# Réalisation D'un Tamagotchi

HERNANDEZ Dylan
MAGHNOUJI Zakaria
MISSICHINI Julien
License 3

# SOMMAIRE

# Equipe de Projet

- HERNANDEZ Dylan : Chef de projet
- MAGHNOUJI Zakaria
- MISSICHINI Julien

# Cahier des Charges

## Description du Projet :

Le projet a réalisé consiste à développer un jeu qui simule la vie d'un petit animal virtuel : un tamagotchi. L'utilisateur devra s'occuper de lui régulièrement pour qu'il puisse survivre dans de bonnes conditions. Il pourra notamment intervenir sur son environnement.
Ce projet sera réalisé par notre groupe de 3 étudiants dans son ensemble (cahier des charges, méthodes de développement, spécifications techniques, modélisation UML, application...)

## Implémentation:

Notre application sera implémenté en C++ et pour l'interface graphique, on utilisera les bibliothèques SDL et GTK

## Outils et Ressources :

Pour réaliser ce projet, nous allons utilisé plusieurs outils et ressources que l'on va décrire si dessous :

Tout d'abord les outils nécessaires au développement de ce projet sont :

- un editeur de texte
- un compilateur c++
- un serveur de gestions des versions du projet

Ces outils ne pourront être réalisé sans les outils suivants :

- CodeBlocks IDE

Et au niveau des ressources nous allons utilisé :

- la plate-forme Github qui gère les versions du projet et permet de partager nos fichiers.

## Aperçu des fonctionnalités du Tamagotchi :

. Inteface Graphique de type fenêtre
. Choix du Tamagotchi
    - Nom

. Interaction avec le joueur
- Remplir les gamelles (eau et nourriture)
- S'amuser avec
- Le Laver

. Fonction automatique
- Dormir si barre fatigue à 0
- Mange si assez joyeux et si les gamelles sont remplis

## Style de Programmation et Gestion de Versions

Les noms des fichiers, écrit en français, seront représentatifs du contenu du fichier.
Pour ce qui est du contenu, les fonctions et variables seront aussi écrites en français.
Les commentaires, si il y en a, seront aussi rédigé en français.
Pour ce qui est de la gestion des versions, chaque membre du projet synchronisera son travail avant et a la fin de sa journée de travail sur la plate-forme Github, ce qui permettra à chaque membre de voir l'avancement du projet et de se mettre à jour.

## Protocole de Test

Pour ce qui concerne les tests, ils seront réalisés par deux membres de l'equipe. Le Chef de projet se chargera de la compilation et l'implémentation des différents modules.

L'application lorsqu'elle sera terminé ne pourra être validé qu'après avoir rempli les conditions suivantes :

- être compilable sur Windows et Ubuntu.
- Que l'application tourne sans interruption pendant au moins 10 minutes.

Un makefile a aussi été crée pour compiler l'application.

# Spécifications Techniques

## Modélisation UML(diagramme)

Les diagrammes ont été réalisé en début de projet, mais aussi modifié au cours du projet pour un meilleure compréhension et une meilleure conformité par rapport au code. Nous avons réalisé au total 4 diagrammes : Le diagramme d'utilisation, le diagramme de classes, le diagramme d’état-transition et le diagramme d'activité

Diagramme d'utilisation :

