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.

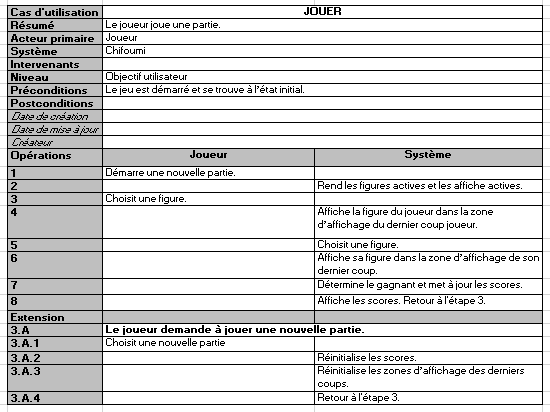
## Diagramme des Cas d’Utilisation

## 

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

## Scénarios

1. **Exemple Scénario**

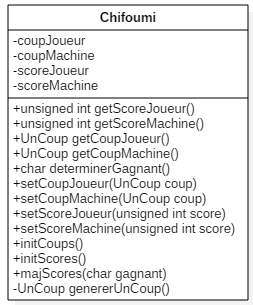


*Tableau 1 : Scénario nominal*

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 : c’est l’interface de la classe chifoumi

### - chifoumi.cpp : c’est le corps de la classe

-main.cpp : C’est le jeu et les tests de la classe

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

### 

### 5.2 Test

Test avec le programme fourni main.cpp

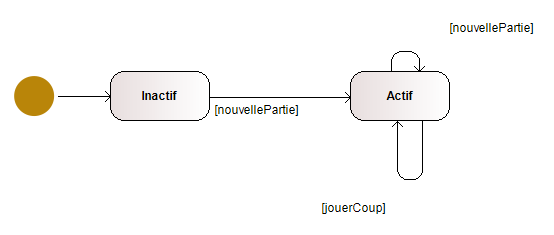
*Valeurs fournies / attendues… comme montré dans la ressource R2.03 (partie tests)*

