Germain Jr. OLEA-OYOUGOU

Memoire de mise en situation professionnel — Création d'une bibliothèque PHP[1]

[1]IUT RCC
Barbara ROMANIUK

Table des matières

Liste des Tableaux **iii**

Liste des figures **iv**

Avant-propos **v**

Remerciements **1**

Résumé **1**

Introduction **2**

1 Contexte **3**

1.1 Mise en situation professionnelle 3

1.2 Pourquoi faire une bibliothèque en PHP 4

2 Développement de la bibliothèque **6**

2.1 Programme de la mise en situation professionnelle. 6

2.2 Pré-requis avant le développement de la bibliothèque 7

2.3 Conception 8

2.4 Implémentation de la bibliothèque 10

2.5 Distribution de la bibliothèque 11

2.6 Tableau récapitulatif 13

3 Utilisation de la bibliothèque **14**

3.1 Pré-requis à l'utilisation 14

3.2 Exemple d'utilisation et documentations 16

4 Résultat **21**

4.1 Finalité de la bibliothèque 21

4.2 Bilan de la mise en situation professionnelle 21

4.3 Ressenti personnel 22

Conclusion **23**

Liste des Abbreviations **24**

Annexe **26**

Trello **26**

.1 Tableau trello au début du projet 26

.2 Tableau trello du mémoire 26

Documentations **28**

.3 Extrait de Documentation 1 28

.4 Extrait de Documentation 2 28

.5 Extrait de documentation textuelles 28

Liste des Tableaux

2.1 Planning journalier 6
2.2 Récapitulatif des dépendances 13

Liste des figures

2.1 Diagramme de classe : AbstractFFI 9

1 Tableau trello au début du projet 26
2 Tableau trello du mémoire 27

3 Extrait de Documentation 1 28
4 Extrait de Documentation 2 29

Avant-propos

Ce document est un mémoire de mise en situation professionnelle, réalisé dans le cadre de l'obtention de mon DUT Informatique à l'IUT Reims-Châlons-Charleville. Il abordera les moyens de développement d'une bibliothèque graphique en PHP avec l'aide de l'extension FFI, objectif fixé par ma tutrice de projet madame Romaniuk Barbara, maître de conférence en Informatique et enseignante au département Informatique de l'IUT de Reims.

En effet elle a remarqué la lacune de ressources dans le domaine et a voulu dans le même temps ré-implémenter une bibliothèque utilisée par le département informatique pour la réalisation des TPs de Programmation Orientée Objet en Java, inutilisable à cause du passage à PHP.

Mon but est alors de créer en quatre semaines une bibliothèque graphique en PHP assez facile d'utilisation pour qu'un débutant en programmation puisse l'utiliser. Par bibliothèque graphique nous entendons un ensemble de fonctionnalités permettant principalement la création de fenêtres et l'affichage de diverse formes sur cette dernière.

Il est sûr que je rencontrerai des difficultés au cours de ce projet assez ambitieux même pour un étudiant de deuxième année, la principale étant la gestion de l'échange de données entre FFI et la bibliothèque C choisie pour l'implémentation de ces fonctionnalités en PHP. C'est notamment l'une des raisons qui me poussera à voir à la baisse mes ambitions pour cette bibliothèque.

Les sources de ce projet sont disponibles sur Github à l'adresse `https://github.com/djuhnix/phpml` j'en ferai souvent références dans ce mémoire.

Remerciements

En premier lieu je voudrais remercier Madame Romaniuk, en tant que professeure elle m'a enseigné et aidé à mieux comprendre la Programmation Orientée Objets dans la majorité de ses aspects et en tant que tutrice pour ce projet elle m'a orienté dans mon travail, je la remercie pour ses conseils qui m'ont aidé à avancer sur certaines problématiques.

J'aimerais également remercier toute l'équipe pédagogique qui m'a apporté toutes les connaissances que j'ai pu acquérir jusqu'ici et sans oublier tous ceux et toutes celles qui participent à la vie du département informatique de l'IUT de Reims.

Je souhaite particulièrement remercier mes amis et proches, pour leur amour, leurs conseils ainsi que leur soutien sans qui ces deux années passées en DUT ne seraient pas ce qu'elles sont pour moi aujourd'hui.

Résumé

L'objectif de ce mémoire est de montrer comment on peut créer une bibliothèque qui pourra faciliter la découverte de la Programmation Orientée Objet (POO) à des débutants via des travaux pratiques guidés. Il relate les faits de réalisations de mon projet de mise en situation professionnelle. Pour faire très simple, la POO est un moyen de programmer avec une représentation du monde courant avec des "objets algorithmiques". Comme dit plus haut cela est une définition très minime car la programmation orienté objets ne se résume pas à ça. Il n'est donc pas si facile de cerner ce qu'est la programmation orientée objet via une simple définition, c'est pour cela qu'il existe différentes méthodes d'apprentissage et l'IUT de Reims s'oriente vers une pédagogie graphique soutenue par ses enseignant-chercheurs. Cependant le passage à PHP pour l'apprentissage de la POO a changé la manière de réaliser les travaux pratiques pendant la formation. En effet il n'existe pas à ce jour de bibliothèque graphique en PHP pour assurer cette méthode d'enseignement. D'où nous vient la problématique de la création d'une bibliothèque PHP ayant la capacité de faire découvrir simplement la Programmation Orientée Objet par la création d'objets graphiques.

Avant le passage en PHP une bibliothèque Java du département Informatique était utilisée pour permettre aux étudiants de créer une fenêtre et des formes sur cette fenêtre, mais cette pratique a été abandonnée lors de l'adoption du PHP comme langage orienté objet principal, jusqu'à l'arrivée de PHP 7.4. L'arrivée de l'extension FFI (Foreign Function Interface) en PHP 7.4 permet d'importer les fonctionnalités d'une bibliothèque externe au sein de PHP. C'est à dire qu'il est possible d'utiliser des librairies partagées écrite en C, en Go ou encore en Rust, etc. Cela ouvre un large panel de possibilités. L'espoir de pouvoir reprendre les anciennes habitudes d'enseignement renaît.

Avec FFI il est donc possible de créer un bibliothèque graphique en PHP capable avant tout de créer une fenêtre, des formes géométriques et des graphiques à l'intérieur de la fenêtre. SFML (Simple and Fast Media Library) qui est une bibliothèque fournissant une interface vers différents éléments de nos ordinateurs comme le système, le fenêtrage, les graphismes, l'audio et réseau, semble être le meilleur choix pour accomplir cette prouesse jusque là inimaginable. La futur bibliothèque utilisera alors les fonctionnalités qu'offre cette dernière pour aboutir à un simple recueil de classes utilisables par n'importe quel débutant de PHP orienté objet.

Bien que cette bibliothèque utilise CSFML elle n'en sera pas une complète copie, d'abord parce que le temps ne le permet pas et ensuite ce n'est que le module graphique qui nous intéresse ici mais malgré cela nous allons imlémenter qu'une parties de ses fonctions. Par ailleurs FFI est une extension expérimentale et possède par conséquent certaines limites. Malgré cela, son apport dans l'univers de PHP ne peut être remis en question car c'est par elle que l'on peut à présent espérer réaliser un jeu d'échec en PHP qui ne soit ni en web ni en ligne de commande.

Introduction

Apprendre la programmation ne devrait pas être une épreuve, mais plutôt un plaisir. Il est vrai qu'il existe beaucoup de styles de programmation, de la programmation procédurale à la programmation logique en passant par la Programmation Orientée Objet. Chacun peut avoir un avis sur le sujet mais peut importe le style choisie apprendre à programmer devrait être le plus simple possible pour ouvrir ce merveilleux monde au plus grand nombre.