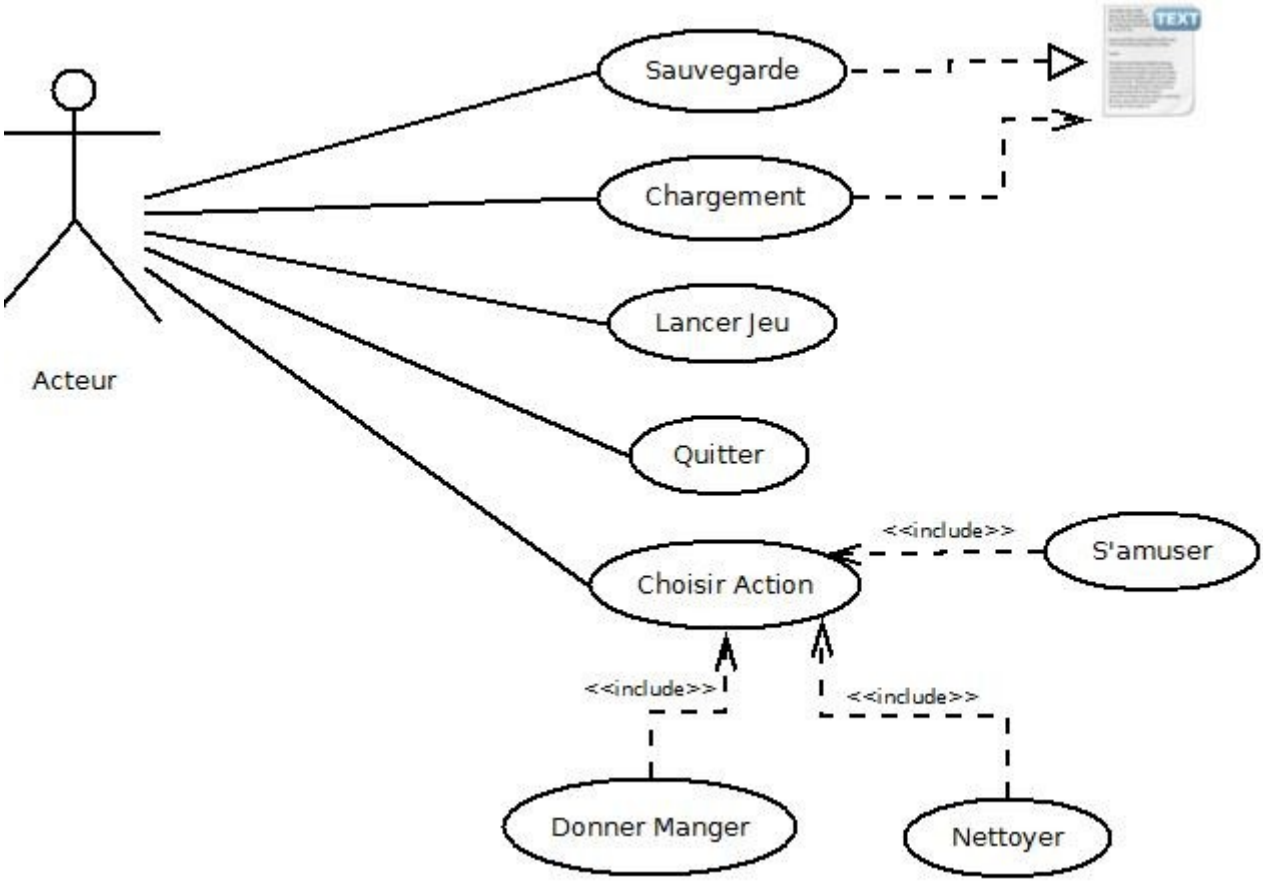


Diagramme de Classes :