| ***Méthode*** | *Description* | *donnée en entrée* | *résultat attendu* | *résultat fournies* | *commentaire* |
| --- | --- | --- | --- | --- | --- |
| *getCoupJoueur****()*** | ***retourne le dernier coup joué par le joueur*** |  | ***Rien*** | ***Rien*** | ***ok*** |
| *getCoupMachine****()*** | ***retourne le dernier coup joué par la machine*** |  | ***Rien*** | ***Rien*** | ***ok*** |
| *getScoreJoueur****()*** | ***retourne le score du joueur*** |  | ***0*** | ***0*** | ***ok*** |
| *getScoreMachine****()*** | ***retourne le score de la machine*** |  | ***0*** | ***0*** | ***ok*** |
| *determinerGagnant****()*** | ***détermine le gagnant 'J' pour joueur, 'M' pour machine, 'N' pour match nul en fonction du dernier coup joué par chacun d'eux*** | ***coupJoueur(papier)***  ***coupMachine(papier)*** | ***N*** | ***N*** | ***ok*** |
| *Chifoumi****()*** | ***initialise le score de la machine et du joueur à 0 et initialise le coup du joueur et de la machine à rien*** |  | ***scores=0***  ***coups = rien*** | ***scores = 0***  ***coups =rien*** | ***ok*** |
| *genererUnCoup****()*** | ***retourne une valeur aléatoire est égal soit à pierre soit à papier ou à ciseau*** |  | ***ciseau*** | ***ciseau*** | ***ok*** |
| *setCoupJoueur****()*** | ***initialise l'attribut coupJoueur avec la valeur***  ***du paramétre p\_coup*** | *UnCoup p\_coup* | ***pierre*** | ***pierre*** | ***ok*** |
| *setCoupMachine()* | ***initialise l'attribut coupMachine avec la valeur***  ***du paramétre p\_coup*** | *UnCoup p\_coup* | ***ciseau*** | ***ciseau*** | ***ok*** |
| *initScores()* | ***initialise à 0 les attributs scoreJoueur et scoreMachine*** |  | ***scoreJoueur = 0***  ***scoreMachine = 0*** | ***scoreJoueur = 0***  ***scoreMachine = 0*** | ***ok*** |
| *initCoups()* | ***initialise à rien les attributs coupJoueur et coupMachine*** |  | ***coupJoueur =***  ***rien***  ***coupMachine = 0*** |  | ***ok*** |
| *setScoreJoueur()* | ***initialise l'attribut scoreJoueur avec la valeur du paramètre p\_score*** | *unsigned int p\_score* | ***scoreJoueur =1*** | ***scoreJoueur = 1*** | ***ok*** |
| *setScoreMachine()* | ***initialise l'attribut scoreMachine avec la valeur du paramètre p\_score*** | *unsigned int p\_score* | ***scoreMachine = 2*** | ***scoreMachine = 2*** | ***ok*** |
| *Joueur: ciseau*  *Machine: ciseau* |  |  | ***scoreJoueur = 0***  ***scoreMachine = 0***  ***CoupJoueur = ciseau***  ***CoupMachine = ciseau*** | ***scoreJoueur = 0***  ***scoreMachine = 0***  ***CoupJoueur = ciseau***  ***CoupMachine = ciseau*** | ***ok*** |
| *Joueur: ciseau*  *Machine: papier* |  |  | ***scoreJoueur = 1***  ***scoreMachine = 0***  ***CoupJoueur = ciseau***  ***CoupMachine = papier*** | ***scoreJoueur = 1***  ***scoreMachine = 0***  ***CoupJoueur = ciseau***  ***CoupMachine = papier*** | ***ok*** |
| *Joueur: ciseau*  *Machine: pierre* |  |  | ***scoreJoueur = 0***  ***scoreMachine = 1***  ***CoupJoueur = ciseau***  ***CoupMachine = pierre*** | ***scoreJoueur = 0***  ***scoreMachine = 1***  ***CoupJoueur = ciseau***  ***CoupMachine = pierre*** | ***ok*** |
| *Joueur: papier*  *Machine: papier* |  |  | ***scoreJoueur = 0***  ***scoreMachine = 0***  ***CoupJoueur = papier***  ***CoupMachine = papier*** | ***scoreJoueur = 0***  ***scoreMachine = 0***  ***CoupJoueur = papier***  ***CoupMachine = papier*** | ***ok*** |
| *Joueur: papier*  *Machine: ciseau* |  |  | ***scoreJoueur = 0***  ***scoreMachine = 1***  ***CoupJoueur = papier***  ***CoupMachine = ciseau*** | ***scoreJoueur = 0***  ***scoreMachine = 1 CoupJoueur = papier***  ***CoupMachine = ciseau*** | ***ok*** |
| *Joueur: papier*  *Machine: pierre* |  |  | ***scoreJoueur = 1***  ***scoreMachine = 0***  ***CoupJoueur = papier***  ***CoupMachine = pierre*** | ***scoreJoueur = 1***  ***scoreMachine = 0***  ***CoupJoueur = papier***  ***CoupMachine = pierre*** | ***ok*** |
| *Joueur: pierre*  *Machine: pierre* |  |  | ***scoreJoueur = 0***  ***scoreMachine = 0***  ***CoupJoueur = pierre***  ***CoupMachine = pierre*** | ***scoreJoueur = 0***  ***scoreMachine = 0***  ***CoupJoueur = pierre***  ***CoupMachine = pierre*** | ***ok*** |
| *Joueur:pierre*  *Machine: ciseau* |  |  | ***scoreJoueur = 1***  ***scoreMachine = 0***  ***CoupJoueur = pierre***  ***CoupMachine = ciseau*** | ***scoreJoueur = 1***  ***scoreMachine = 0***  ***CoupJoueur = pierre***  ***CoupMachine = ciseau*** | ***ok*** |
| *Joueur: pierre*  *Machine: papier* |  |  | ***scoreJoueur = 0***  ***scoreMachine = 1***  ***CoupJoueur = pierre***  ***CoupMachine = papier*** | ***scoreJoueur = 0***  ***scoreMachine = 1***  ***CoupJoueur = pierre***  ***CoupMachine = papier*** | ***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* |
| --- | --- |
| 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**

