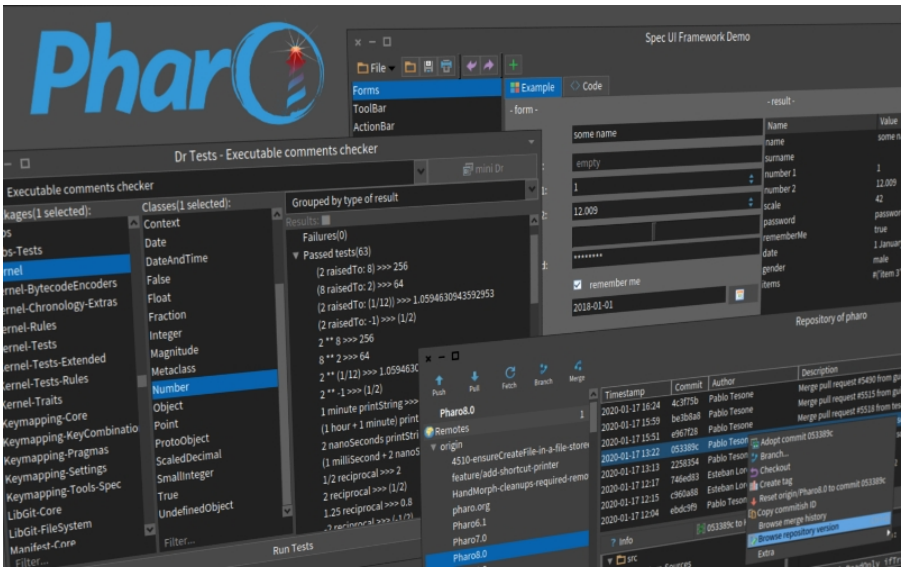

IDE Multi-fenêtre en Pharo

Par Dufloer Rémi

Tuteur Entreprise : Guillermo Polito

Tuteur Universitaire : Sylvain Salvati

Entreprise : Inria, Park Plaza, Parc scientifique de la Haute-Borne, 40 Av. Halley Bât A, 59650 Villeneuve-d'Ascq

Université de Lille : Cité Scientifique, 59650 Villeneuve-d'Ascq

1 Remerciement

Je tiens à remercier l'intégralité de l'équipe Rmod qui ma permis de me sentir intégré dans l'équipe durant ce stage.

Je tiens à remercier particulièrement Guillermo Polito pour le temps qu'il à passé à m'apprendre de nouvelles chose, ce fût un véritable plaisir de l'avoir comme tuteur.

Je tiens à remercier mon tuteur universitaire Sylvain Salvati pour ces conseils.

Je voudrais remercier particulièrement Monsieur Routier, c'est grâce à lui que j'ai eu cette passion pour l'informatique.

Je remercie finalement, l'ensemble des enseignants de Licence Informatique, pour m'avoir apporté les compétences nécessaires pour accomplir ce stage.

2 Résumé

Ce document rapporte les résultats de mon stage de Licence dans l'équipe de recherche Rmod, Inria. L'équipe Rmod est à la base du développement du langage de programmation Pharo et de son environnement de développement (IDE). Pharo est un langage orienté objet de propos général.

Au début de mon stage, l'interface graphique de l'IDE de Pharo s'exécutait dans une seule et unique fenêtre du système d'exploitation. Ces fenêtres sont dites "natives". On retrouve dans cette fenêtre des fenêtres "simulées" qui correspondent aux outils de Pharo. L'IDE ne supportait pas l'utilisation de plusieurs fenêtres natives.

Des entreprises utilisant Pharo veulent la possibilité d'utiliser un IDE avec du multi-fenêtrage. Dans ce contexte, Esteban Lorenzano, ingénieur dans l'équipe Rmod, travaille aujourd'hui sur le développement d'un nouvel IDE sur le moteur graphique GTK.

Mon objectif est donc d'étendre l'IDE avec plusieurs fenêtres natives qui nous permettent de pouvoir utiliser plusieurs écrans sans changer la métaphore de l'IDE, le tout permettant une rétrocompatibilité entre l'IDE actuelle et le prochain.

J'ai mis en place la possibilité de sortir des fenêtres simulées afin d'obtenir plusieurs fenêtres natives. En ce qui concerne les fenêtres natives, j'ai mis en place le moyen de les ouvrir ainsi que de les fermer. J'ai mis en place la possibilité d'ouvrir les fenêtres naturellement dans le même état avant la fermeture de l'IDE. J'ai créé un gestionnaire de fenêtre afin de créer des bureaux comme sur Mac et de gérer ces fenêtres de manières programmatique.