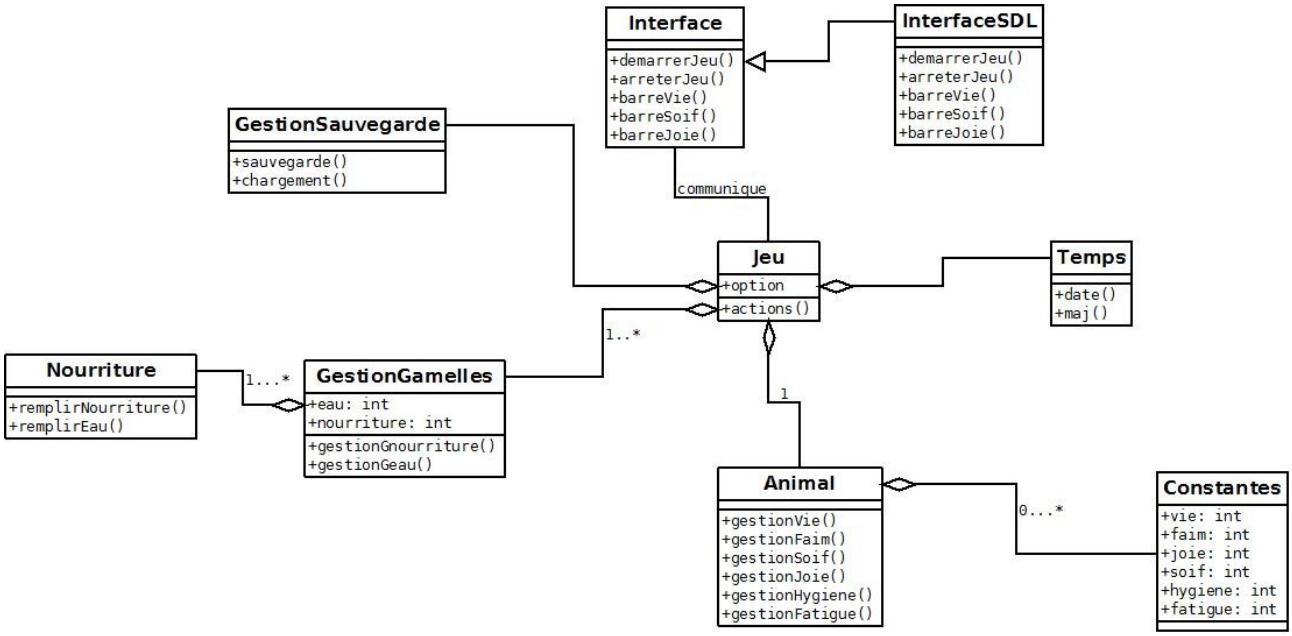



Diagramme d'état-transition :

