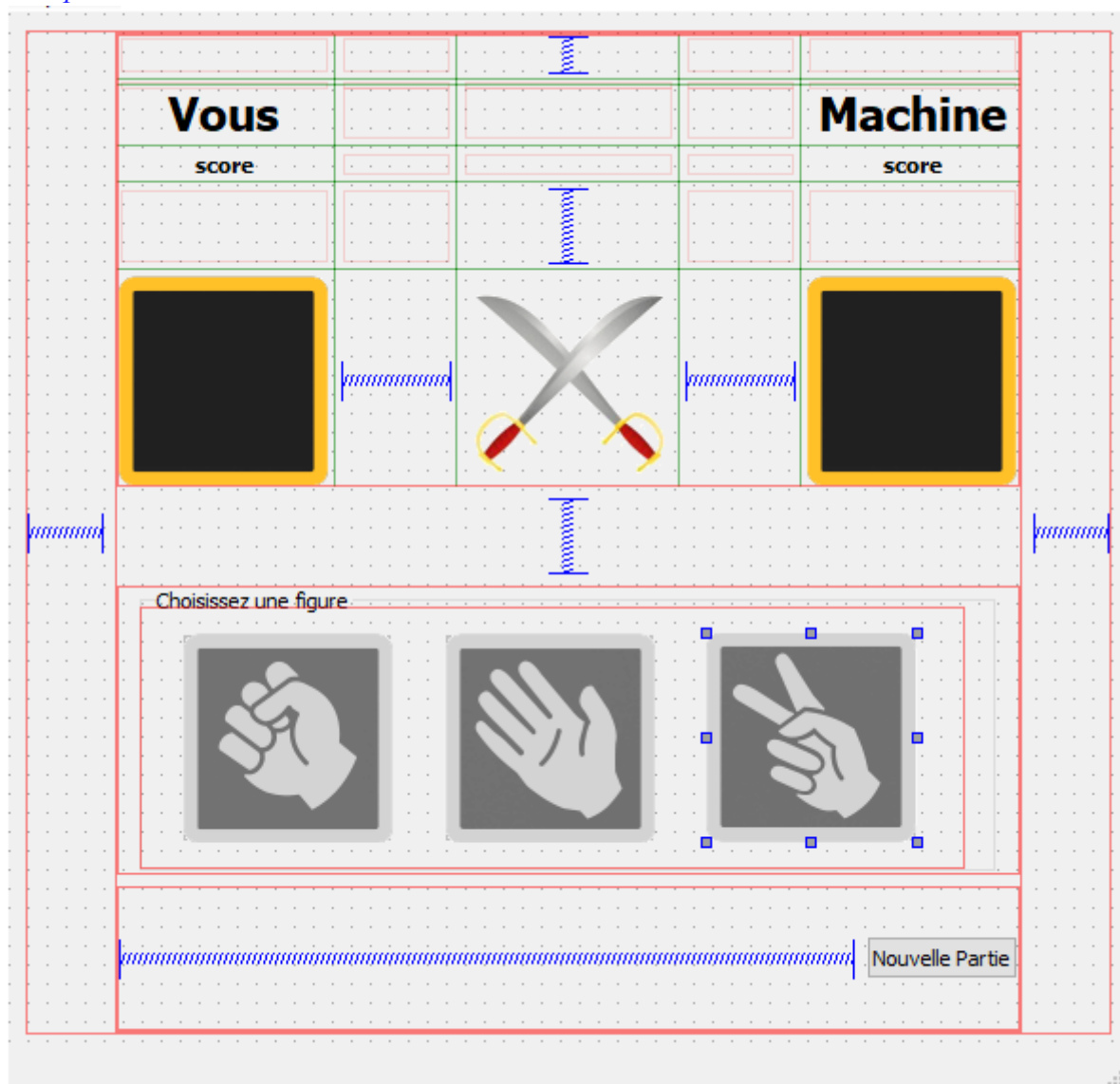
Tableau 5 : Matrice d'états-transitions du jeu chifoumi

*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.*



Le jeu est composé d'une fenêtre principale. Cette fenêtre peut s'agrandir ou se rétrécir. En haut de cette fenêtre sont présents 2 labels avec le texte "Vous" et "Machine" ainsi que leur score juste en dessous. Le score affiche le score de chaque joueur respectif. Vers le centre de la fenêtre sont présentes 3 images avec le signe que le joueur a sélectionné à gauche (Pierre, Papier ou Ciseau), le signe de la machine à droite et une image de deux épées au centre qui sont là en décoration. Les deux photos qui représentent le signe des joueurs sont vides tant que la partie n'est pas commencée. Il y a ensuite, en dessous, les différentes figures que le joueur peut choisir qui sont en fait des boutons. Ces boutons ne sont pas activés tant que la partie n'est pas commencée. Pour lancer une partie, il suffit de cliquer sur le bouton en bas à droite "Nouvelle Partie" qui permet au joueur de jouer.