J'ai appris énormément de choses sur tout ce qui touche le génie logiciel, c'est à dire réaliser un projet maintenable, avec de bonnes conventions ainsi que la pratique du TDD, qui permet de développer à partir de tests, ce qui permet d'être sûr que ce que l'on développe fait exactement ce que l'on veut. J'ai aussi appris différentes choses en dehors de l'informatique. Par exemple, j'ai amélioré grandement mon anglais et j'ai appris de nombreuses choses sur différentes cultures. L'équipe Rmod est une équipe de recherche donc cela m'a permis de découvrir ce monde passionnant où nous avons l'impression que ce dernier est très fermé et peu accessible.

3 Abstract

This document summarizes my License internship in the research team Rmod, Inria.
The team Rmod forms the basis of the programming language development in Pharo and its development environment (IDE). Pharo is a general-purpose object-oriented language.

At the beginning of my internship, the graphical interface of the IDE of Pharo was executed in a single operating window. These windows are called "native".
We can find in those native windows some "simulated" windows which correspond to Pharo's tools. The IDE did not support the use of several native windows.
Some companies want the possibility to use an IDE with multi-windowing.

In this context, Esteban Lorenzano, engineer in the Rmod team, works today on the development of a new IDE on the GTK graphic engine.

My goal is therefore to extend the IDE with several native windows which allow us to use several screens without changing the IDE metaphor. All of this allows backward compatibility between the current IDE and the next one.

I have implemented the possibility to exit simulated windows in order to obtain several native windows. Concerning native windows, I have set up a way to open and close them.
I have implemented the ability to open windows naturally in the same state as before the IDE had been closed.
I have made a window manager to create desktops like on Mac to manage these windows programmatically.

I have learnt a lot about everything related to software engineering like what making a maintenable project with good conventions means as well as the practice of TDD, which allows to develop starting from tests, which allows us to be sure that what you develop does exactly what you want.
I have also learnt a lot of different things besides computing. For example, I greatly improved my English and I have learnt many things about different cultures.
The Rmod team is a research team, which allowed me to discover this exciting field where we have the impression that it's very closed and not very accessible.

4 DPP, Enjeux environnementaux du numérique

Lors du semestre 6, j'ai suivi l'option "Enjeux environnementaux du numérique". Lors de cette option j'ai eu la chance de participer a la fresque du numérique. Ce qui ma permis de faire évoluer ma vision de l'impact du numérique sur l'environnement.

En effet je connaissais l'impact désastreux des data centers et de l'infrastructure sur l 'environnement, via leur consommation et leur chaleur que cela dégage. Mais , je ne savais pas que pour la fabrication d'un ordinateur de 2kg nous nécessitons 200 kg d'énergies fossiles ainsi que 600 kg de minéraux, qui servent pour l'extraction et le raffinage des métaux. Il faut donc 800 kg de ressources auquel il faut ajouter plusieurs milliers de litres d'eau douce utilisées pour la fabrication ainsi que l'extraction. Savoir que l'on utilise des milliers de litre d'eau dans des pays où les habitants n'en n'ont pas pour vivre et vraiment choquant.

Le numérique à donc un impact important à la création de matériels cependant cette impact, impact la stabilité géopolitique entre différent pays et même entre dirigeant et peuple. Cela ma permis de remettre en cause mon utilisation du numérique.

Avant cette fresque j'éteignais certes mes équipement numérique et j'en prenais déjà soins. Mais maintenant si j'ai besoin d'acheter un outils numérique je vais d'abord regarder au niveaux occasion et circuit court au lieu d'acheter un outils neuf.

Lors de mon stage j'ai discuté avec les chercheurs de l'impact du numérique de l'INRIA. J'ai ainsi appris qu'ils ont ouvert un MOOC ouvert à tous afin de sensibiliser et d'apprendre les bons geste vis à vis de l'impact du numérique. L'équipe de recherche Spirals développe un logiciel, *PowerApiI ,* qui permet de mesurer la consommation énergétique d'autres logiciels. Cela permet à l'utilisateur de ce rendre compte de son impact

Table des matières

5 Introduction

Ce document rapporte les résultats de mon stage de Licence dans l'équipe de recherche Rmod, Inria. L'équipe Rmod est à la base du développement du langage de programmation Pharo et de son environnement de développement (IDE). Pharo est un dérivé de SmallTalk. Pharo est un langage orienté objet de propos général.

Au début de mon stage, l'interface graphique de L'IDE de Pharo s'exécutait dans une seule et unique fenêtre du système d'exploitation, ces fenêtres sont dîtes natives. On retrouve dans cette fenêtre des fenêtres simulées qui corresponde au outils de Pharo. L'IDE ne supportait pas l'utilisation de plusieurs fenêtres natives. Nous n'avions pas la possibilités de docker les fenêtres non plus.

