Université de Pau et des Pays de l'Adour - IUT DE BAYONNE - PAYS-BASQUE année 2021-2022

Département Informatique saé 20.1

Saé 2.01 – Développement d’une application

Chifoumi – Dossier d’Analyse et conception



## 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.

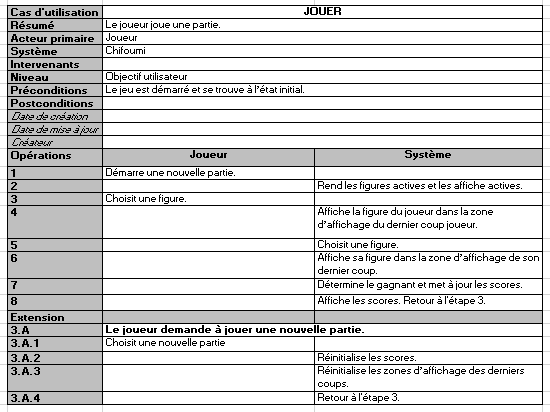
## Diagramme des Cas d’Utilisation

## 

*Figure 1 : Diagramme des Cas d’Utilisation du jeu Chifoumi*

## Scénarios

1. **Exemple Scénario**

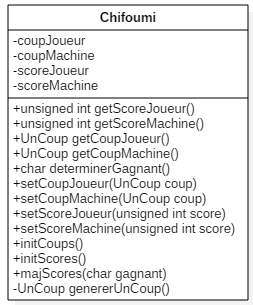


*Tableau 1 : Scénario nominal*

1. **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*

## Diagramme de classe (UML)

1. 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*

1. 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*

1. 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*

1. 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.

# Version v0

## 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

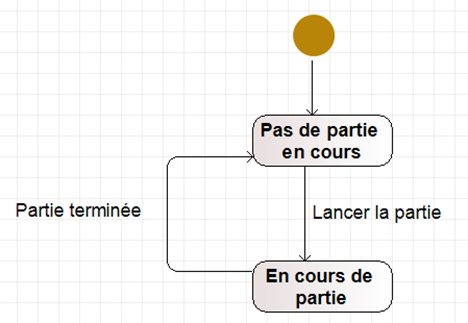
*Valeurs fournies / attendues… comme montré dans la ressource R2.03 (partie tests)*

| Classe | Description | valeur en entrée | valeur en sortie | résultat |
| --- | --- | --- | --- | --- |
| getScoreJoueur | récupère la valeur du dernier score du joueur | 0 | 0 | ok |
| getScoreMachine | récupère la valeur du dernier score de la machine | 0 | 0 | ok |
| getCoupJoueur | récupère la valeur du dernier coup de la machine | rien | rien | ok |
| getCoupMachine | récupère la valeur du dernier coup de la machine | rien | rien | ok |
| setScoreJoueur | Initialise le score du joueur avec une valeur (p\_score) | 0 | 1 | ok |
| setScoreMachine | Initialise le score de la machineune valeur (p\_score) | 0 | 2 | ok |
| initScore | Initialise les scores de la machine et du joueur | 0 | 0 | ok |
| setCoupMachine | Initialise le coup du joueur avec une valeur (p\_score) | rien | ciseau | ok |
| setCoupJoueur | Initialise le coup du joueur avec une valeur (p\_score) | rien | Pierre | ok |

# Version v1

## Classe Chifoumi : Diagramme états-transitions

1. **Diagramme états-transitions -actions du jeu**



*Figure 9 : Diagramme états-transitions*

Dictionnaires des états, événements et Actions

**Dictionnaire** **des états du jeu**

| *nomEtat* | *Signification* |
| --- | --- |
| partieStoppe | La partie n’a pas débuté |
| partieEnCours | La partie est en cours |

Tableau 2 : États du jeu

**Dictionnaire des événements faisant changer le jeu d’état**

| *nomEvénement* | *Signification* |
| --- | --- |
| partieCommence | La partie est lancé par le joueur |
| partieTermine | La partie se termine soit par victoire ou plus de temps |

Tableau 3 : Evénements faisant changer le jeu d’état

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

| Clique sur commencer la partie | l’utilisateur clique sur commencer la partie |
| --- | --- |
| Le score maximal est atteint | Soit l’utilisateur soit le joueur à atteint le score maximal |
| le temps maximal est atteint | le temps est atteint et la partie se termine |