C'est dans cette optique que j'ai accepté de refaire pour mon projet de mise en situation professionnelle la bibliothèque graphique utilisée par l'IUT de Reims pour l'apprentissage de la Programmation Orientée Objet (POO). Avec elle un débutant en programmation apprend en interagissant avec des entités visuelles ou tangibles via un programme préalablement écris par ce dernier. Le principe de la POO est de pouvoir manipuler dans son programme les concepts de la "vraie vie", étant donné que voir le résultat de ses travaux fait partie des meilleurs moyens d'apprentissage il est évident d'allier ces deux visions pour aboutir à un enseignement efficace. Ainsi avoir un ensemble de fonctions qui permettent de créer et manipuler des objets graphiques aplanirait grandement la tâche d'enseignement des professeurs en favorisant l'autonomie des étudiants.

Cependant le développement d'une bibliothèque en générale fait intervenir différentes questions à ne pas prendre à la légère. Il faut s'interroger sur comment la réaliser mais surtout s'il est possible de la réaliser. Par exemple il était inimaginable de développer une application hors environnement web en PHP jusqu'à la sortie de sa version 7.4. En effet l'arrivée de l'extension FFI a ouvert les possibilités du langage car elle emmène avec elle le moyen d'inclure des librairies C directement en PHP sans avoir besoin de préalablement créer des extensions PHP en C, ce qui allège la courbe de développement.

Alors comment pouvons nous créer une bibliothèque PHP C via FFI qui nous donnerais la possibilité de manipuler nos objets abstraits de programmation graphiquement ? C'est la question à la quelle j'ai essayé de répondre tout au long de ma mise en situation professionnelle.

Afin de traiter correctement ce sujet j'ai du établir un plan pour profiter au maximum des 4 semaines qui m'ont été données. Documentation, conception du projet, développement et distribution de celui-ci, sont les principales tâches que je me suis donné de terminer d'ici la fin de ces quatre semaines, et je vais les détailler dans ce mémoire.

Nous verrons dans un premier temps le contexte qui nous a amené à nous interroger sur la création d'une bibliothèque 1, puis nous nous intéresserons au développement 2 et l'utilisation 3 de celle-ci pour terminer par le résultat final de cette période de mise en situation professionnelle et par conséquent l'aboutissement du développement de la bibliothèque 4.

1

Contexte

1.1 Mise en situation professionnelle

Avant d'entrer dans le vif du sujet il est évident de définir le contexte dans lequel je me suis vu développer cette bibliothèque.

1.1.1 Le stage en DUT Informatique

En DUT Informatique on doit **obligatoirement** effectuer un stage de fin d'études en deuxième année. Le but premier de ce stage est d'appliquer les connaissances scolaire "théoriques" apprises dans un contexte professionnel, c'est à dire entouré de personnes qualifiées qui sauront corriger et nous montrer nos erreurs. Ce cadre est nécessaire non seulement pour faire valoir ce que l'on a appris mais est aussi pour certains le début de leur parcours dans le monde de l'emploi. En effet le DUT est une formation professionnalisante, ce qui signifie que les titulaires de ce diplôme ont la possibilité et la facilité d'entrer dans le monde professionnel car par

> "l'acquisition de compétences professionnelles multiples et une solide culture générale, le DUT vise la polyvalence."
>
> — (Onisep, 2019)

Bien que la plupart préfèrent poursuivre leurs études.

1.1.2 Annulation de stage

En ce qui me concerne, mon stage était censé se dérouler à la direction du numérique de l'Université de Reims Champagne Ardenne (URCA). "Censé" car le contexte de ces derniers mois lié à la pandémie du Covid-19 n'a pas permis la tenue de celui-ci. Malheureusement plusieurs stages ont ainsi été annulés parce qu'ils n'ont pas pu se poursuivre même en télétravail. Il a donc fallu que le personnel de l'université et les responsables de formations trouvent

une solution pour palier à cette situation. D'où la **mise en situation professionnelle** dont le but ne s'éloigne pas de celui du stage, cependant la mise en situation professionnelle en DUT Informatique s'apparente plus à un projet à réaliser seul(e), en télétravail, sous le tutorat d'un enseignant. Elle :

- fait confronter l'étudiant à une problématique en rapport avec son choix d'orientation
- permet d'approfondir les connaissances déjà acquises et de découvrir de nouvelles
- ouvre vers de nouvelles méthodes de travail.

Et ce sont *mes* méthodes de travail en particulier que je détaillerai dans la suite. Mais avant il est bon de se pencher sur la raison d'être de ce sujet de mise en situation professionnelle qu'est la réalisation d'une bibliothèque.

1.2 Pourquoi faire une bibliothèque en PHP

Le DUT Informatique est une formation généraliste, on y vois donc, de façon très théorique pour certains, un peu de chaque domaine du numérique. Mais la programmation fait quand même partie des matières principales. Et la Programmation Orientée Objet est le principal paradigme de programmation que l'on apprend.

1.2.1 L'arrière-plan

L'IUT utilisait auparavant le Java comme langage principal pour apprendre la POO, et pour ce faire les enseignants ont choisis une pédagogie orienté graphique permettant aux étudiants une facilité de compréhension et d'adaptation au langage puis à l'orienté objet.

L'objectif premier étant de réussir à manipuler de manière simple et intuitifs les objets d'une classe graphiquement, de telle sorte que ce soit facilement compréhensible par les nouveaux étudiants.

Java était plus adapté et plus simple à mettre en place, une bibliothèque interne à été développé pour la cause. Cependant le Java a montré certaines contraintes, la principale étant la lourdeur syntaxique, le java a donc été abandonné pour un premier contact avec la POO pour laisser place à du PHP typé. Cependant le côté graphique a été délaissé avec le Java vu qu'il n'existait pas de bibliothèques et ni de fonctionnalités similaire en PHP. Il est certes possible de le faire sous certaines contraintes dans un environnement web, mais cela éloignerait l'étudiant de la console, étant donné qu'il apprend et doit s'en servir en TP.

1.2.2 Le problème avec PHP

Le passage à PHP pour l'apprentissage de la POO a changé la manière de réaliser les travaux pratiques à l'IUT de Reims. L'utilisation de scritps écrits par l'étudiant pour créer des figures

et en faire des objets graphiques affichables sur une fenêtre système pour comprendre les notions de classes, d'objets et d'instances était très efficace. Mais PHP est un langage orienté web qui ne permet pas la création de fenêtres graphiques sur le système d'exploitation et par conséquent laisse impossible la manipulation d'éléments sur cette dernière.

L'arrivée de l'extension FFI en PHP 7.4 a ouvert les possibilités du langage, il est désormais envisageable d'utiliser une bibliothèque C pour étendre les limites de ce dernier et apporter les fonctionnalités de la dite bibliothèque en PHP.

1.2.3 Définition de l'objectif

Madame Romaniuk, maître de conférences en Informatique et ma tutrice pour cette mise en situation professionnelle, m'a donc proposé de refaire la bibliothèque Java utilisé pour les TPs de POO en PHP avec l'aide de CSFML et FFI.

CSFML est un pont écrit en C pour la bibliothèque SFML (Simple and Fast Media Libray). Cette dernière contient toutes les fonctionnalités que nous cherchons à implémenter et n'a pas une grande courbe d'apprentissage pour être maîtrisée quand on a un minimum de connaissance de C++. Voilà qui explique le choix de madame Romaniuk pour le développement de la nouvelle bibliothèque.