| *nomEvénement* | *Signification* |
| --- | --- |
| NouvellePartie | C’est quand le joueur demande soit à jouer une partie |
| JouerCoup | C’est quand le joueur clique sur son choix |

*Tableau 3 : Evénements 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*

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* | JouerCoup | NouvellePartie |
| --- | --- | --- |
| Inactif | Bouton “nouvelle partie” | X |
| Actif | bouton ciseau/pierre/papier | Bouton “nouvelle partie” |

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

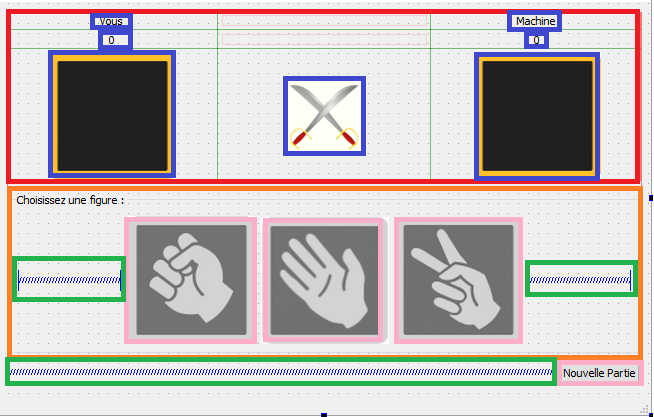
*L’intérêt de cette vue matricielle est qu’elle permet une préparation naturelle et aisée de l’étape suivante de programmation.*

## 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



Cadre rouge: GridLayout

Cadre Bleu: Label

Cadre Orange: GroupBox

Cadre Rose: PushButton

Cadre Vert: Spacer

### 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.

### 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

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.

### 8.2 Test

### 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 | lCoupJoueur = image de pierre | lCoupJoueur = image de pierre | ok |
| choix du joueur : feuille | le joueur appuie sur le bouton feuille | bFeuille | lCoupJoueur = image de feuille | lCoupJoueur = image de feuille | ok |
| choix du joueur : ciseaux | le joueur appuie sur le bouton ciseaux | bCiseaux | lCoupJoueur =  image de ciseaux | lCoupJoueur =  image de ciseaux | ok |
| lancer partie | le joueur appuie sur le bouton NouvellePartie | bNouvellePartie | scoreMachine = 0  scoreJoueur = 0  lCoupJoueur = image de rien  lCoupMachine = image rien  les boutons bPierre,bFeuille et bCiseau sont maintenant actif | scoreMachine = 0  scoreJoueur = 0  lCoupJoueur = image de rien  lCoupMachine = image rien  les boutons bPierre,bFeuille et bCiseau sont maintenant actif | ok |
| 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.

### Décrire les tests prévus / réalisés pour montrer :

### Le comportement fonctionnel du programme

### Le comportement de l’interface non lié aux aspects fonctionnels du programme

# Version v2

Implémentation: liste des fichiers

-chifoumimodele.h: C’est le fichier qui va contenir toutes les déclarations des méthodes pour manipuler les données du chifoumi.

-chifoumimodele.cpp:C’est le fichier qui va contenir toutes les définitions des méthodes du chifoumimodele.h.

-chifoumivue.h: C’est le fichier qui va contenir toutes les déclarations des méthodes pour gérer tout ce qui est l’affichage de l’application.

