

KOFFI

Jean-Jonathan

TD2-TP3

But Informatique première année

Saé 2.01 – Développement d'une application

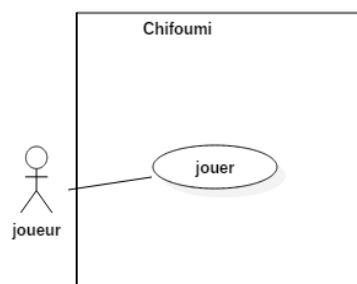
Chifoumi – Dossier d'Analyse et conception

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.

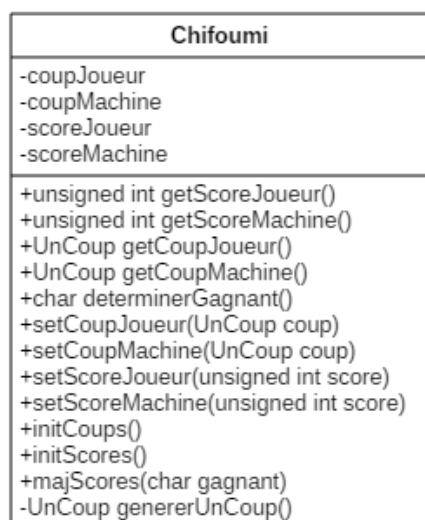


Figure 2 : Diagramme de Classes UML du jeu Chifoumi

(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 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

(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 :
- chifoumi.cpp :

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

5.2 Test

Test avec le programme fourni main.cpp

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

<i>Classe</i>	<i>Description</i>	<i>Valeur fournie</i>	<i>Résultat attendue</i>
Valide n°1	Test les methodes get() associer aux attribut 'score'	scoreJoueur : 0 scoreMachine : 0	scoreJoueur : 0 scoreMachine : 0
Valide n°2	Test les methodes get() associer aux attribut 'coup'	coupJoueur : rien coupMachine : rien	coupJoueur :rien coupMachine : rien
Valide n°3	Test les methodes set() associer aux attribut 'score'	setScore(1) setScore(2)	scoreJoueur : 1 scoreMachine :2
Valide n°4	Test initScores()	scoreJoueur : 0 scoreMachine :0	scoreJoueur : 0 scoreMachine :0
Valide n°4 Valide n°5	Test les methodes set() et get() associés aux attributs 'coup/choix'	setCoupJoueur(pierre) setCoupMachine(ciseau)	coupJoueur :pierre coupMachine : ciseau
Valide n°6	Test sur l'identification du gagnant et la maj des scores	coupJoueur : papier coupMachine : papier	scoreJoueur = 0 scoreMachine =0
Valide n°7	Test sur l'identification du gagnant et la maj des scores	coupJoueur : ciseau coupMachine : papier	scoreJoueur = 1 scoreMachine =0

Version v1

6. Classe Chifoumi : Diagramme états-transitions

(a) Diagramme états-transitions -actions du jeu

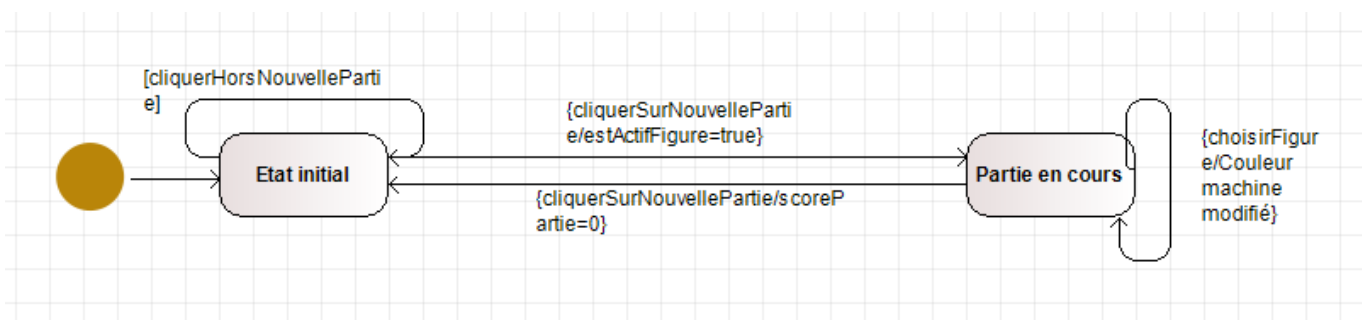


Figure 4 : Diagramme états-transitions

(b) Dictionnaires des états, événements et Actions

Dictionnaire des états du jeu

<i>nomEtat</i>	<i>Signification</i>
Etat initial	L'état dans lequel l'application se trouve lors de son déclenchement et dans lequel elle se met en attente et en réponse à tout événement extérieur
Partie en cours	L'état dans lequel l'application se trouve lorsque l'événement associé au bouton Nouvelle partie est déclenché et que les boutons figures sont actifs

Tableau 3 : États du jeu

Dictionnaire des événements faisant changer le jeu d'état

<i>nomEvénement</i>	<i>Signification</i>
CliquerSurNouvellePartie partie	L'événement qui permet d'établir la transition entre l'état initial et partie en cours Ou entre la partie en cours et l'état initial (lorsque le joueur veut recommencer une nouvelle partie)
ChoisirFigure	L'événement qui permet au joueur de jouer lorsque les boutons figures sont actifs

Tableau 4 : Événements faisant changer le jeu d'état

Description des actions réalisées lors de la traversée des transitions

<i>ActionRealisé</i>	<i>Signification</i>
estActifFigure = True	Cette action traduit le fait que lorsque le joueur clique sur Nouvelle partie, les boutons-figures sont actifs. Le nom et score du joueur sont de couleur bleue (la partie en cours)
ScorePartie = 0	Cette action correspond à une affectation qui traduit le fait que le joueur décide de recommencer la partie, les scores sont remis à 0
couleurMachineModifier	A la fin du tour du joueur, les nom et score du joueur machine sont de couleur bleue pour dire que c'est le tour de la machine et réciproquement pour le joueur

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

<i>Événement →</i> <i>nomEtatJeu</i>	CliquerSurNouvellePartie	choisirFigure
Etat initial	Partie en cours	-----
Partie en cours	Etat initial	Partie en cours

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.

8. Implémentation et tests

8.1 Implémentation

A faire :

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

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

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*