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 :

### - main.cpp : Contient le jeu et les tests

### - chifoumi.h : L’en tête de la classe pour programmer le jeu

### - chifoumi.cpp : Le corps de la classe pour programmer le jeu

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

### 

### 5.2 Test

Test avec le programme fourni main.cpp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Méthodes Testées | Fichier | Données | Valeur(s) attendue(s) | Valeur(s) Obtenue(s) | Commentaire |
| afficherScores() ; | main.exe | Score joueur et score machine égales à 0 | score Joueur : 0  score Machine : 0 | score Joueur : 0  score Machine : 0 | OK |
| afficherCoups(); | main.exe | Score joueur et score machine égales à « rien » | coup Joueur : rien  coup Machine : rien | coup Joueur : rien  coup Machine : rien | OK |
| setScoreJoueur();  setScoreMachine(); | main.exe | scoreJoueur = 1  scoreMachine = 2 | scoreJoueur : 1  scoreMachine : 2 | scoreJoueur : 1  scoreMachine : 2 | OK |
| initScores(); | main.exe | Aucune | ScoreJoueur = 0  scoreMachine = 0 | scoreJoueur : 0  scoreMachine :0 | OK |
| setCoupJoueur()  setCoupMachine() | main.exe | coupJoueur= pierre  coupMachine= papier | coupJoueur : pierre  coupMachine : papier | coupJoueur : pierre  coupMachine : papier | OK |
| determinerGagnant(); | main.exe | coupJoueur=pierre  coupMachine=ciseau | scoreJoueur++  scoreMachine== | scoreJoueur++  scoreMachine== | OK |

# Version v1

## Classe Chifoumi : Diagramme états-transitions

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

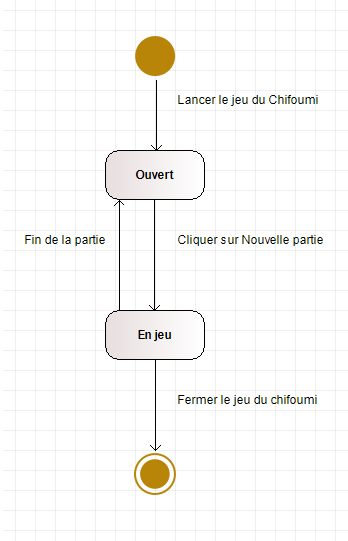


Figure 9 : Diagramme états-transitions

## Dictionnaires des états, événements et Actions

**Dictionnaire** **des états du jeu**

|  |  |
| --- | --- |
| *nomEtat* | *Signification* |
| Ouvert | L’application est ouverte et le jeu n’est pas lancé |
| En Jeu | Le jeu est en cours de partie |

Tableau 2 : États du jeu

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

|  |  |
| --- | --- |
| *nomEvénement* | *Signification* |
| Nouvelle partie | Lance une nouvelle partie |
| Fin de partie | Stop la partie, la partie est finie |

Tableau 3 : Evénements faisant changer le jeu d’état

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

|  |  |
| --- | --- |
| Lancer le jeu  du chifoumi | L’utilisateur lance le jeu et la fenêtre se crée… |
| Cliquer sur nouvelle  partie | L’utilisateur clique sur bouton nouvelle partie |
| Fin de la partie | L’utilisateur a fini sa partie et le jeu se réinitialise |
| Fermer le jeu  Du chifoumi | L’utilisateur ferme le jeu |

Tableau 4 : Actions à réaliser lors des changements d’état

1. **Préparation au codage :**

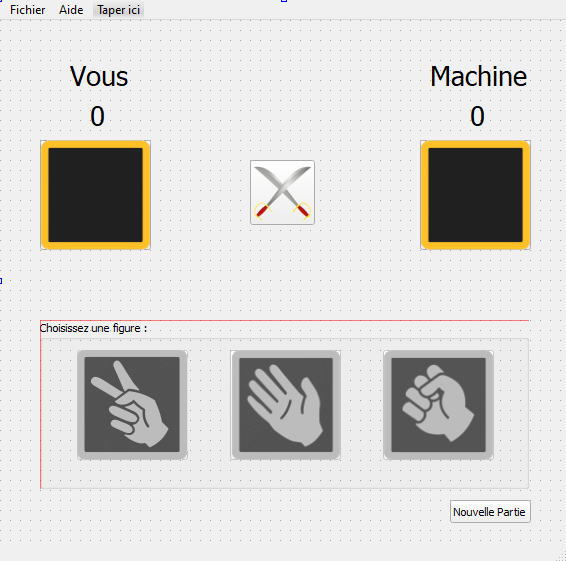
**Table T\_EtatsEvenementsJeu** correspondant à la version matricielle du diagramme états-transitions du jeu :

* en *ligne : les* ***événements*** faisant changer le jeu d’état
* en *colonne : les* ***états*** du jeu

|  |  |  |
| --- | --- | --- |
| *Événement 🡪*  *nomEtatJeu* | Cliquer sur nouvelle partie | Fin de la partie |
| Ouvert | Faux | Vrai |
| En jeu | Vrai | Faux |

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

## Éléments d’interface



-Les menus fichier et aide en haut ne sont pas fonctionnels

-Dans la partie supérieure de la fenêtre, on voit d’abord le score joueur et le score machine, juste en dessous, le coup du joueur et le coup de la machine

-Dans la partie Inférieure, le joueur choisit la figure qui correspond au coup qu’il veut jouer en appuyant sur le bouton associé (pour l’instant grisé parce que la partie n’est pas lancée), il doit d’abord cliquer sur le bouton nouvelle partie en bas à droite

## Implémentation et tests

### 8.1 Implémentation

### MonChifoumi :

### chifoumi.cpp : Définition des méthodes de la classe chifoumi.

### chifoumi.h : Déclaration de la classe chifoumi et toutes ses méthodes.

### chifoumivue.cpp : Définition des méthodes de la classe chifoumivue et connection boutons-slots

### chifoumivue.h : Déclaration de la classe chifoumivue et toutes ses méthodes (ce fichier fera le lien avec l’ui)

### chifoumivue.ui : Interface de la fenêtre principale

### main.cpp : prépare et affiche la fenêtre principale

### MonChifoumi.pro : Fichier de configuration

### Images :

### (Toutes les images de pierre/papier/ciseaux)

### chifoumi\_dossierAnalyseConceptionACompléter\_v2.docx : Fichier d’analyse

### 8.2 Test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Méthodes Testées | Fichier | Données | Valeur(s) attendue(s) | Valeur(s) Obtenue(s) | Commentaire |
| NouvellePartie() | chifoumivue.cpp | Signal du BoutonPartie cliqué | Coups débloqués | Coups débloqués | OK |
| CoupPierre() | chifoumivue.cpp | Signal de Pierre cliquée | Mise à jour des scores suivant le gagnant | Mise à jour des scores suivant le gagnant | OK |
| CoupFeuille() | chifoumivue.cpp | Signal de Feuille cliquée | Mise à jour des scores suivant le gagnant | Mise à jour des scores suivant le gagnant | OK |
| CoupCiseaux() | chifoumivue.cpp | Signal de Ciseaux cliquée | Mise à jour des scores suivant le gagnant | Mise à jour des scores suivant le gagnant | OK |