-chifoumivue.cpp: C’est le fichier qui va contenir toutes les définitions des méthodes du chifoumivue.h

-chifoumipresentation.h: C’est le fichier qui va contenir toutes les déclarations des méthodes pour centraliser la vue et le modèle.

-chifoumipresentation.cpp:C’est le fichier qui va contenir toutes les définitions des méthodes du chifoumipresentation.h

# Version v3

3.- Présentation des seuls fichiers .h modifiés par la mise en œuvre de la v3

chifoumivue.h:

-Ajout de la méthode majInterfaceScore(unsigned int p\_scoreJoueur, unsigned int p\_scoreMachine) qui permet d’actualiser graphiquement les scores .

-Ajout de la méthode majInterfaceCoups( ChifoumiModele::UnCoup p\_coupJoueur, ChifoumiModele::UnCoup p\_scoreMachine) qui permet d’actualiser graphiquement les coups jouer.

-Ajout de la méthode majInterfaceGlobale(ChifoumiModele::UnEtat e, ChifoumiModele::UnCoup p\_coupJoueur, ChifoumiModele::UnCoup p\_coupMachine, unsigned int p\_scoreJoueur, unsigned int p\_scoreMachine) qui actualise tout l’affichage.

chifoumipresentation.h:

-Ajout de la méthode Apropos() qui va permettre d’afficher la description de l’application dans un QMessageBox

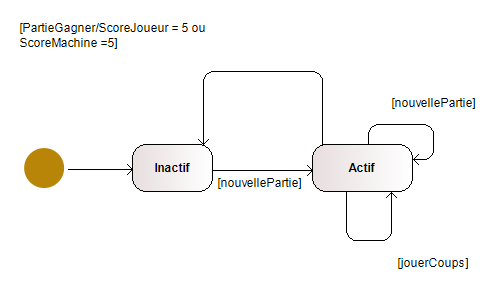
5.- Résultats des tests réalisés

| Méthode | Description | Variable | Résultat attendu | Résultat fournie | Commentaire |
| --- | --- | --- | --- | --- | --- |
| quitter | permet de quitter l'application |  | l’application se ferme | l’application se ferme | OK |
| aPropos | permet de voir la description de l'application |  | un message apparaît ou on voit la description de l’application | Un message apparaît ou on voit la description de l’application | OK |

# Version v4

1.- Diagramme état-transitions de cette version du jeu

(mettre en évidence les changements)



il y a maintenant une passerelle de actif vers inactif quand le score du joueur ou de la machine est égal à 5

**Dictionnaire** **des états du jeu**

| *nomEtat* | *Signification* |
| --- | --- |
| 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**

| *nomEvénement* | *Signification* |
| --- | --- |
| NouvellePartie | C’est quand le joueur demande soit à jouer une partie |
| JouerCoup | C’est quand le joueur clique sur son choix |
| PartieGagner | C’est quand la machine ou le joueur gagne la partie |

*Tableau 3 : Evénements 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 |
| MachineGagne | Le score de la machine atteint 5 |
| JoueurGagne | Le score du joueur atteint 5 |

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

1. **Préparation au codage :**

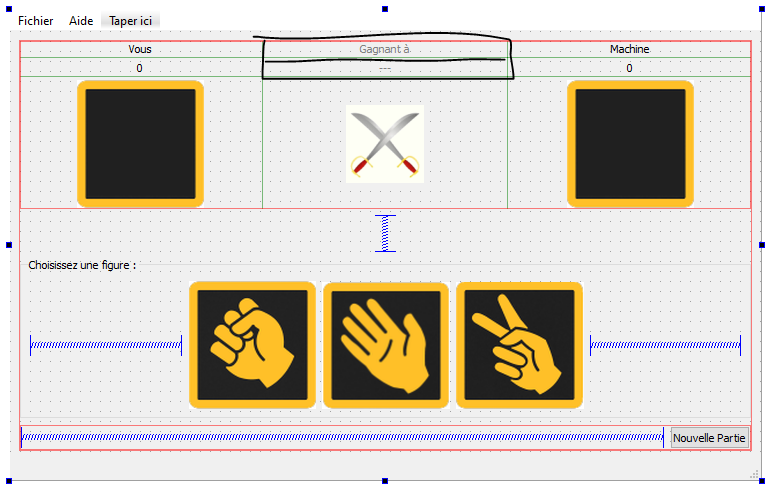
**Table T\_EtatsEvenementsJeu** correspondant à la version matricielle du diagramme états-transitions du jeu :