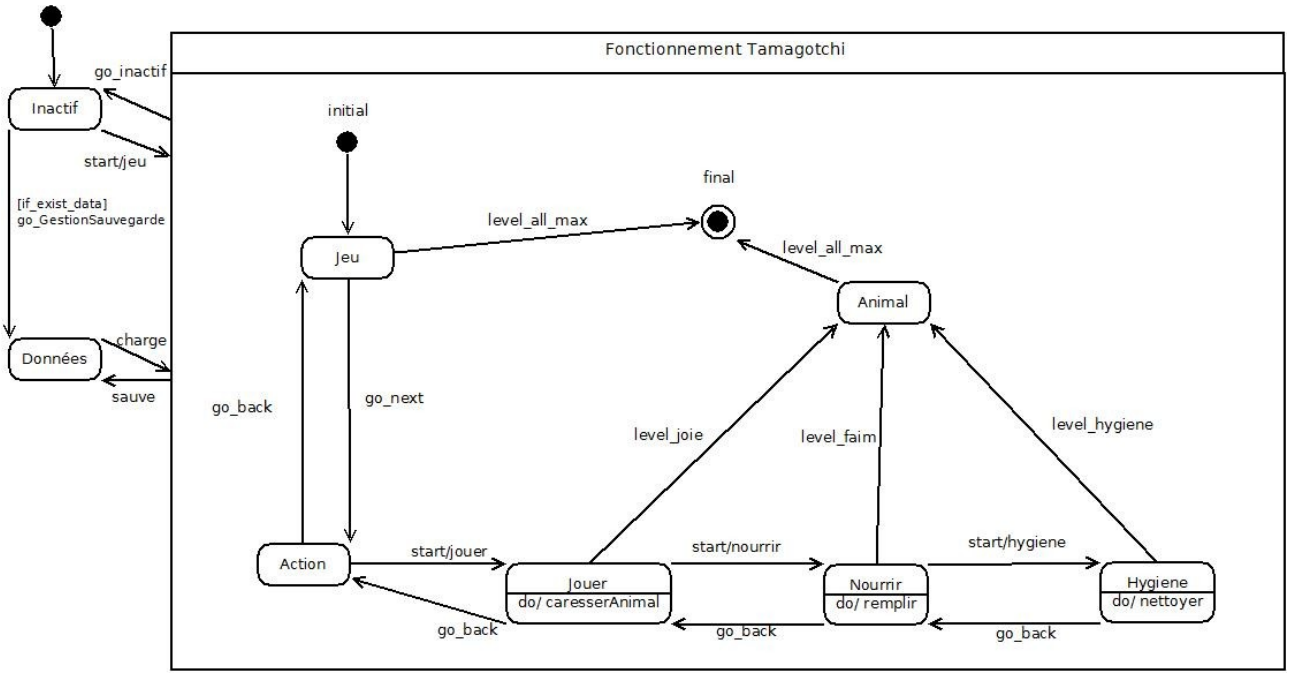
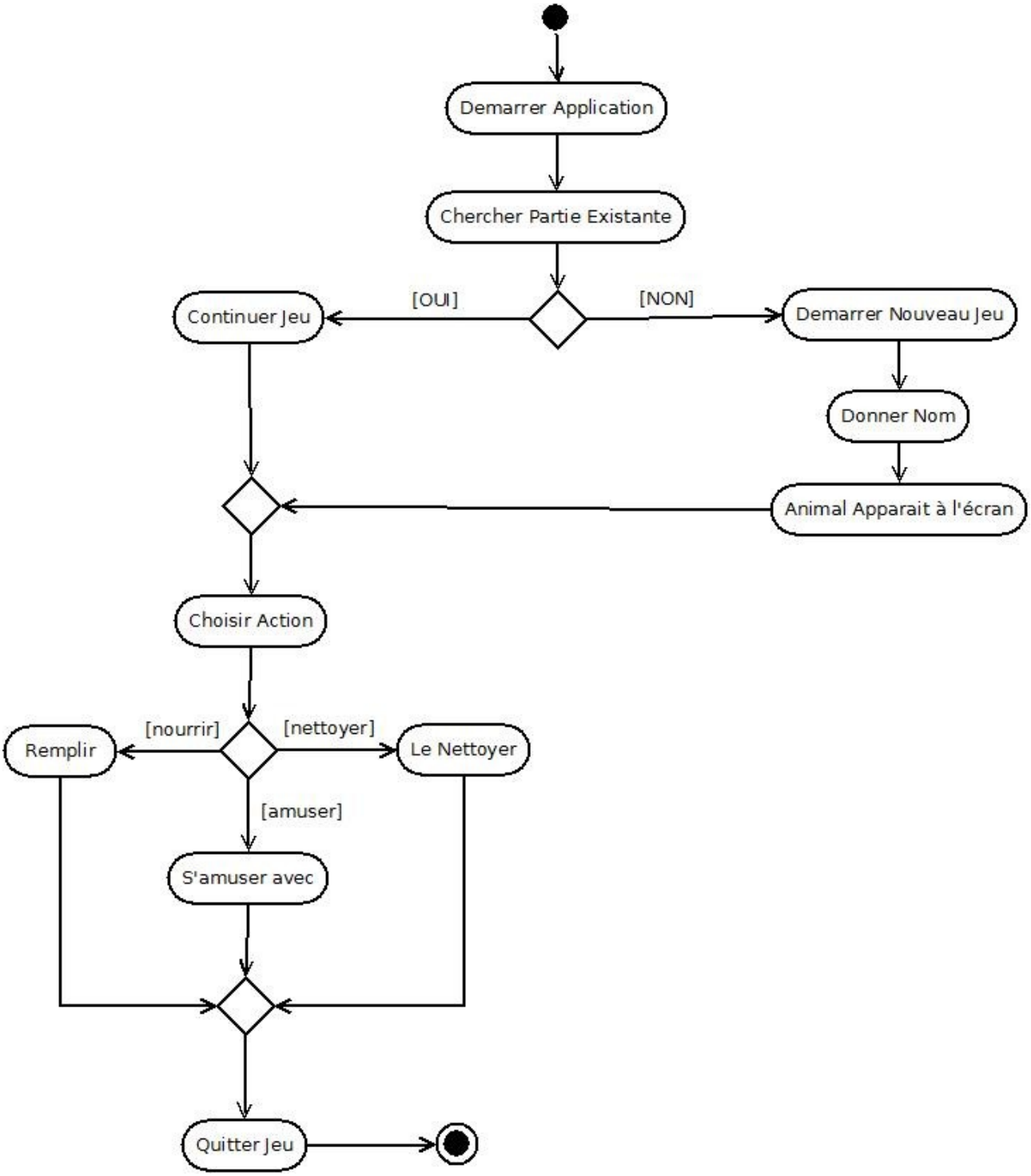