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

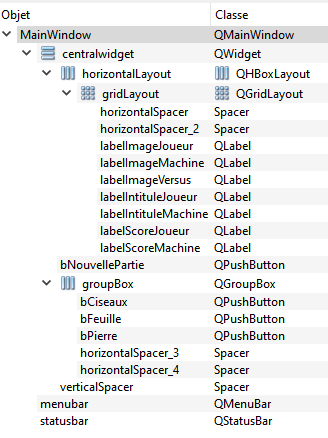
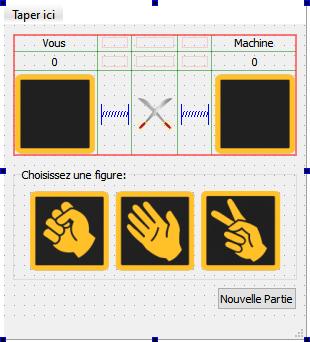
-  en *colonne : les* ***états*** du jeu

| *Événement  à*  *nomEtatJeu* | partie à l'arrêt | partie en cours |
| --- | --- | --- |
| lance l’application | 1 | 0 |
| clique sur “nouvelle partie” | 0 | 1 |

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.*

## Éléments d’interface



L’interface est séparée en deux parties, la première en haut est l’affichage des scores et et le déroulement du jeu, la deuxième étant le choix de la figure.

La première partie étant une partie regroupant l’affichage des scores et l’affichage du coup machine et du coup joueur nous avons fait le choix de tout regrouper sur une grille afin d’avoir un comportement adéquat lors de l’élargissement de la page.

Sur la première colonne, nous avons mis trois label : le premier précisant à qui appartient le score (labelIntituleJoueur), le deuxième le score joueur (labelScoreJoueur) et le troisième l’affichage du choix joueur sous forme d’image (labelImageJoueur).

Les 2,3 et 4èmes colonnes sont dédiées à l'habillage et l’image des épées croisées entre les deux (labelImageVersus).

La dernière colonne est la même que la première à la différence que c’est l’affichage du score machine et du choix de celle-ci (labelIntituleMachine, labelScoreMachine, labelImageMachine).

Dans la seconde partie se déroule le choix des figures jouées par le joueur, il a donc 3 boutons symbolisant les figures jouables (Pierre, Feuille, Ciseaux). Chaque choix est représenté par un bouton contenant l’image du coup à jouer (bPierre, bFeuille, bCiseaux). En dessous de se bloque, on retrouve le bouton de relance d’une nouvelle partie, remettant les scores à 0 (bNouvellePartie).

*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.*

## Implémentation et tests

### 8.1 Implémentation

**FICHIERS:**

mainwindow.h: Il gère l’initialisation du code de la fenêtre de jeu

chifoumi..h: Il gère l’initialisation du code du déroulement de la partie

chifoumi.cpp : Il gère le déroulement des différentes actions décrite dans le chifoumi.h

main.cpp: Lancement de la partie en appelant et la fenêtre et le code de la partie

mainwindow.cpp: Il gère l’affichage direct de la fenêtre de la partie

**SIGNAL/SLOT:**

### A faire :

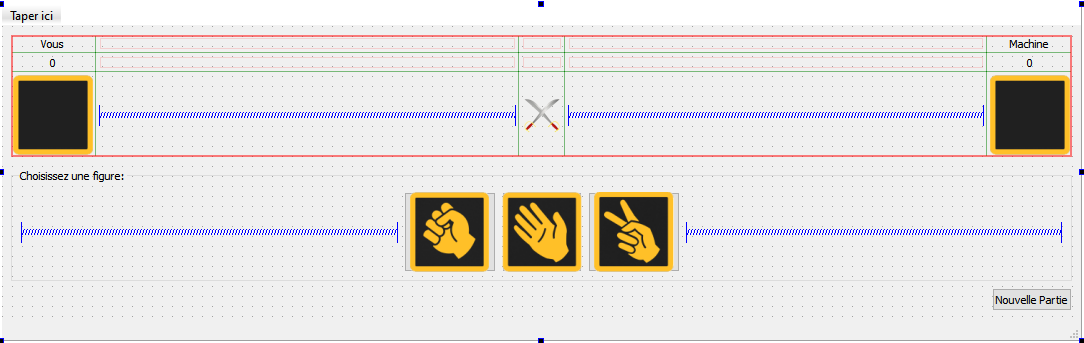
### lister les fichiers impliqués dans cette version (répertoire, nom de fichier, rôle de chaque fichier)