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

| *Événement 🡪*  *nomEtatJeu* | JouerCoup | PartieGagner | NouvellePartie |
| --- | --- | --- | --- |
| Inactif | Bouton “nouvelle partie” | X | X |
| Actif | bouton ciseau/pierre/papier | ScoreMachine= 5  ou  ScoreJoueur = 5 | Bouton “nouvelle partie” |

(mettre en évidence les changements)

5.- Décrire les nouveaux éléments d'interface



Les deux labels dans les deux cadres noir ont été rajouté pour permettre d’afficher le nombre de point nécessaire pour gagner.

9.- Présentation des seuls fichiers .h modifiés par la mise en œuvre de la v4

chifoumimodele.h:

* Ajout de determinerObjectifAtteint() qui permet de savoir qui a atteint le score pour gagner.
* Ajout de setScoreObjectif(unsigned int p\_score = 5) qui permet d’initialiser le nombre de points qu’il faut pour gagner.

chifoumipresentation.h:

-Ajout de FinDePartie(char p\_gagnant) qui permet de demande l’affichage du message de fin de la partie.

chifoumivue.h:

-Ajout de afficherMsBFinDePartie(QString p\_nomGagnant, unsigned int p\_scoreGagnant) qui permet de créer et d’afficher le message de fin de partie.

-Ajout de majInterfaceScoreObjectif(unsigned int p\_scoreObjectif) qui permet la mise à jour graphique du nombre de points à atteindre pour gagner.

10.- Si pertinent, explications sur des points importants à savoir concernant les .cpp

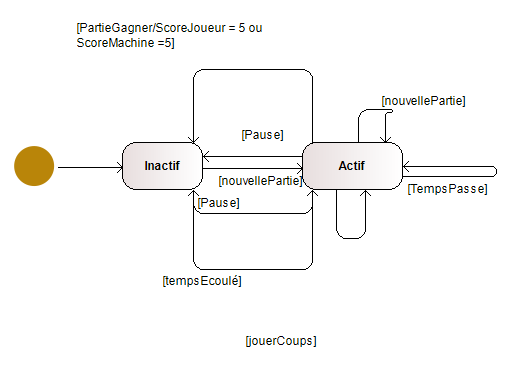
11.- Résultats des tests réalisés

| Méthode | Description | Variable | Résultat attendu | Résultat fournie | Commentaire |
| --- | --- | --- | --- | --- | --- |
| determiner le gagnant de la partie | permet de savoir qui a gagner la partie entre le joueur et la machine |  | J | J | OK |
| afficher le message de fin de partie | permet d’afficher le message de fin partie avec le nom du gagnant | p\_nomGagnant et p\_scoreObjectif | Bravo Joueur ! Vous gagnez avec 5 points. | Bravo Joueur ! Vous gagnez avec 5 points. | Ok |

# Version v5

1.- Diagramme état-transitions de cette version du jeu

(mettre en évidence les changements)



2.- Dictionnaires états, événements, actions associés

(mettre en évidence les changements)

| *nomEtat* | *Signification* |
| --- | --- |
| 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**

| *nomEvénement* | *Signification* |
| --- | --- |
| NouvellePartie | C’est quand le joueur demande soit à jouer une partie |
| JouerCoup | C’est quand le joueur clique sur son choix |
| TempsEcoulé | C’est quand le timer arrive à 0 |
| PartieGagner | C’est quand la machine ou le joueur gagne la partie |
| TempsPasse | C’est quand le timer diminue de 1 |
| Pause | c’est quand le timer est arrêté |