**boutonPierre:** QPixmap contenant l'image de la pierre.  
**boutonFeuille:** QPixmap contenant l'image de la feuille.  
**boutonCiseau:** QPixmap contenant l'image du ciseau.  
**vs:** QLabel contenant l'image de l'épée.  
**vous:** QLabel contenant "vous"  
**machine:** QLabel contenant "Machine"  
**ScoreJoueur:** QLabel contenant le score du joueur

**ScoreMachine:** QLabel contenant le score de la machine

**boutonNouvellePartie:** QPushButton permet de lancer une nouvelle partie

**coupJoueur:** QLabel Affiche la figure du joueur (Pierre, Papier, Ciseau)

**CoupMachine:** QLabel Affiche la figure de la machine (Pierre, Papier, Ciseau)

## 8. Implémentation et tests

### 8.1 Implémentation

*A faire :*

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

**chifoumi.h:** Liste des sous programmes et des variables composants la classe chifoumi

**chifoumi.cpp:** Programmation des sous programmes de la classe chifoumi. La classe chifoumi permet de programmer le jeu

**mainwindow.h:** Entête du module Chifoumi qui permet de programmer le jeu.

**mainwindow.cpp:** Corps du modèle graphique du jeu chifoumi.

**chifoumi.pro:** Fichier du projet Qt.

**mainwindow.ui:** Fichier contenant les éléments graphiques du jeu.

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

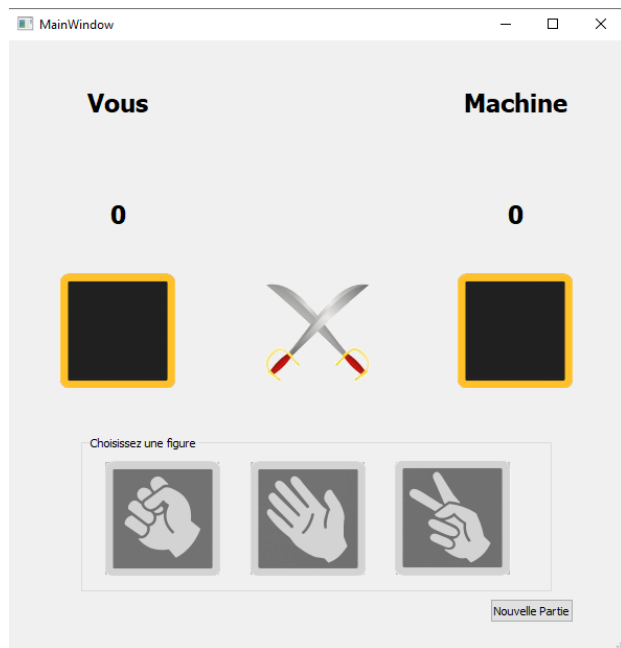
### 8.2 Test

*A faire :*

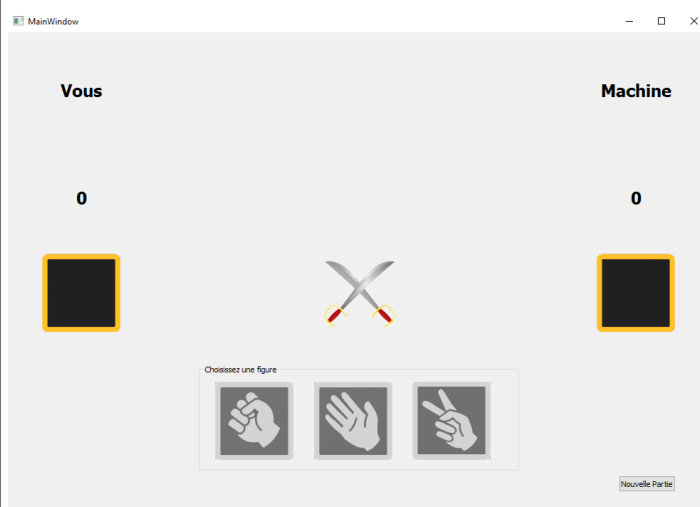
*Décrire les tests prévus / réalisés pour montrer :*

*- Le comportement fonctionnel du programme*

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



**Image1:** avant agrandissement



**Image2:** après agrandissement