Diagramme d'activité :



# Description du code :

Lorsque nous avons commencer ce projet notre objectif était de suivre le mieux possible notre modélisation UML.

Dans cette modélisation nous avons divisé notre programme en 9 fichiers. 2 headers et 7 fichiers sources.

Le fichiers Nourriture.h contient 2 classes (Nourriture et Constante) contenant la déclaration de toutes les variables et de toutes les fonctions entrant en compte dans la gestion des gamelles et le fichier Constante.h contient la déclaration de toutes les variables et toutes les fonctions utilisé pour gérer les constantes du tamagotchi.

Le fichiers GestionGamelles.cpp contient deux fonctions et le constructeurs et le destructeur de la classe Nourriture. Ces fonctions servent à remplir les gamelles d'eau et de nourriture .

Le fichier Animal.cpp contient toutes les fonctions intervenant dans les gestions des constantes du tamagotchi et le constructeur et le destructeur de la classe Constante:

- giveName : servira à donner un nom au tamagotchi
- gestionVie : cette fonction servira à gérer la barre de vie du tamagotchi
- gestionFin : cette fonction servira à gérer la barre de faim
- gestionSoif : cette fonction servira à gérer la barre de soif
- gestionHygiene : cette fonction servira à gérer la barre d'hygiène
- gestionFatigue : cette fonction servira à gérer la barre de fatigue
- caresserAnimal : cette fonction servira à caresser l'animal pour augmenter la joie
- donnerFriandise : cette fonction remonte la barre de faim et la joie
- chrono : cette fonction sert à appeler les fonctions de gestion en fonction du temps d'execution du programme.

Le fichier Temps.cpp contient de fonction. La fonction date servira à convertir une date de string à entier. La fonction MAJ quand elle servira à comparer la date actuel à la date de sauvegarde de la dernière partie pour mettre à jour les constantes.

Il y a dans notre fichier 2 fichiers interface, une interface terminal et une interfaceSDL.  La première interface nous servira surtout pour les phases de tests et si nous n'arrivons pas à programmer l'interface SDL.

C'est 2 fichiers contiennent tous les 2 une fonctions interface qui ouvre une

autre fenêtre afin que le terminal de départ nous servent à rentrer les différentes actions.

Le fichiers GestionSauvegarde.cpp contient 2 fonctions permettant de sauvegarder toutes les informations nécessaire dans un fichier texte et de charger la partie à partir de ce même fichier texte.

Enfin le fichier Jeu.cpp contient le main. Nous utilisons dans ce fichiers des thread pour que plusieurs fonctions puissent fonctionner en même temps ainsi, grâce a un premier thread nous lançons l'interface, grâce à un second le chrono et ensuite nous pouvons entrer dans le premier terminal des actions pour interagir avec le programme.

## Rapport de Tests :

L'application n’étant pas terminé dans les temps, nous n'avons pas pu réalisé pour le moment les tests prévus. Ils seront réalisé dès que l'application sera terminé.