*Tableau 3 : Evénements 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 |
| MachineGagne | Le score de la machine atteint 5 |
| Diminution du temps du timer | La valeur du timer diminue de 1 |
| JoueurGagne | Le score du joueur atteint 5 |
| mettre en pause | le joueur clique sur le bouton pause |

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

3.- Version matricielle du diagramme états-transitions + identification des éléments d'interface supplémentaires éventuellement nécessaires

(mettre en évidence les changements)

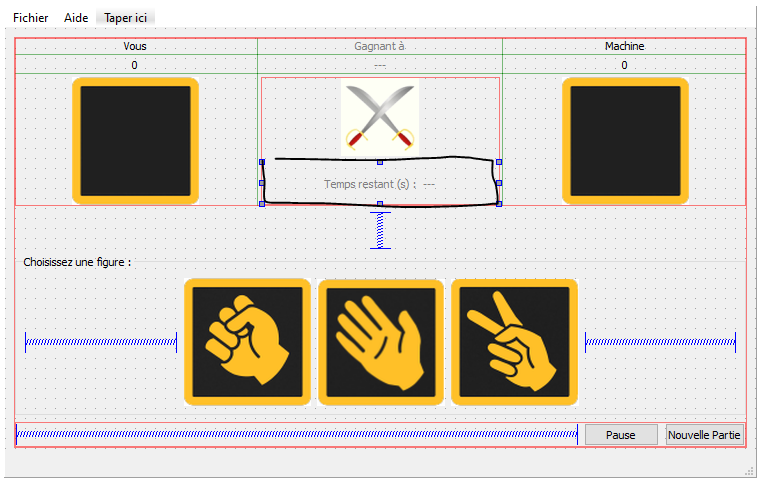
1. **Préparation au codage :**

**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* | JouerCoup | PartieGagner | TempsPasse | TempsEcoulé | mettre en pause | NouvellePartie |
| --- | --- | --- | --- | --- | --- | --- |
| Inactif | Bouton “nouvelle partie” | X | Diminution du temps du timer | X | bouton pause | X |
| Actif | bouton ciseau/pierre/papier | ScoreMachine= 5  ou  ScoreJoueur = 5 | X | Diminution du temps du timer | bouton pause | Bouton “nouvelle partie” |

5.- Décrire les nouveaux éléments d'interface



Ce label dans le cadre en noir permet d’affiché le temps qui reste

9.- Présentation des seuls fichiers .h modifiés par la mise en œuvre de la v5

chifoumimodele.h:

-Ajout de getDureePartie() permet de retourner la durée max de la partie

-Ajout de getTempsRestant() permet de retourner le temps restant avant la fin de la partie

-Ajout de getNomJoueur() permet de retourner le nom du joueur

-Ajout de setDureePartie(unsigned int p\_temps = 30) permet d'initialiser l'attribut dureePartie avec la valeur du paramètre p\_temps

-Ajout de setTempsRestant(unsigned int p\_temps) permet d’ initialiser l'attribut tempsRestant avec la valeur du paramètre p\_temps

-Ajout de setNomJoueur(QString p\_nom= "Vous") permet d’ initialiser l'attribut nomJoueur avec la valeur du paramètre p\_nom

chifoumipresentation.h:

-Ajout de finDePartieTemps() permet demander à afficher la QMessageBox décrivant la fin de partie par temps écoulé

-Ajout de gererPause() permet de s'occuper des éléments liés à la pause

-Ajout de lancerTimer() permet de lancer le timer pour une durée de 1s

-Ajout de stopperTimer( )permet d’arrêter le timer

-Ajout de gererTimer() permet de s'occuper des éléments liés au timer

chifoumivue.h:

-Ajout de afficherMsBFinDePartieTemps(char resultat) permet de créer et afficher la QMessageBox MsBFinDePartie contenant le message final

