Germain Jr. OLEA-OYOUGOU

Memoire de mise en situation professionnel — Créer une bibliothèque PHP[1]

[1]IUT RCC
Barbara ROMANIUK

Note Personnelle

Une magnifique citation qui explique l'objectif et le sens profond de ce mémoire

Table des matières

Liste des Tableaux

Liste des figures

Avant-propos

Résumé

L'objectif de ce mémoire est de montrer comment on peut créer une bibliothèque qui pourra faciliter la découverte de la Programmation Orientée Objets (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 orienté objet par la création d'objet graphique.

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 Objets (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 "théorique" apprise 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 aussi mais est aussi pour certains un accomplissement de vie d'étudiants. 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 stage ont ainsi été annulé parce qu'il n'ont pas pu non plus se poursuivre 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 c'est *mes* méthodes de travail que je détaillerai dans la suite. Mais avant il est bon de se pencher sur le pourquoi et le comment j'ai eu comme sujet de mise en situation professionnelle 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 utilisais auparavant le Java comme langage principale 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 montrer 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 étant donné qu'il était impossible de créer des éléments graphique car inexistence de bibliothèque et de fonctionnalité. Il est certes possible de le faire sous certaines contraintes dans un environnement web, mais cela éloignerait l'étudiant de la console, étant 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 à changer la manière de réaliser les travaux pratiques à l'IUT de Reims. L'utilisation de scritps créer par l'étudiant pour créer des figures et en faire objets graphiques affichable sur une fenêtre système pour comprendre les notions de classes, d'objets et d'instances était très efficace. Malheureusement PHP est un langage orienté web qui ne permet pas la création de fenêtre graphique sur le système d'exploitation et par conséquent pas de manipulation d'objets graphique sur cette dernière.

L'arriver 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 bibliothèques en PHP.

1.2.3 L'intervention de Germain

Madame Romaniuk, maître de conférences en Informatique et ma tutrice pour cette mise en situation professionnel, 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 a le mérite d'être complète sur les fonctionnalités recherchées et n'a pas une grande courbe d'apprentissage pour être maîtrise quand on a un minimum de connaissance de C++. D'où le choix de madame Romaniuk pour l'implémentation 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 de POO ne serait pas fortement affecté si cette bibliothèque ne naissait pas mais il est certains que son existence allégera fortement la tâche pédagogique des professeurs.
- L'ouverture de possibilité pour les projets de fin d'année, car cette bibliothèque pourra être réutiliser et favoriser la créativité des étudiants.

Maintenant que nous savons pourquoi et quelle bibliothèque faut-il créer, il est temps d'entrer dans le vif du sujet et de parler du déroulement du développement de celle-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, se déroule dans un cadre du confinement donc il m'était dans l'obligation de faire du télétravail.

2.1.1 Organisation

Dès le début je me suis fait un programme pour répartir convenablement les 4 semaines qui m'étaient donné 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'as 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 à partir de la version 7.4 est obligatoirement pour développer et faire fonctionner la bibliothèque. Elle est intégré mais "inactive" par défaut (The PHP Group, 2019) sauf si l'on utilise PHP en ligne de commande ou la fonctionnalité de préchargement de PHP en environnement web. Pour activer FFI il faut s'assurer d'avoir la librairie FFI `libffi` installé 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 installé le nécessaire pour activer l'extension FFI de PHP.

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

- Sur Windows je n'ai pas explorer 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 bibliothèques 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 point clé de l'architecture de la bibliothèque. La première chose dont je me suis donnée l'objectif de réaliser lors de ma phase de conception était de trouver 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 cellui-ci.