1.2.4 La pertinence du projets

Avec tout ce qui a été dit jusqu'à présent, il est vrai que la question sur la pertinence de ce sujet ne se pose plus, mais il n'est pas trivial de rappeler que la création de cette bibliothèque ne résoudra pas tous les problèmes **susmentionnés**. Toutefois :

- Bien que les TPs n'en seront pas fortement affecté, il est certains que son existence allégera fortement la tâche pédagogique des professeurs.
- L'ouverture de possibilités pour les projets de fin d'année, car cette bibliothèque pourra être réutiliser et favoriser la créativité des étudiants.

Maintenant que nous savons quelles missions m'ont été confiées et leurs objectifs, on peut parler du déroulement de celles-ci en commençant par l'organisation de mon temps pendant la mise en situation professionnelle.

2

Développement de la bibliothèque

Impossible de parler du développement de la bibliothèque sans aborder le programme que je me suis fixé pour le faire.

2.1 Programme de la mise en situation professionnelle.

La mise en situation professionnelle s'est déroulé dans le cadre du confinement, il m'était donc obligatoire de le faire en télétravail.

2.1.1 Organisation

Dès le début je me suis fait un programme pour répartir convenablement les 4 semaines qui m'étaient données pour la réalisation de la bibliothèque.

- **Semaine 1** : documentation et découverte
- **Semaine 2** : Conception de l'architecture de la bibliothèque et template de développement
- **Semaine 3** : Développement de la bibliothèque
- **Semaine 4** :
  - Poursuite de développement et distribution de la bibliothèque
  - Finalisation du mémoire

En plus de cette organisation hebdomadaire je me suis fixé un planning journalier.

TABLEAU 2.1: Planning journalier

Horaire	Activité
09h00 - 13h00	Projet
Pause	
15h00 - 17h00	Projet
Pause	
18h00 - 20h00	Mémoire

Je me suis également servi de trello qui fut d'une aide précieuse pour mettre en place toute cette organisation.

2.1.2 Impression sur le télétravail

Objectivement j'ai pu respecté ce planning à 70%. Il est évident que travailler de chez soi n'est pas une compétence innée mais bien une compétence qui se pratique et s'améliore avec l'expérience. Il faut bien sûr apprendre à faire l'impasse sur certains divertissement sans pour autant se priver.

2.2 Pré-requis avant le développement de la bibliothèque

Avant de réellement commencer le développement de la bibliothèque il m'a fallu une période de documentation et de découverte qui a durée une semaine. Evidemment j'en avait besoin car avant cette mise en situation professionnel je ne connaissais rien de FFI ou de CSFML. Je vais résumé ce que j'ai pu apprendre en donnant les pré-requis au développement et fonctionnement de la bibliothèque.

2.2.1 PHP FFI

L'extension FFI de PHP, disponible dès la version 7.4, est obligatoire pour développer et faire fonctionner la bibliothèque. Elle est intégrée mais "inactive" par défaut (The PHP Group, 2019) sauf si l'on utilise PHP en ligne de commande ou la fonctionnalité de pré-chargement de PHP en environnement web. Pour activer FFI il faut s'assurer d'avoir la librairie FFI `libffi` installée sur sa machine. Sourceware libffi[1].

2.2.1.1 Installation de `libffi`

- Sur linux (Debian pour moi) la librairie est accessible via le paquet `libffi-dev`. Donc l'exécution de la commande suivante devrait installer les paquets nécessaires au bon fonctionnement de l'extension FFI de PHP.

```
sudo apt install libffi-dev
```

- Sur Windows je n'ai pas exploré le processus d'installation mais il est disponible à l'adresse suivante : Goldencode.com : Building and Installing libffi on Windows[2]

[1] https://sourceware.org/libffi/
[2] https://proj.goldencode.com/projects/p2j/wiki/Building_and_Installing_libffi_on_Windows

2.2.1.2 Configuration/Activation de FFI

Après avoir installé `libffi`, il faut désormais l'activer. On peut le faire de plusieurs manières mais je détaillerai uniquement celle que j'utilise pour faire fonctionner la bibliothèque. Voir 3.1.

2.2.2 CSFML et SFML

CSFML est juste un pont vers la bibliothèque graphique SFML il faut donc installer SFML si l'on veut utiliser CSFML.

- Sur Debian, ces bibliothèques sont accessibles respectivement via les paquets `libsfml-dev` et `libcsfml-dev`

```
sudo apt install libsfml-dev libcsfml-dev
```

- Sur Windows on peut les télécharger sur le site officiel de la bibliothèque sfml.org[3]

Après avoir eu ces dépendances sur ma machine, j'ai pu découvrir comment fonctionnait CSFML et SFML d'abord puis j'ai été capable de réaliser un programme en C qui affiche une fenêtre. Par la suite j'ai également réussi à écrire un script qui ouvre une fenêtre en PHP (accessible sur le dépôt git) utilisant CSFML et FFI fraîchement installé. Maintenant que j'étais au point sur les dépendances de ma future bibliothèque j'ai pu commencé la phase de conception.

2.3 Conception

En ce qui concerne la conception, je n'ai pas l'intention de détailler tout mon processus de réflexion mais juste d'éclaircir sur certains points clés de l'architecture de la bibliothèque. Avant tout il me fallait trouver un moyen de m'abstraire du processus de chargement d'une bibliothèque avec FFI. La première tâche que je me suis donnée lors de ma phase de conception fut alors de réaliser une abstraction à la manipulation d'objets FFI.

2.3.1 Ma première classe AbstractFFI

> En informatique, le concept d'abstraction identifie et regroupe des caractéristiques et

[3] https://www.sfml-dev.org/download.php

> traitements communs applicables à des entités ou concepts variés ; une représentation abstraite commune de tels objets permet d'en simplifier et d'en unifier la manipulation.
>
> — (Wikipédia, 2019)

L'objectif est de ne pas utiliser directement FFI mais plutôt de passer par une classe intermédiaire. Cette abstraction devrait dans sa version la plus simple être capable de me retourner un objet FFI contenant la bibliothèque que je compte utiliser. Je suis passé par différentes idée de conception, et j'ai fini par aboutir à celle-ci.

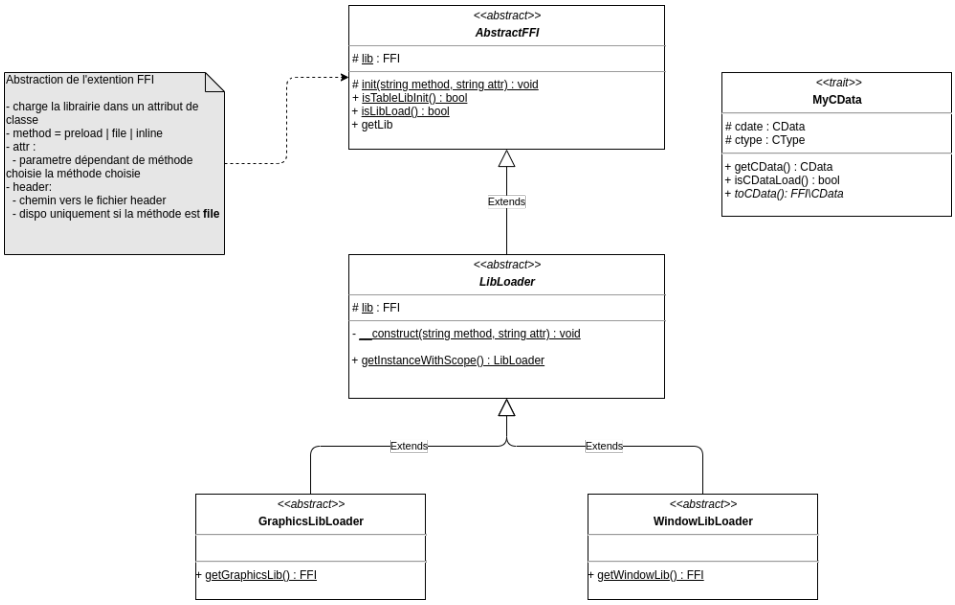

FIGURE 2.1: Diagramme de classe : AbstractFFI

On y voit plusieurs classes abstraites, la principale étant `AbstractFFI` dont hérite `LibLoader`, leurs rôles est dans leur nom :