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

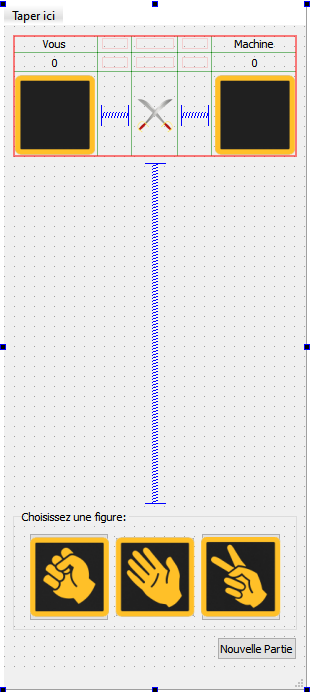
### 8.2 Test

Tests prévus pour les layout :

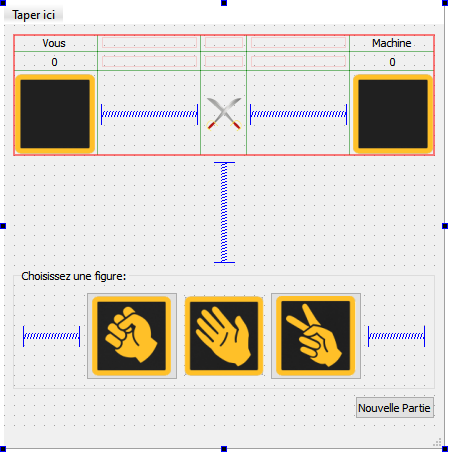
* Élargissement de la page sur la largeur avec les élargissements adéquats.



Élargissement de la page sur la longueur avec les élargissements adéquats.

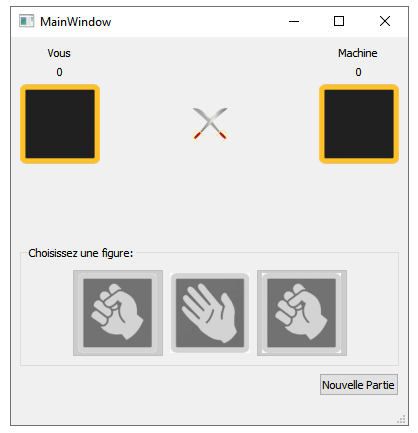


* Élargissement de la page sur la diagonale avec les élargissements adéquats.

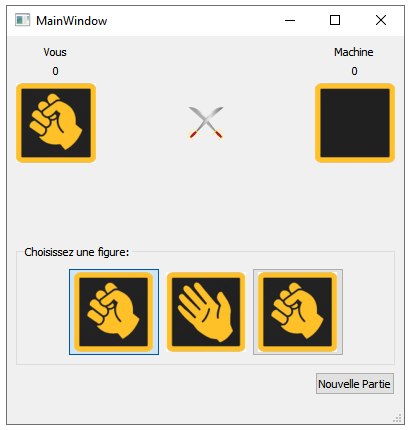


Tests prévu sur l'exécution du programme :

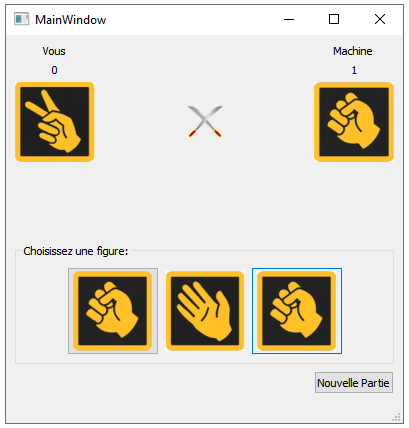
* État initiale de l’application, impossibilité de cliqué sur les boutons figures et affichage nul sur les images et les scores



* L’utilisateur clique sur le bouton pierre et la pierre s’affiche sur le labelImageMachine avec le score associé



* L’utilisateur clique sur le bouton ciseaux et la pierre s’affiche sur le labelImageMachine avec le score associé



* L’utilisateur clique sur le bouton feuille et la pierre s’affiche sur le labelImageMachine avec le score associé

vbgjf

### 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