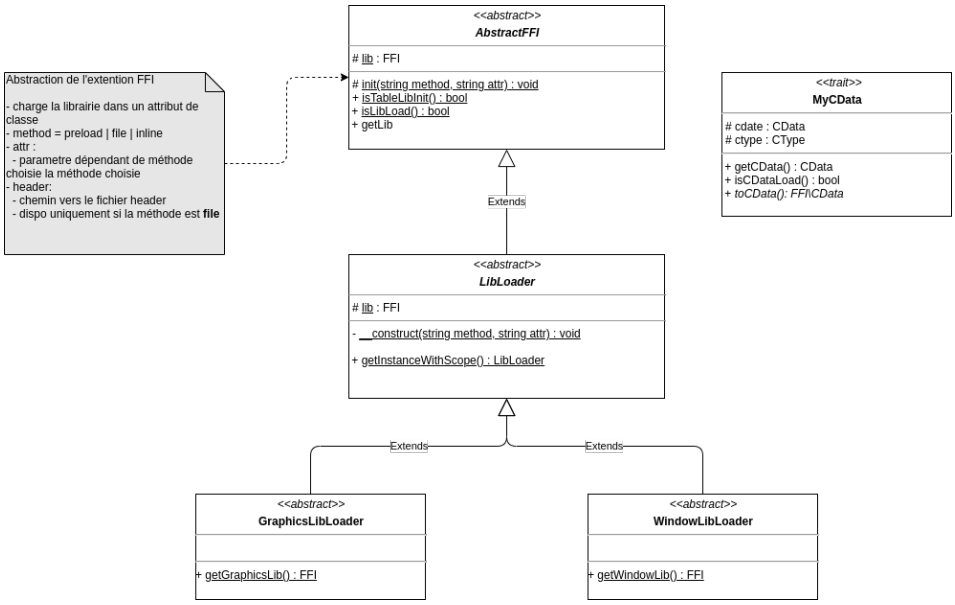

FIGURE 2.1: Diagramme de classe : AbstractFFI

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 bibliothèque tout au long du programme.
- **LibLoader** : héritant de AbstractFFI elle a les mêmes fonctionnalités, mais en plus elle donne 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, c'est simplement une classe abstraite qui s'utilise comme une interface pour faire simple. Son objectifs est d'avoir un ensemble de fonctions et d'attribut prêtes à être réutiliser pour définir une donnée C qui serait importer de la bibliothèque chargé avec FFI.

2.3.2 Architecture globale

Le reste de la bibliothèque est un ensemble de classe inspiré de SFML qui s'emboîtent autour de l'abstraction FFI. Effectivement, SFML et pas CSFML, car CSFML est écrit en C, or le C n'est pas un langage orienté objet et CSFML est juste un pont vers SFML qui est écrit en C++ qui lui est bien orienté objet. 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 classe

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 classe 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 grands accrocs 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. N'empêche que c'est au moment de l'implémentation qu'il faut penser à l'ordre de programmation des composants (bien que cela peut 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 ce 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 commencé 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 commise.

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 technique lié soit à FFI soit à CSFML. Celle qui m'a pris le plus de ressources cérébrales 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 noté 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 on a 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 résumé 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 télécharger vers son dossier bin :

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

Avec composer installé on peut 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 "demandé" 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, on peut à présent apprendre à l'utiliser, ce qui n'est pas aussi trivial qu'on pourrait le croire.

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, 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 donné 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 chargé la bibliothèque C en PHP, **à ne pas modifier**
- le chemin *absolu* vers le fichier binaire de la bibliothèque à chargé, 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 dans le dossier `tests/functionnal` du dépôt git. Voici par exemple les sources du test sur l'ouverture d'une fenêtre qui est le minimum à exécuter si vous voulez vérifier la bonne installation de la bibliothèque.

```
require_once __DIR__ . '/../../vendor/autoload.php';

use PHPML\Graphics\Event;
use PHPML\Graphics\VideoMode;
use PHPML\Graphics\Window;


$window = new Window(
    new VideoMode(800, 600)
);

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

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 bibliothèques sur le dépôt github.

4

Résultat

J'ai hâte moi aussi

4.1 Exemple 1

4.2 Exemple 2

Conclusion

J'aurai enfin fini…

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.

URCA Université de Reims Champagne Ardenne

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

Bibliographie

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.