- **AbstractFFI** : la principale classe qui se charge de s'abstraire du chargement de la bibliothèque et de faire les vérifications nécessaires. Elle a comme attribut un tableau d'objets FFI pour permettre l'utilisation de plusieurs bibliothèques tout au long du programme.
- **LibLoader** : héritant de AbstractFFI elle a les mêmes fonctionnalités, mais donne en plus la base pour mettre en place un *singleton* de génération de bibliothèque — une classe limité à une instance dont le seul objectif est de retourner une bibliothèque précise.

Pour ce qui est de **MyCData**, il s'agit d'un `trait`, particularité de PHP, qui est pour faire simple une classe abstraite qui s'utilise comme une interface. Son objectif est d'avoir un ensemble de fonctions et d'attributs prêts à être réutilisé pour définir une donnée C qui serait importée de la bibliothèque chargée avec FFI.

2.3.2 Architecture globale

Le reste de la bibliothèque est un ensemble de classes inspirées de SFML qui s'emboîtent autour de l'abstraction FFI. Effectivement, **SFML** et pas **C**SFML comme dit précédemment, car CSFML est juste un pont écrit en C vers SFML, or le C n'est pas un langage orienté objet. Par conséquent SFML qui est écrit en C++ qui lui est bien un langage orienté objet est idéal pour de servir de bases pour l'implémentation de mes classes. Toujours est-il que j'ai du simplifier un maximum l'architecture pour ne pas alourdir la bibliothèque en elle même et son utilisation finale.

2.3.3 Exemple de la classe Window

La classe Window est le deuxième pilier de l'architecture de la bibliothèque, comme la plupart des classes elle utilise MyCData pour bénéficier des méthodes et des attributs lier à l'échange de données entre PHP et la bibliothèque C via FFI. C'est le cas de la méthode `toCData()` qui convertit les attributs actuel de la classe en donnée C.

2.3.4 Diagramme de classes

Le résultat de la phase de conception est le diagramme de classe suivant qui a constamment évolué même lors de l'implémentation des fonctionnalités de la bibliothèque.

Diagramme de classes intéractif accessible à l'adresse `https://cutt.ly/phpml-class-diagram`

Le temps passé sur la réflexion de l'architecture de la bibliothèque n'a pas été en vain car il va nous permettre d'en gagner sur la partie principale qui est l'implémentation des fonctionnalités trouvées lors de la phase de conception.

2.4 Implémentation de la bibliothèque

L'implémentation des fonctionnalités se déroule lors de la phase de programmation, elle se déroule sans problèmes particuliers lorsque la phase de conception s'est bien déroulé car il n'y a pas besoin de réfléchir sur la structure de l'application ou de la bibliothèque car elle a déjà été définie. Néanmoins c'est au moment de l'implémentation qu'il faut penser à l'ordre de programmation des composants (bien que cela puisse être également fait lors de la conception) et que certaines faiblesses de conception apparaissent et je vais essayer d'en citer quelques unes.

2.4.1 L'ordre d'implémentation

Si la bibliothèque que l'on souhaite créer a des dépendances il est également probable que les éléments internes soient également interdépendants, cependant il est important de garder

le niveau de dépendance le plus bas possible. C'est là que se pose la question de l'ordre l'implémentation. En ce qui me concerne j'ai préféré commencer par la classe principale et redescendre l'arbre de dépendance, ce qui n'est pas forcément la meilleure des solutions mais elle a eu le mérite d'être efficace et de me révéler certaines erreurs de conceptions que j'avais commises.

2.4.2 Erreurs de conception

La plus grande erreur que j'ai pu faire a été au niveau de l'abstraction, car j'utilisais un `trait` qui n'était pas vraiment adapté à ce que je concevais à la base, erreur qui m'a été notifié par ma tutrice de projet après consultation. En corrigeant cela j'ai pu aboutir à la version que vous pouvez voir au 2.1.

En plus de diverses erreurs de conception corrigées sur le vif, j'ai été régulièrement confrontés à des erreurs de programmation plus techniques lié soit à FFI soit à CSFML. Celle qui m'a pris le plus de reflexions et de temps à corriger était une erreur de segmentation.

> Une erreur de segmentation (en anglais segmentation fault, en abrégé segfault) est un plantage d'une application qui a tenté d'accéder à un emplacement mémoire qui ne lui était pas alloué.
>
> — (Wikipédia, 2020b)

Il faut également noter que je n'ai jamais été confronté à cette erreur *en PHP* et il est très contraignant de retrouver la trace d'une erreur de segmentation quand on est développeur junior. Finalement il s'est avéré que retirer l'utilisation de la méthode `var_dump()` de mon script résolvait mon erreur.

Une fois l'implémentation finis il faut à présent s'attarder à la distribution de notre bibliothèque.

2.5 Distribution de la bibliothèque

Cette section est purement subjective car il s'agit de choisir comment partager la version finale de la bibliothèque. Evidemment chacun peut avoir son avis, je vais donc plus m'intéresser au *"comment"* plutôt qu'au *"pourquoi"* j'ai choisi cette méthode.

2.5.1 Un gestionnaire de paquets : composer

> Composer est un logiciel gestionnaire de dépendances libre écrit en PHP. Il permet à ses utilisateurs de déclarer et d'installer les bibliothèques dont le projet principal a besoin.
>
> — (Wikipédia, 2020a)

2.5.2 Installation de composer

Pour installer composer nous avons juste besoin d'avoir php installé sur sa machine, ensuite les étapes d'installation sont détaillées sur la page d'introduction à composer[4]. Mais pour résumer l'une des méthodes d'installation pour avoir composer globalement sur linux :

- Exécuter ces commandes pour télécharger la dernière archive de composer.

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === 'e0012edf3e80b6978849f5eff0d4b4e4c79
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

- Déplacer l'archive téléchargée vers son dossier bin :

```
mv composer.phar /usr/local/bin/composer
```

Avec composer installé nous pouvons créer intéractivement un fichier`composer.json` dans n'importe quel dossier ou projet avec la commande `composer init`.

2.5.3 Publication de bibliothèque avec composer et packagist

Avec composer il est facile de gérer les dépendances de son projet et de publier ses propres packets/bibliothèques. Les différentes étapes pour faire l'un ou l'autre sont brièvement décrites sur packagist.org[5]. Pour faire court il est nécessaire d'avoir :

- un compte un packagist.org[6]
- un dépôt GIT

[4] https://getcomposer.org/doc/00-intro.md#installation-linux-unix-macos
[5] https://packagist.org/
[6] https://packagist.org/

- un fichier `composer.json` minimal à la racine de son dépôt :

```
{
    "name": "vendor-name/package-name",
    "description": "A short description of what your package does",
    "require": {
        "php": "^7.2",
        "another-vendor/package": "1.*"
    }
}
```

2.5.4 Maintenabilité

Une fois son packet ou sa bibliothèque publié sur packagist il est désormais possible à n'importe qui de "demander" votre bibliothèque via composer avec la commande `composer require vendor-name/package-name` à condition d'avoir également un fichier `composer.json` à la racine de son projet.