-Ajout de majInterfaceScoreObjectif(unsigned int p\_scoreObjectif) permet d’actualiser l'objectif de point dans l'ui

-Ajout de majBoutonPause(ChifoumiModele::UnEtat e) permet d'actualiser le bouton de pause

-Ajout de majTempsRestant(unsigned int p\_tempsRestant) permet d’actualiser le temps restant dans l'ui

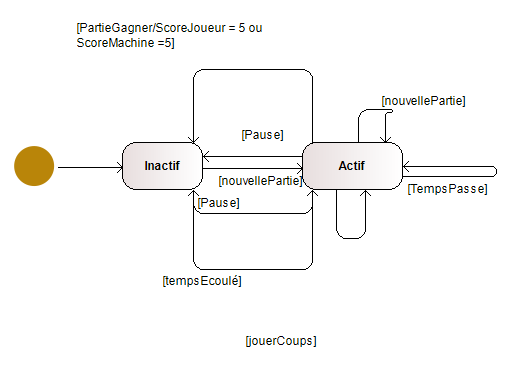
10.- Si pertinent, explications sur des points importants à savoir concernant les .cpp

11.- Résultats des tests réalisés

| Méthode | Description | Variable | Résultat attendu | Résultat fournie | Commentaire |
| --- | --- | --- | --- | --- | --- |
| getDureePartie() | retourne la durée max de la partie |  | 30 | 30 | OK |
| getTempsRestant() | retourne le temps restant avant la fin de la partie |  | 25 | 25 | OK |
| getNomJoueur() | retourne le nom du joueur |  | Vous | Vous | OK |
| setDureePartie() | initialise l'attribut dureePartie avec la valeur du paramètre p\_temps | unsigned int p\_temps = 30 | 30 | 30 | OK |
| setTempsRestant() | initialise l'attribut tempsRestant avec la valeur du paramètre p\_temps | unsigned int p\_temps | 25 | 25 | OK |
| setNomJoueur() | initialise l'attribut nomJoueur avec la valeur du paramètre p\_nom | QString p\_nom= "Vous" | Vous | Vous | OK |
| gererPause() | s’occupe des éléments liés à la pause |  | tous les boutons sauf pause sont inutilisable | tous les boutons sauf pause sont inutilisable | OK |
| afficherMsBFinDePartieTemps() | créé et affiche la QMessageBox MsBFinDePartie contenant le message final | char resultat | affiche le message finale | affiche le message finale | OK |

# Version v6

1.- Diagramme état-transitions de cette version du jeu



2.- Dictionnaires états, événements, actions associés

| *nomEtat* | *Signification* |
| --- | --- |
| 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**

| *nomEvénement* | *Signification* |
| --- | --- |
| NouvellePartie | C’est quand le joueur demande soit à jouer une partie |
| JouerCoup | C’est quand le joueur clique sur son choix |
| TempsEcoulé | C’est quand le timer arrive à 0 |
| PartieGagner | C’est quand la machine ou le joueur gagne la partie |
| TempsPasse | C’est quand le timer diminue de 1 |
| Pause | c’est quand le timer est arrêté |

*Tableau 3 : Evénements 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 |
| MachineGagne | Le score de la machine atteint 5 |
| Diminution du temps du timer | La valeur du timer diminue de 1 |
| JoueurGagne | Le score du joueur atteint 5 |
| mettre en pause | le joueur clique sur le bouton pause |
| parametre la partie | initialise le temps de la partie, les points nécessaire pour gagner et le nom du joueur |

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

3.- Version matricielle du diagramme états-transitions + identification des éléments d'interface supplémentaires éventuellement nécessaires

