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

Département Informatique Saé S2.01



KOFFI

Jean-Jonathan

TD2-TP3

But Informatique première année

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 fournie* | *Résultat attendue* |
| *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

## Classe Chifoumi : Diagramme états-transitions

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

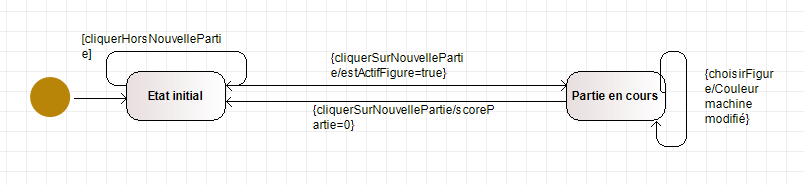
**

Figure 9 : Diagramme états-transitions

## Dictionnaires des états, événements et Actions

**Dictionnaire** **des états du jeu**

|  |  |
| --- | --- |
| *nomEtat* | *Signification* |
| Etat initial | L’état dans lequel l’application se trouve lors de son déclanchement et dans lequel elle se met en attente et en réponse à tout évènement extérieur |
| Partie en cours | L’état dans lequel l’application se trouve lorsque l’évènement associer au bouton Nouvelle partie est déclenché et que les boutons figures sont actifs |

Tableau 2 : États du jeu

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

|  |  |
| --- | --- |
| *nomEvénement* | *Signification* |
| CliquerSurNouvellePartie partie | L’évènement qui permet d’établir la transition entre l’etat intial et partie en cours  Ou entre la partie en cours et l’état initial( lorsque le joueur veut recommencer une nouvelle partie) |
| ChoisirFigure | L’évènement qui permet au joueur de jouer lorsque les boutons figures sont actifs |

Tableau 3 : Evénements faisant changer le jeu d’état

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

|  |  |
| --- | --- |
| *ActionRealisé* | *Signification* |
| 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 affeciton qui traduit le fait que le joueur decide 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 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

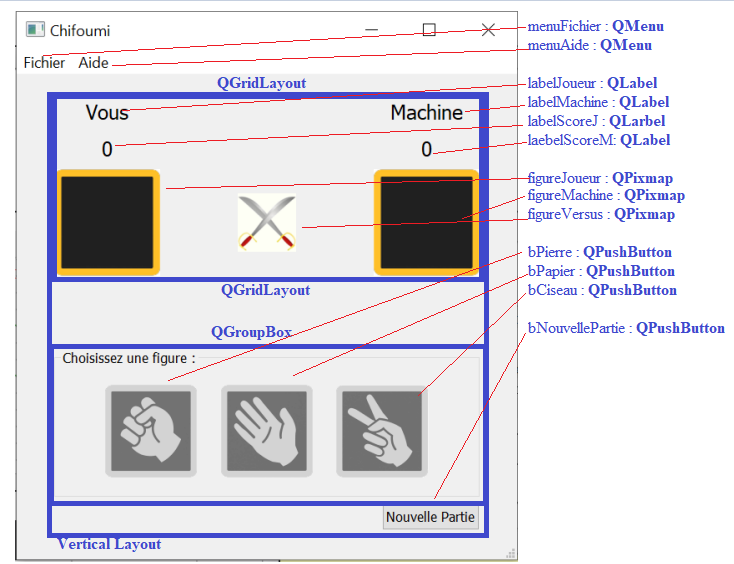
|  |  |  |
| --- | --- | --- |
| *Événement 🡪*  *nomEtatJeu* | CliquerSurNouvellePartie | choisirFigure |
| Etat initial | Partie en cours | -------- |
| Partie en cours | Etat initial | Partie en cours |

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

*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

### A faire :

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

### Liste des fichiers sources et rôle de chaque fichier

### chifoumivue.h

### chifoumivue.cpp

### main.cpp

### chifoumivue.ui

### Expliquer pourquoi on a pas utiliser de repectoire qrc (comme indiquer dans les consigne Nous avons elaborer l’interface graphique seule avec QDesigner, l’ajout des imaes a été fait directment sur QDesigner )

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

### Explication de l’implementation des signals/slot Tableau (

### Signal : click

### Slot : DemanderNouvellePartie

### Signal : click

### Slot : Jouer (Changement de l’image figureJoueur en bouton selon le choix du joueur)

### Je n’oublie aussi de mettre à jour le label en numero selon le sujet d’exam

### 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

Comportement fonctionnel du programme lors de la réalisation des tests

*--Etat des éléments graphiques lorsque le jeu d’état du programme est dans l’état « Etat initial »*

*Capture*

*Explication de l’incativité des bouton avant le declenchement du slot nouvele partie*

*--Etat des éléments graphiques lorsque le jeu d’état du programme est dans l’état « Partie en Cours »*

*Capture*

*Explicaiton de l’activité des bouton apres le declenchement du slot nouvelle partie*

Comportement de l’interface (non lié aux aspects fonctionnels du programme) lors de la réalisation des tests

*--Etat des éléments graphiques avant le redimensionnement*

*Ma Capture*

*Et explication (copié coller de la prof)*

*--Etat des éléments graphiques lors du redimensionnement*

*Ma Capture*

*Explication*