Ce qu'il nous reste à faire est de maintenir la bibliothèques, c'est à dire :

- aux nouvelles pratiques de notre langage et mises à jour de nos dépendances
- ajouter de nouvelles fonctionnalités si on le souhaite.
- re-publier sur packagist quand cela est fait (ou activer la mise à jour automatique sur son compte).

Nous avons donc terminé le développement y compris la distribution de notre bibliothèque, nous pouvons à présent apprendre à l'utiliser, ce qui n'est pas aussi trivial que nous pourrions le croire.

2.6 Tableau récapitulatif

TABLEAU 2.2: Récapitulatif des dépendances

Dépendances	Commandes
PHP 7.4	`sudo apt install php7.4`
FFI	`sudo apt install libffi-dev`
SFML	`sudo apt install libsfml-dev`
CSFML	`sudo apt install libcsfml-dev`
Composer	voir le lien `https://getcomposer.org/download/`

3

Utilisation de la bibliothèque

3.1 Pré-requis à l'utilisation

Nous avons vu dans le 2.2 que la bibliothèque développée comportait certaines dépendances résumé dans le 2.2, PHP FFI et CSFML. Après les avoir installées il reste un minimum de configuration pour pouvoir parfaitement exécuter votre premier script.

3.1.1 Préchargement FFI et header CSFML

La bibliothèque utilise la fonctionnalité de préchargement de PHP pour précharger les header CSFML nécessaire pour charger et utiliser cette dernière en PHP. Il s'agit donc de modifier certaines variables de configuration de `php.ini` pour convenir à nos besoins. *Les instructions données seront pour linux, il est possible que la procédure change sur windows*

3.1.1.1 Activation de FFI :

Il faut s'assurer que FFI est en mode `preload` et est bien activé

1. dans le `php.ini` il faut que cette ligne soit présente `ffi.enable="preload"`
2. on peut vérifier la bonne activation de FFI avec la commande `php -m | grep FFI` qui doit afficher `FFI`
3. activer **opcache** en ligne de commande en modifiant la valeur du paramètre `opcache.enable_cli` à `true`.

3.1.1.2 Mettre en place le préchargement :

Les fichiers header (extension en `.h`) à précharger sont dans le dossier `preloading` du dépot de la bibliothèque. Les deux premières ligne de chaque fichiers définissent :

- l'espace de définition utilisé pour charger la bibliothèque C en PHP, **à ne pas modifier**
- le chemin *absolu* vers le fichier binaire de la bibliothèque à charger, CSFML pour notre cas.
  - Il faut bien s'assurer que ce chemin est le bon pour réussir le chargement de la bibliothèque avec FFI. On peut vérifier cela avec la commande `locate`. Par exemple pour le module Graphics de CSFML il faut exécuter la commande `locate libcsfml-graphics.so`.

  - **Attention !** si *SFML* n'est pas installé avec *CSFML* la bibliothèque ne pourra pas se charger.

Le chemin absolu vers le dossier contenant ces fichiers doit être renseigné dans la variable `ffi.preload` du fichier de configuration de PHP en **ligne de commande**, sur ma machine son chemins est `/etc/php/7.4/cli/php.ini`. Ce chemin doit ensuite être suffixé de `*.h` pour préciser que nous voulons tous les fichiers d'extension `.h` situé dans ce dossier.

- Par exemple si le chemin vers le dossier `preloading` du dépôt est `/home/user/phpml/preloading/` la nouvelle valeur de la variable sera : `ffi.preload="/home/user/phpml/preloading/*.h"`

Si tout s'est bien passé, avec cette nouvelle configuration on devrait être capable d'exécuter un script de test du dépôt sans erreurs dès qu'on l'aura installé dans un nouveau projet avec composer.

3.1.2 Installation dans un nouveau projet

Partant du contexte que vous commencez un nouveau projet appelé *nice-stuff* et que vous vouliez y intégrer cette bibliothèque, les étapes que vous aurez à réaliser doivent comprendre :

- la création du dossier `nice_stuff`
- l'initialisation de votre fichier `composer.json` avec `composer init`, après l'avoir préalablement installé 2.5.2
- l'ajout de la bibliothèque **PHPML** en tant que dépendance à votre projet : `composer require djuhnix/phpml`
- il est important de noter que vous aurez besoin de la fonction d'autochargement fournie par composer :
  - il vous suffit de mettre au début de votre script la ligne ajoutant le fichier `vendor/autoload.php`. Cette ligne pourrait ressembler à `require_once("vendor/autoload.php")` mais veillez à ce que le chemin passé à la fonction soit relatif à l'emplacement de votre script.

A présent vous êtes prêt à utiliser la bibliothèque PHPML dans votre projet, composer s'occupera de charger les différentes classes que vous appellerez.

3.2 Exemple d'utilisation et documentations

3.2.1 Utilisation

Vous pouvez trouver des exemples d'utilisation de la bibliothèque dans le dossier `tests/functionnal` du dépôt git. Mais voici comment réaliser certaines tâches avec PHPML.

3.2.1.1 Comment ouvrir une fenêtre

Il deux façons d'ouvrir une fenêtre, la première consiste à utiliser la classe de base `Window` et gérer par sois même les boucles qui permettent de garder la fenêtre ouverte et de gérer les événements, notamment la fermeture de la fenêtre pour éviter de démarrer une boucle infini.

```
$window = new Window(
    new VideoMode(400, 400),
    'Test Window Opening'
);
$event = new Event();

//Début de la boucle (tant que la fenêtre est ouverte)
while ($window->isOpen()) {

    // Gestion des événements
    while ($window->pollEvent($event->toCData())) {
        // Ferme la fenêtre si l'événement 'closed' est enregistré
        if ($event->getType()->getValue() == EventType::CLOSED) {
            $window->close();
        }
    }

    // DESSINER ICI


    // Nettoyage de l'écran de la fenêtre et affichage
    $window->clear($window->getBackgroundColor());

    // rafraîchissement et affichage des dessins s'il y en a
    $window->display();
}
```

La création d'une instance de la classe `Window` nécessite obligatoirement de définir un mode de rendu qui contient également la taille de la fenêtre. Pour simplifier, la création de la fenêtre

j'ai pris la décision de ne pas permettre la surcharge du mode de rendu vidéo et j'utilise celui de l'ordinateur par défaut, généralement de 32 bits par pixel.

Malgré cela, cette procédure restant fastidieuse à réaliser à chacun de ses scripts j'ai pensé à réaliser une autre classe `ExtendedWindow` héritant de la classe `Window` et qui étend ou plutôt facilite l'utilisation des fonctionnalités de cette dernière.

Désormais ouvrir une fenêtre revient à écrire

```
$window = new ExtendedWindow(
    new VideoMode(800, 600)
);

$window->run(new Event());
```

Comme le dit la documentation, la méthode `run()` de la classe `Window`

Lance la boucle principale de la fenêtre et l'ouvre dans le même temps.

En plus de cela elle gère également la boucle d'événement qui s'occupe de la fermeture de la fenêtre, s'occupe de dessiner à chaque rafraîchissement les objets qui lui sont attachés et possède bien plus d'atouts que nous détaillerons plus tard. Cette méthode facilite l'utilisation de bibliothèque à un point considérable car elle permet de s'abstraire des tâches répétitives lié à la création d'une fenêtre dans les bibliothèques graphiques classiques, c'est également la pierre angulaire de la librairie parce que c'est par elle que commence toutes activités que l'on voudrait réaliser avec la bibliothèque, comme dessiner des formes.

