

## Saé 2.01 – Développement d'une application

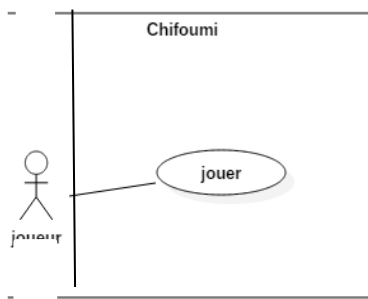
### Chifoumi – Dossier d'Analyse et conception

#### 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.

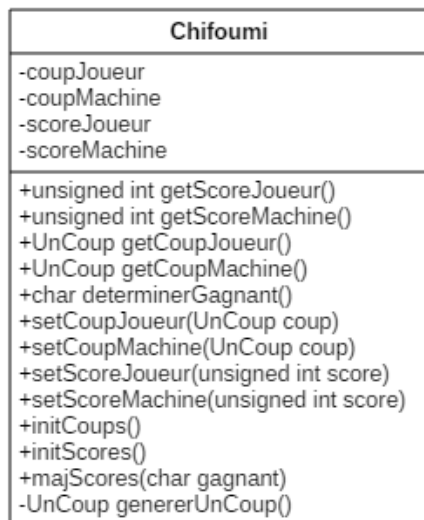


Figure 2 : Diagramme de Classes UML du jeu Chifoumi

(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**

# 1. Version v0

## 5. Implémentation et tests

### 1. 5.1 Implémentation

2. Liste des fichiers de cette version :
3. - chifoumi.h : c'est l'interface de la classe chifoumi
4. - chifoumi.cpp : c'est le corps de la classe
- main.cpp : C'est le jeu et les tests de la classe
5. Respectivement spécification et corps de la classe Chifoumi décrite au paragraphe 4.

### 6. 5.2 Test

Test avec le programme fourni main.cpp

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

### 9.

Méthode	Description	donnée en entrée	résultat attendu	résultat fournies	commentaire
<i>getCoupJoueur()</i>	<i>retourne le dernier coup joué par le joueur</i>		<i>Rien</i>	<i>Rien</i>	<i>ok</i>
<i>getCoupMachine()</i>	<i>retourne le dernier coup joué par la machine</i>		<i>Rien</i>	<i>Rien</i>	<i>ok</i>
<i>getScoreJoueur()</i>	<i>retourne le score du joueur</i>		<i>0</i>	<i>0</i>	<i>ok</i>
<i>getScoreMachine()</i>	<i>retourne le score de la machine</i>		<i>0</i>	<i>0</i>	<i>ok</i>
<i>determinerGagnant()</i>	<i>détermine le gagnant 'J' pour joueur, 'M' pour machine, 'N' pour match nul en fonction du dernier coup joué par chacun d'eux</i>	<i>coupJoueur(p papier) coupMachine(p papier)</i>	<i>N</i>	<i>N</i>	<i>ok</i>
<i>Chifoumi()</i>	<i>initialise le score de la machine et du joueur à 0 et initialise le coup du joueur et de la machine à rien</i>		<i>scores=0 coups = rien</i>	<i>scores = 0 coups =rien</i>	<i>ok</i>

<i>genererUnCoup()</i>	<i>retourne une valeur aléatoire est égal soit à pierre soit à papier ou à ciseau</i>		<i>ciseau</i>	<i>ciseau</i>	<i>ok</i>
<i>setCoupJoueur()</i>	<i>initialise l'attribut coupJoueur avec la valeur du paramètre p_coup</i>	<i>UnCoup p_coup</i>	<i>pierre</i>	<i>pierre</i>	<i>ok</i>
<i>setCoupMachine()</i>	<i>initialise l'attribut coupMachine avec la valeur du paramètre p_coup</i>	<i>UnCoup p_coup</i>	<i>ciseau</i>	<i>ciseau</i>	<i>ok</i>
<i>initScores()</i>	<i>initialise à 0 les attributs scoreJoueur et scoreMachine</i>		<i>scoreJoueur = 0</i> <i>scoreMachine = 0</i>	<i>scoreJoueur = 0</i> <i>scoreMachine = 0</i>	<i>ok</i>
<i>initCoups()</i>	<i>initialise à rien les attributs coupJoueur et coupMachine</i>		<i>coupJoueur = rien</i> <i>coupMachine = 0</i>		<i>ok</i>
<i>setScoreJoueur()</i>	<i>initialise l'attribut scoreJoueur avec la valeur du paramètre p_score</i>	<i>unsigned int p_score</i>	<i>scoreJoueur = 1</i>	<i>scoreJoueur = 1</i>	<i>ok</i>
<i>setScoreMachine()</i>	<i>initialise l'attribut scoreMachine avec la valeur du paramètre p_score</i>	<i>unsigned int p_score</i>	<i>scoreMachine = 2</i>	<i>scoreMachine = 2</i>	<i>ok</i>
<i>Joueur: ciseau</i> <i>Machine: ciseau</i>			<i>scoreJoueur = 0</i> <i>scoreMachine = 0</i> <i>CoupJoueur = ciseau</i>	<i>scoreJoueur = 0</i> <i>scoreMachine = 0</i> <i>CoupJoueur = ciseau</i>	<i>ok</i>