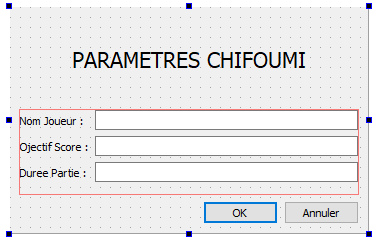
1. **Préparation au codage :**

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

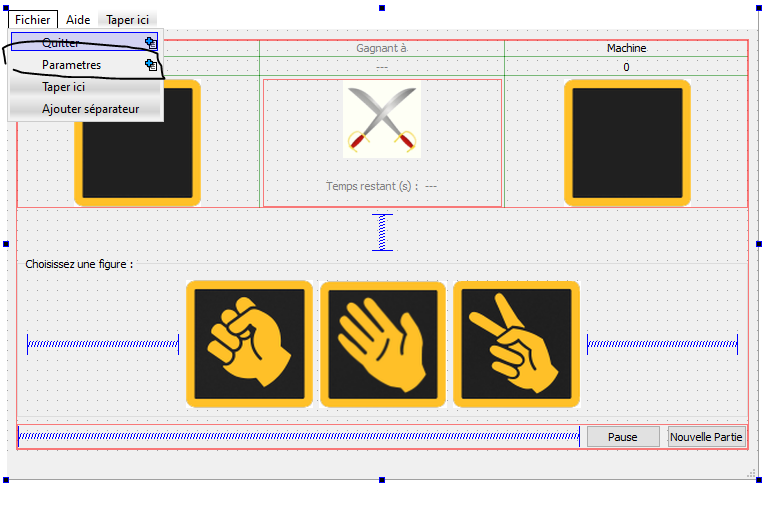
* en *ligne : les* ***événements*** faisant changer le jeu d’état
* en *colonne : les* ***états*** du jeu

| *Événement 🡪*  *nomEtatJeu* | JouerCoup | PartieGagner | TempsPasse | TempsEcoulé | mettre en pause | parametre la partie | NouvellePartie |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Inactif | Bouton “nouvelle partie” | X | Diminution du temps du timer | X | bouton pause | X | X |
| Actif | bouton ciseau/pierre/papier | ScoreMachine= 5  ou  ScoreJoueur = 5 | X | Diminution du temps du timer | bouton pause | parametre | Bouton “nouvelle partie” |

5.- Décrire les nouveaux éléments d'interface



On a rajouté un Qdialog pour paramétrer la partie



On a ajouté la fonctionnalité Parametres dans le menu fichier qui permet d’aller sur la page parametres.ui

8.- Liste des fichiers sources de cette version (et rôle de chacun)

parametres.h: Ce fichier permet de déclarer la classe parametres qui permet d’afficher et de gérer une fenêtre de dialogue.

parametres.cpp:Ce fichier permet de définir les méthodes de la classe parametres.

9.- Présentation des seuls fichiers .h modifiés par la mise en œuvre de la v6

parametres.h:

-Ajout de getLineEditNomJoueur() permet de retourner le contenu du line edit eNomJoueur

-Ajout de getLineEditObjectifScore() permet de retourner le contenu du line edit eObjectifScore

-Ajout de getLineEditDureePartie() permet de retourner le contenu du line edit eDureePartie

chifoumipresentation.h:

-Ajout de ouvrirParametres() permet d’ouvrir la fenetre QDialog parametre

chifoumivue.h:

-Ajout de majnomJoueur(QString p\_nom) permet de changer le nom du joueur

11.- Résultats des tests réalisés

| Méthode | Description | Variable | Résultat attendu | Résultat fournie | Commentaire |
| --- | --- | --- | --- | --- | --- |
| getLineEditNomJoueur() | retourne le contenu du line edit eNomJoueur |  | dieu | dieu | OK |
| getLineEditObjectifScore() | retourne le contenu du line edit eObjectifScore |  | 100 | 100 | OK |
| getLineEditDureePartie() | retourne le contenu du line edit eDureePartie |  | 10 | 10 | OK |
| majnomJoueur() | change le nom du joueur | QString p\_nom | dieu | dieu | OK |