3.2.1.2 Comment dessiner

Pour dessiner sur une fenêtre il n'y a rien de plus simple que l'appelle à la méthode `draw()` de la classe `Window` qui prend en paramètre un objet "dessinable" préalablement initialisé (de préférence à l'extérieur de la boucle) et l'affiche sur la fenêtre actuelle.

Après avoir ouvert notre fenêtre, en supposant qu'on utilise la première méthode qu'offre notre bibliothèque, il faudrait écrire dans la partie réserver aux dessins (mise en évidence plus haut) :

```
$window->draw(
            new RectangleShape(
                [400, 200],
                [100, 200],
                $blue
```

```
            )
        );
```

Cependant faire cela est une mauvaise pratique et il serait préférable de créer ses formes avant le lancement de la boucle de la fenêtre, cela empêcherait l'inutile création de la même forme tout au long de la boucle qui maintient la fenêtre ouverte.

Cela reste valable même si l'on utilise la seconde possibilité d'ouverture de fenêtre qu'est la méthode `run()` de la `ExtendedWindow`. Pour revenir sur cette méthode, il faut savoir qu'elle prend en paramètre en plus de la variable d'événements, deux fonctions qui sont respectivement utilisé pour la gestion personnalisée d'événements par l'utilisateur et la réalisation de dessins sur la fenêtre, s'il n'y a que cette dernière option qui nous intéresse on peut toujours passé null à la fonction `run()` à la place de la fonction de gestion d'événements. Il est vrai que cette façon de faire nécessite de connaître le concept de fonction anonyme (closure) qui relève de l'algorithmie légèrement avancé pour un débutant mais dès qu'on sait comment l'utiliser le reste n'est que de simples détails théoriques.

```
$text = new Text("Hello");
$text->setFillColor($blue);
$text->setPosition([150, 50]);
$text->setCharacterSize(50);

$window->run(
    new Event(),
    null,
    function () use ($text, $window) {
        $window->draw($text);
    }
);
```

Le mot clé use est là pour dire que nous voulons utiliser ces variables pré-définie à l'interieur de notre fonction anonyme, sinon PHP ne sait pas que ces variables ont bien été déclaré et initialisé. Notre fonction anonyme peut ne pas être si anonyme qu'on peut le croire car on peut lui donner un nom, l'enregistrer dans une variable et passer cette dernière dans à la méthode `run()` plus tard dans notre script.

```
function dessine() {
...
}

$maFonction = 'dessine';
//ou
$maFonction = function () {
...
}
```

```
$window->run(new Event(), null, $maFonction);
```

Cependant il faut prendre en compte le fait que si l'on utiliser des variables mise à jour dynamiquement par la bibliothèque comme les événements cette méthode est à proscrire.

3.2.1.3 Comment garder les modifications appliquées à ses objets

Par défaut la bibliothèque n'enregistre pas toutes les modifications que vous appliquer à vos formes et autres objets dessinables parce qu'ils ne sont pas attachés à la fenêtre. Pour ce faire il faut utiliser la méthode `addToDrawingList()` qui comme son nom l'indique ajoute un objet identifié par une clé à la liste des éléments à dessiner qui seront enregistrés avec les données de la fenêtre et mis à jour à toutes futures modifications. L'objet enregistré peut toujours être retrouvé grâce à son identifiant et modifié si besoin.

```
$window->addToDrawingList(
    'rectangle', // la clé de l'objet
    new RectangleShape(
        [400, 200],
        [100, 200],
        new Color(Color::BLUE) // définition de la couleur de fond en bleue
    )
);

$window->run(
    new Event(),
    null,
    function () use ($red, $window) {

        $window
            ->getDrawingList()
            ->getObject('rectangle')
            ->setFillColor($red) // Le rectangle affiché sera rouge au lieu d'être bleue comme
    ;
    }
);
```

Tout cela dans le but de faciliter l'utilisation de la bibliothèque par des débutants et surtout flexibilité de celle-ci.

3.2.1.4 Extra

3.2.1.4.1 Créer une couleur

Des couleurs sont prédéfinie dans l'énumération prévu pour. Mais pour s'en servir il faut créer une instance de couleur avec l'une de ces valeurs.

```
$color = new Color(Color::RED) // équivaut à Color::DYNAMIC()
```

Il est évidemment possible de créer des couleurs dynamique avec des valeur de couleurs primaires.

```
$color = (new Color(Color::DYNAMIC)) // équivaut à Color::DYNAMIC()
    ->fromRGB(12, 12, 13);
```

3.2.1.4.2 *Gérer des événements*

On ne pourrait pas parler proprement de bibliothèque graphique sans gestion des événements déclencher sur la fenêtre, voilà pourquoi l'implémentation de la gestion des événements et des entrées (clavier et souris) faisaient parties de mes priorités.

On a précédemment vu sur les morceaux de codes qu'on pouvait créer une instance d'événement avec

```
new Event()
```

Le constructeur de cette classe ne prend rien en paramètre car tout est géré en interne pour l'utilisateur. Tout ce qu'il reste à faire pour l'utiliser c'est de vérifier qu'un certain type d'événement à bien été reçu et le traiter en tant que tel. Un exemple d'utilisation serait :

```
$event = new Event();
$window->run(
    $event,
    function () use ($event, $window) {
        if ($event->getType()->getValue() == EventType::MOUSE_BUTTON_PRESSED) {
            // réaliser une action si un bouton de la souris est cliqué
        }
    },
    function () use ($event, $window) {
       // faire ses dessins
);
```

3.2.2 Documentation

Actuellement la documentation de la bibliothèque n'est pas hébergé mais elle peut être lu directement sur les sources de la bibliothèque sur le dépôt github.

4

Résultat

4.1 Finalité de la bibliothèque

A la fin de mes quatre semaines de travail j'ai pu implémenté la majorité sinon toutes les fonctionnalités demandées par ma tutrice de projets pour cette mise en situation professionnelle. On peut alors être fier et dire que l'objectif est atteint, malgré tout la bibliothèque n'est pas complète et il reste des choses à approfondir.

Par exemple la possibilité de créer des formes personnalisé en redéfinissant ou en héritant de la classe `Shape` comme cela est expliqué dans un tutoriel de SFML sur la création de forme[1]. C'est l'une des choses que j'aurai aimé implémenter sur la fin de ma période de travail avec la création de formes convexes — la classe `ConvexShape`[2]. Ce qui ne pourra pas être fait par manque de temps mais surtout de capacités offertes par FFI.

Le principal étant que la bibliothèque soit utilisable pour les TPs de POO, l'implémentation de ce genre de fonctionnalités complexes passe au second plan. Même si cela n'est pas impossible, il est peut probable qu'un débutant puisse avoir l'initiative de se lancer dans ce genre de défi.

4.2 Bilan de la mise en situation professionnelle

La mise en situation professionnelle fut une épreuve très enrichissante sur tellement de points que je ne pourrai tous les aborder.

Elle m'a d'abord aidé à développer des compétences nouvelles tout comme à approfondir et étendre celles que je possédais déjà. Je pourrai citer la découverte et l'utilisation maximale des nouveautés qu'apporte le PHP 7.4 dans la création de PHPML mais aussi la connaissance du C — que je n'ai jamais pratiqué — et la bibliothèque SFML. Elle a également permis la culture de bonnes habitudes personnelle et professionnel grâce à la confrontation au télétravail, ce qui n'aurait pas forcément été possible lors d'un stage.