			<i>CoupMachine = ciseau</i>	<i>CoupMachine = ciseau</i>	
<i>Joueur: ciseau Machine: papier</i>			<i>scoreJoueur = 1 scoreMachine = 0 CoupJoueur = ciseau CoupMachine = papier</i>	<i>scoreJoueur = 1 scoreMachine = 0 CoupJoueur = ciseau CoupMachine = papier</i>	<i>ok</i>
<i>Joueur: ciseau Machine: pierre</i>			<i>scoreJoueur = 0 scoreMachine = 1 CoupJoueur = ciseau CoupMachine = pierre</i>	<i>scoreJoueur = 0 scoreMachine = 1 CoupJoueur = ciseau CoupMachine = pierre</i>	<i>ok</i>
<i>Joueur: papier Machine: papier</i>			<i>scoreJoueur = 0 scoreMachine = 0 CoupJoueur = papier CoupMachine = papier</i>	<i>scoreJoueur = 0 scoreMachine = 0 CoupJoueur = papier CoupMachine = papier</i>	<i>ok</i>
<i>Joueur: papier Machine: ciseau</i>			<i>scoreJoueur = 0 scoreMachine = 1 CoupJoueur = papier CoupMachine = ciseau</i>	<i>scoreJoueur = 0 scoreMachine = 1 CoupJoueur = papier CoupMachine = ciseau</i>	<i>ok</i>
<i>Joueur: papier Machine: pierre</i>			<i>scoreJoueur = 1 scoreMachine = 0 CoupJoueur = papier CoupMachine = pierre</i>	<i>scoreJoueur = 1 scoreMachine = 0 CoupJoueur = papier CoupMachine = pierre</i>	<i>ok</i>
<i>Joueur: pierre Machine: pierre</i>			<i>scoreJoueur = 0 scoreMachine = 0 CoupJoueur = pierre CoupMachine = pierre</i>	<i>scoreJoueur = 0 scoreMachine = 0 CoupJoueur = pierre CoupMachine = pierre</i>	<i>ok</i>
<i>Joueur: pierre Machine: ciseau</i>			<i>scoreJoueur = 1 scoreMachine = 0 CoupJoueur = pierre CoupMachine = ciseau</i>	<i>scoreJoueur = 1 scoreMachine = 0 CoupJoueur = pierre CoupMachine = ciseau</i>	<i>ok</i>
<i>Joueur: pierre Machine: papier</i>			<i>scoreJoueur = 0 scoreMachine = 1</i>	<i>scoreJoueur = 0 scoreMachine = 1</i>	<i>ok</i>

			<i>CoupJoueur = pierre CoupMachine = papier</i>	<i>CoupJoueur = pierre CoupMachine = papier</i>	
--	--	--	---	---	--

10.



## 2. Version v1

### 6. Classe Chifoumi : Diagramme états-transitions

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

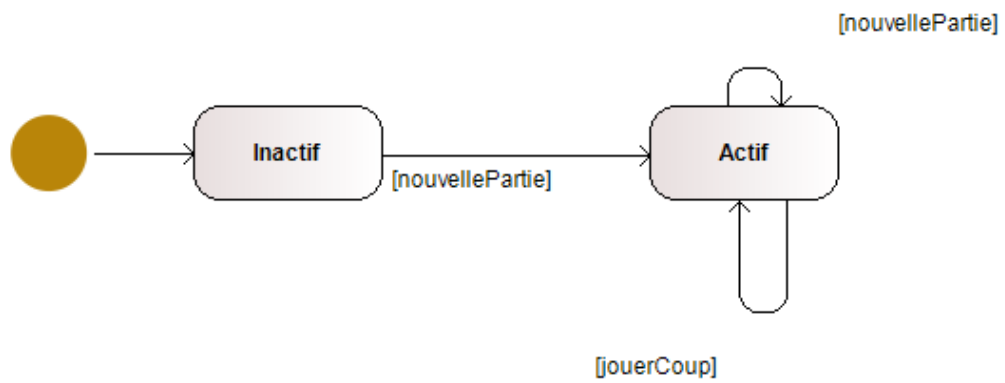


Figure 9 : Diagramme états-transitions

(b) Dictionnaires des états, événements et Actions

Dictionnaire des états du jeu

<i>nomEtat</i>	<i>Signification</i>
Inactif	C'est quand la partie n'a pas encore commencé
Actif	C'est quand la partie a commencé