Il est certains que créer un package PHP n'est pas un projet aisé qui se fait dès la première idée. J'ai donc du établir un plan de développement et l'adapter à chaque évolution du projet que ce soit sur la conception ou l'implémentation, cela à évidemment contribuer à élargir la

[1] https://www.sfml-dev.org/tutorials/2.5/graphics-shape-fr.php#formes-personnaliscees
[2] https://www.sfml-dev.org/documentation/2.5.1-fr/classsf_1_1ConvexShape.php

vision que j'avais de la gestion de projet et dans le même temps améliorer mes compétences dans le domaine.

Pourtant il est difficile de croire que la mise ne situation professionnelle n'a que des qualités. En effet, malgré tout ce que j'ai gagné à réaliser ce projet il ne sera pas officiellement compté comme une expérience professionnelle; la où mon stage aurait permis d'en avoir un minimum et me démarquer parmi tous les autres débouchés la mise en situation professionnelle n'est qu'un projet de plus dans mon cursus d'étudiants.

4.3 Ressenti personnel

Globalement j'ai un ressenti positif envers l'expérience que j'ai vécu lors de ma mise en situation professionnelle. Il est plus pertinent de se faire un point de vue sur les bénéfices que sur les inconvénients.

Il était éprouvant psychologiquement de travailler au même endroit ou je me divertis, se retenir d'arrêter le travail pour jouer par exemple, alors que presque tout était réuni pour, fut l'un des efforts que j'ai du fournir pour arriver au terme du projet ou du moins à un minimum viable.

J'ai été confronté à divers problématiques que je n'ai pas tout le temps réussi à résoudre ou que j'ai failli abandonner, mais quoiqu'il en était contourner le problème pour y arriver ou demander conseil à ma tutrice de projet a toujours été le meilleur choix. Finalement j'en suis sorti avec plus d'acquis qu'avant et c'est je pense le plus important. ^

Conclusion

Pour décharger les tâches des professeurs du département informatique de l'IUT de Reims, liées à l'enseignement de la POO, il a été décidé de réaliser une bibliothèque graphique Java. Cependant elle a du être abandonné à cause du changement de langage dans le nouveau programme. Le projet de construire une nouvelle bibliothèque en PHP m'a donc été confié pour ma mise en situation professionnelle.

Le but était alors de trouver comment créer une bibliothèque orientée objets en PHP avec les mêmes possibilités que celle précédemment utilisé par le département, la principale étant l'ouverture d'une fenêtre système et le dessin de formes sur cette fenêtre à l'aide de manipulations et d'instanciations d'objets. Tout en considérant le fait que ce sujet n'avait pas encore été abordé en profondeur pour avoir une base de travail et que de plus je n'en avait aucune connaissance jusqu'à ce que je plonge dans la phase de recherches.

Avec les pointeurs que j'ai reçu en début de projet j'ai réussi grâce à FFI à implémenter une partie du module Graphics de CSFML en PHP et grâce à composer et packagist j'ai pu la distribuer pour une utilisation public. **PHPML** répond parfaitement aux réquisitions d'une bibliothèque PHP orientée objets et en ce qui concerne ses fonctionnalités elle comprend celles demandés en priorité par ma tutrice de projet. Bien que légère par rapport à toutes les fonctionnalités de l'ancienne bibliothèque et de CSFML elle pourra satisfaire les premiers besoins pour lesquels elle a été créer: alléger la création des travaux pratiques par les professeurs et leurs réalisations par les étudiants.

Ainsi j'ai pu apprendre à créer une bibliothèque en PHP et me servir de FFI, sans oublier la gestion de mon temps et de mon projets en télétravail. Néanmoins tout n'est pas accompli, le minimum pour la suite serait l'implémentation complète du module Graphics de CSFML pour mettre à jour cette librairie et l'apogée l'ajout de tous les modules composant CSFML pour que PHPML porte réellement son nom. Malgré la motivation éprouvé à l'égard de ce que serait cet exploit il ne sera pas aisé de réaliser notre volonté à ce jour étant donné les limitations actuelles de FFI et l'investissement de temps que cela représente. Mais on peut déjà espérer développer de petits projets avec ce qu'est PHPML aujourd'hui.

Liste des Abbreviations

POO Programmation Orientée Objets

> La programmation orientée objet (POO), ou programmation par objet, est un paradigme de programmation informatique. Il consiste en la définition et l'interaction de briques logicielles appelées objets ; un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre.

— (Wikipedia, 2020)

FFI Foreign Function Interface

C'est un mécanisme par lequel un programme écrit dans un langage de programmation peut appeler des routines ou utiliser des services écrits dans un autre.

SFML Simple and Fast Media Library

> SFML offre une interface simple vers les différents composants de votre PC, afin de faciliter le développement de jeux ou d'applications multimedia. Elle se compose de cinq modules : système, fenêtrage, graphisme, audio et réseau.

— (Laurent, 2020)

CSFML C Simple and Fast Media Library

> CSFML est le binding officiel de SFML pour le langage C. Son API est aussi proche que possible que l'API C++ (mais dans le style C, bien entendu), ce qui en fait un candidat parfait pour construire des bindings SFML pour d'autres langages qui ne supportent pas directement les bibliothèques C++.

— (Laurent, 2020)