Tableau 2 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

<i>nomÉvénement</i>	<i>Signification</i>
NouvellePartie	C'est quand le joueur demande soit à jouer une partie
JouerCoup	C'est quand le joueur clique sur son choix

Tableau 3 : Événements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

Nouvelle partie	on réinitialise la partie en appuyant sur le bouton nouvelle partie
Jouer un coup	le joueur choisit son coup
Quitter la partie	L'application réinitialise les scores

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> 	JouerCoup	NouvellePartie
<i>nomEtatJeu</i>		
Inactif	Bouton "nouvelle partie"	X
Actif	bouton ciseau/pierre/papier	Bouton "nouvelle partie"

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.*

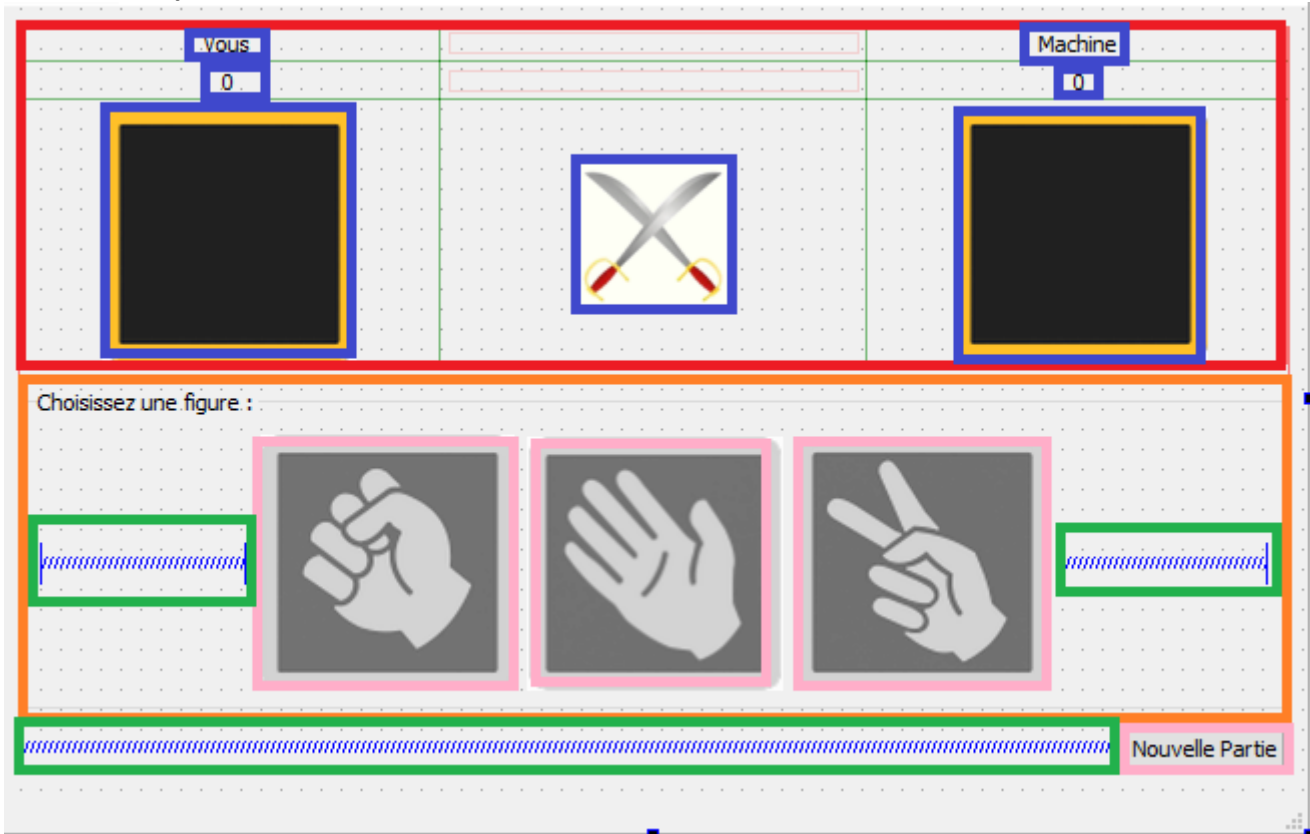
1.