URCA Université de Reims Champagne Ardenne

IUT-RCC Institut Universitaire de Technologie de Reims-Châlons-Charleville

Trello

.1 Tableau trello au début du projet

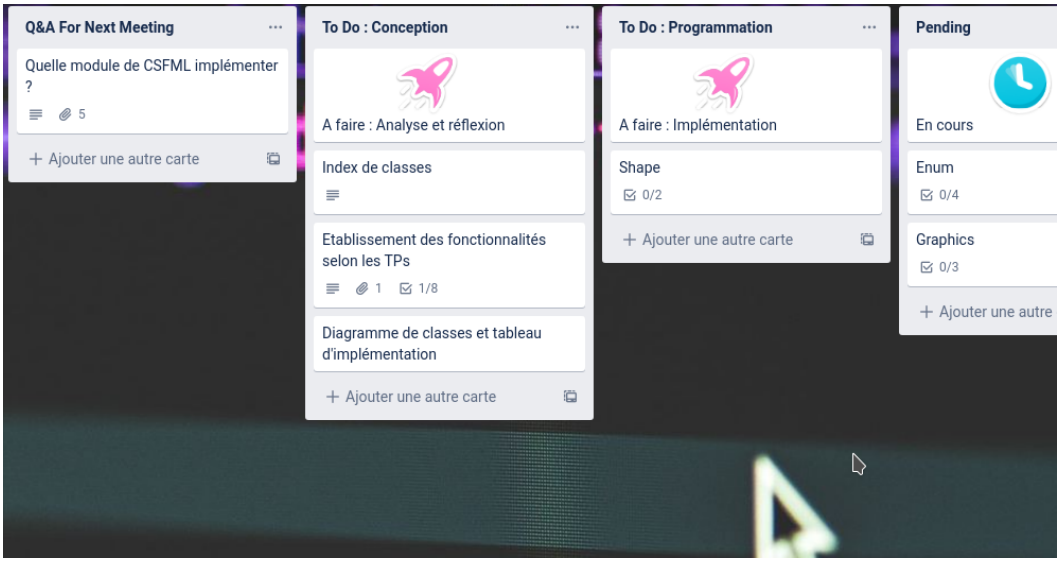

FIGURE 1: Tableau trello au début du projet

.2 Tableau trello du mémoire

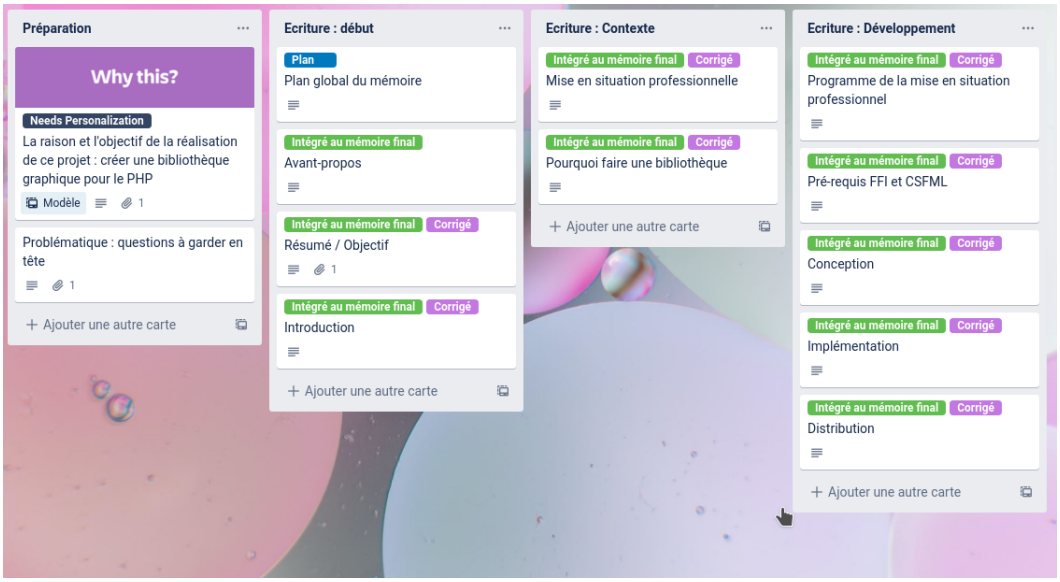

FIGURE 2: Tableau trello du mémoire

.3 Extrait de Documentation 1

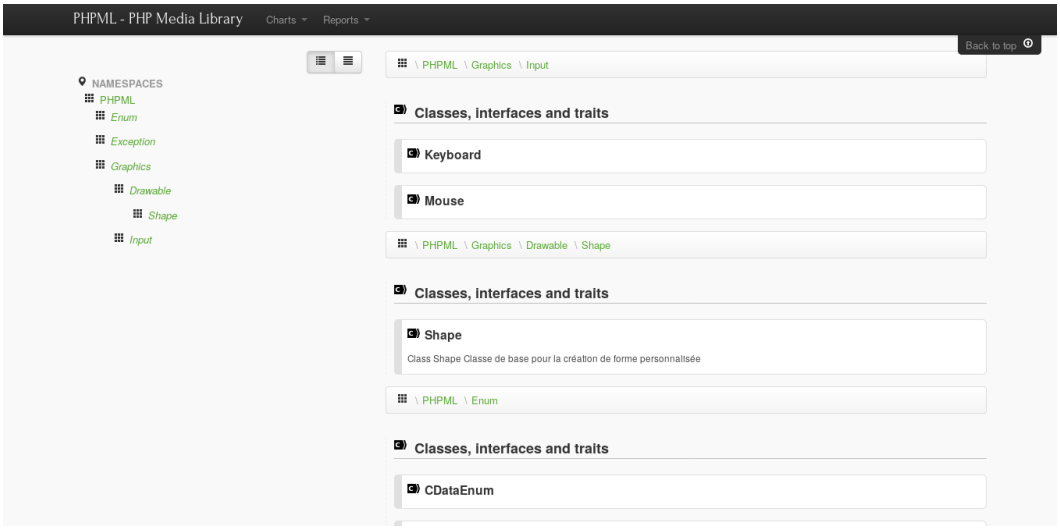

FIGURE 3: Extrait de Documentation 1

.4 Extrait de Documentation 2

.5 Extrait de documentation textuelles

protected function addLib(string $method, string $attr, string $key) : void Instancie un objet FFI contenant une bibliothèque C selon la méthode de chargement, et l'ajoute à la liste de bibliothèque. Si la bibliothèque est correctement chargé elle est accessible via l'attribut de classe $lib

- **param** `string $method` la méthode de chargement de la bibliothèque C. N'accepte que trois valeur :
  - *inline* : une séquence de déclaration en C doit être fourni au second paramètre.

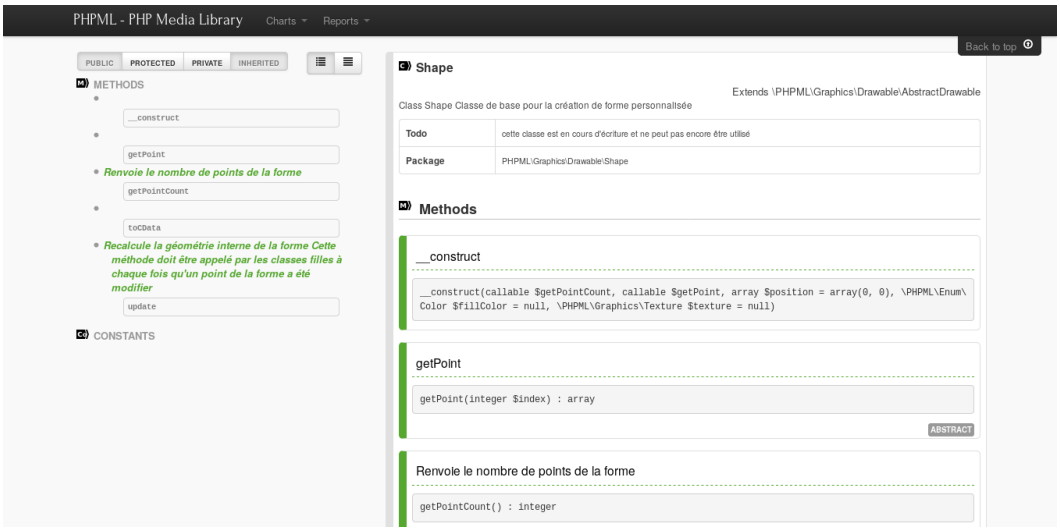

FIGURE 4: Extrait de Documentation 2

Un tableau peut aussi être passé au second paramètre. La première valeur étant la séquence de déclaration en C, et la deuxième le fichier de la bibliothèque à charger.
  - *preload* : charge dans l'objet FFI une bibliothèque déjà préchargé par PHP. Dans ce cas le nom de l'espace de définition 'scope' doit être passer au second paramètre.
  - *file* : instancie l'objet FFI selon les déclarations C contenu dans un fichier preloading (.h) Le chemin vers le fichier doit être passer au second paramètre.

- **param** `string $attr` dépend du premier paramètre. Elle peut recevoir une chaine de caractère représentant une séquence de déclaration en C ou un chemin vers le fichier h. Un tableau de taille 2 peut également être passé si la méthode est 'inline'. Si un type non attendu, un mauvais chemin ou une valeur contraire à la méthode définie est passé, un comportement non attendu pourrait survenir et la bibliothèque ne serait pas chargé.
- **param** `string $key` clé d'accès à la bibliothèque ajouté.

Bibliographie

Laurent, G. (2020). SFML.

Onisep (2019). Les DUT (diplômes universitaires de technologie).

The PHP Group (2019). PHP: Installation/configuration - manual.

Wikipedia (2020). Programmation orientée objet. Page Version ID: 171683789.

Wikipédia (2019). Abstraction (informatique). Page Version ID: 164904836.

Wikipédia (2020a). Composer (logiciel). Page Version ID: 169750090.

Wikipédia (2020b). Erreur de segmentation. Page Version ID: 171768684.