## 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.*

## 8. Implémentation et tests

### 1. 8.1 Implémentation



Cadre rouge: GridLayout

Cadre Bleu: Label

Cadre Orange: GroupBox

Cadre Rose: PushButton

Cadre Vert: Spacer

### 2. *A faire :*

mainwindow.cpp: C'est toutes les définitions des fonctions et des procédures qui sont déclaré dans le mainwindow.h.Elle permet aussi de connecter des fonctions ou des procédures

mainwindow.h: C'est la documentation du sous-programme du mainwindow(l'entête).Elle permet de aussi de déclarer les fonctions ou les procédures en tant que slot.

main.cpp: C'est le programme principale qui permet d'afficher le mainwindow.ui.

ressourcesChifoumi.qrc: C'est le dossier qui contient nos images que nous utilisons dans le mainwindow.ui.

mainwindow.ui: C'est le programme qui va nous permettre de modiliser graphiquement l'application.

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

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

Signals:

bPierre: C'est le bouton "pierre" qui va permettre au joueur de jouer pierre.  
bFeuille: C'est le bouton "feuille" qui va permettre au joueur de jouer feuille.  
bCiseaux: C'est le bouton "ciseaux" qui va permettre au joueur de jouer ciseaux.  
bNouvellePartie: C'est le bouton qui va permettre au joueur de réinitialiser la partie et de lancer une nouvelle partie.

Slots:

jouerTourPierre(): Permet de mettre la variable CoupJoueur à pierre, d'afficher l'image pierre au joueur, de faire jouer la machine aléatoirement, d'augmenter de un le score du gagnant et d'afficher le nouveau score.  
jouerTourFeuille(): Permet de mettre la variable CoupJoueur à pierre, d'afficher l'image feuille au joueur, de faire jouer la machine aléatoirement, d'augmenter de un le score du gagnant et d'afficher le nouveau score.  
jouerTourCiseaux(): Permet de mettre la variable CoupJoueur à pierre, d'afficher l'image feuille au joueur, de faire jouer la machine aléatoirement, d'augmenter de un le score du gagnant et d'afficher le nouveau score.  
nouvellePartie(): Permet d'initialiser le coup du joueur et de la machine à rien, initialise les scores de la machine et du joueur à zéro et met les boutons "pierre", "feuille" et "ciseaux" en cliquable.

5.

6.

7. **8.2 Test**

8. *A faire :*

tests:

comme dans la version 0 avec DeterminerGagnant():

les boutons:

Classe	description	variable	résultat attendu	résultat fournie	commentaire
choix du joueur : pierre	le joueur appuie sur le bouton pierre	bPierre	ICoupJoueur = image de pierre	ICoupJoueur = image de pierre	ok
choix du joueur : feuille	le joueur appuie sur le bouton feuille	bFeuille	ICoupJoueur = image de feuille	ICoupJoueur = image de feuille	ok
choix du joueur : ciseaux	le joueur appuie sur le bouton ciseaux	bCiseaux	ICoupJoueur = image de ciseaux	ICoupJoueur = image de ciseaux	ok
lancer partie	le joueur appuie	bNouvellePartie	scoreMachine =	scoreMachine =	ok

	sur le bouton NouvellePartie		0 scoreJoueur = 0 lCoupJoueur = image de rien lCoupMachine = image rien les boutons bPierre,bFeuille et bCiseau sont maintenant actif	0 scoreJoueur = 0 lCoupJoueur = image de rien lCoupMachine = image rien les boutons bPierre,bFeuille et bCiseau sont maintenant actif	
démarrer une nouvelle partie	appuyer sur le bouton nouvelle partie pendant une partie	bNouvellePartie	scoreMachine = 0 scoreJoueur = 0 lCoupJoueur = image de rien lCoupMachine = image rien	scoreMachine = 0 scoreJoueur = 0 lCoupJoueur = image de rien lCoupMachine = image rien	ok

bPierre: Que si le joueur appuie sur le bouton pierre, l'image pierre est affiché

bFeuille:Que si le joueur appuie sur le bouton feuille, l'image feuille est affiché

bCiseaux:Que si le joueur appuie sur le bouton ciseaux, l'image ciseaux est affiché

bNouvellePartie: Que si le joueur appuie sur le bouton NouvellePartie, les boutons bPierre,bFeuille,bCiseaux soit cliquable, que le score de la machine et du joueur soit remis à zéro et que les coups de la machine et du joueur ne soient mis à rien.

9. *Décrire les tests prévus / réalisés pour montrer :*

- *Le comportement fonctionnel du programme*

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