Des entreprises utilisant Pharo ne veulent plus de cela, ils veulent la possibilité d'utiliser un IDE avec du multi-fenêtrage. Dans ce contexte ,Esteban Lorenzano, ingénieur chez l'équipe Rmod, travaille aujourd'hui sur le développement d'un nouvel IDE sur le moteur graphique GTK. Ce projet à long terme a comme objectif de changer complètement la métaphore de l'IDE et de le moderniser avec un ensemble de composants graphiques standard et bien maintenus, ce qui permettra de mieux répondre aux besoins des entreprises.

Mon objectif est donc d'étendre L'IDE avec plusieurs fenêtres natives qui nous permettent de pouvoir utilisez plusieurs écran sans changer la métaphore de l'IDE. Le tout permettant une rétrocompatibilité entre l'IDE actuelle et le prochain, c'est à dire être la passerelle entre l'ancien et le nouveaux IDE. Cette rétrocompatibilité permet aussi de ne pas devoir réimplémenter GTK par exemple.

J'ai mis en place la possibilités de sortir des fenêtres simulées afin d'obtenir plusieurs fenêtres natives. Pour cela j'ai du passez d'une fenêtre native qui était à un singleton à plusieurs fenêtres natives. Tout en gérant le rendu graphique et les événements de chacune de ces fenêtres natives.
En ce qui concerne les fenêtres j'ai mis en place le moyen de les ouvrir ainsi que les fermer. J'ai mis en place la possibilité d'ouvrir les fenêtres naturellement dans le même états avant la fermeture de l'IDE. Cela reprend une partie importante de la métaphore de L'IDE actuelle de Pharo.
J'ai créer un gestionnaire de fenêtre afin de créer des bureaux un peu comme sur Mac, et de géré ces fenêtres de manières programmatique. Via cette implémentation j'ai implémenté une TaskBar pour les fenêtres natives.

J'ai appris énormément de chose sur tout ce qui touche le Génie logiciel, c'est à dire faire un projet maintenable, avec de bonne convention ainsi que la pratique du TDD, qui permet de développer en fonction du test, ça permet d'être sûr que ce qu'on développe fait exactement ce que l'on veut. J'ai aussi appris différentes choses en dehors de l'informatique, par exemple, j'ai amélioré grandement mon anglais, j'ai appris de nombreuses choses sur différentes

cultures. L'équipe Rmod est une équipe de recherche ainsi cela ma permit de découvrir ce monde passionnant où nous avons l'impression que ce dernier est très fermés et peu accessible.

6 Contexte

6.1 L'Inria

J'ai eu la chance de faire mon stage de licence au cœur de l'équipe Rmod de L'Inria. L'Inria est l'institut national de recherche en sciences et technologies du numérique. La recherche de rang mondial, l'innovation technologique et le risque entrepreneurial constituent son ADN. Au sein de 200 équipes-projets, pour la plupart communes avec les grandes universités de recherche, plus de 3 900 chercheurs et ingénieurs y explorent des voies nouvelles, souvent dans l'interdisciplinarité et en collaboration avec des partenaires industriels pour répondre à des défis ambitieux.

Créé pour être au carrefour entre le monde académique et l'industrie, Inria est positionné sur les nouvelles frontières de la recherche numérique. Pionnier pour faire émerger les nouvelles disciplines à partir des mathématiques appliquées et de l'informatique, pionnier pour accompagner la dynamique des startups technologiques en France et en Europe, l'institut a été visionnaire et précurseur dans de nombreux domaines (calcul scientifique, Internet, Web).

Au sein de l'INRIA je me trouve en stage dans l'équipe RMOD, une équipe de passionné. L'objectif de RMoD est d'aider à la re-modularisation des applications orientées à objets. Cet objectif est attaqué suivant deux axes complémentaires : la réingénierie et la définition de nouveaux constructions dans les langages de programmation. Dans le cadre de la réingénierie nous allons proposer de nouvelles analyses pour comprendre et restructurer de grandes applications (métriques spécialisées, visualisations adaptées). Rmod travaille sur le développement et la programmation du langage Pharo.

Le but du logiciel Pharo est de fournir un environnement de développement clair, innovant, libre et open source à travers d'un cœur léger et stable. Ainsi que des outils de développements intégré de grande qualité permettant le déploiement d'applications critiques.

De plus Pharo est un langage objet minimal, élégant, pur et réfléchissant, tout ce qui s'y trouve n'est que objet, la syntaxe complète de Pharo tient sur une carte postale et le développement ce fait dans le débogueur.

6.2 L'IDE de Pharo

L'environnement de développement (IDE) du langage Pharo est organisé dans une unique fenêtre native. Une fenêtre native est une fenêtre géré par le système d'exploitation. Les outils de l'IDE tels que, le gestionnaire de versions, le navigateur de code et bien d'autres, se trouvent dans des fenêtres simulés, c'est à dire bloqués dans la fenêtre native. Voici un exemple qui illustre ça . Dans la figure 1 les deux fenêtres rouges sont les fenêtres simulés, la fenêtre bleu est dite native.

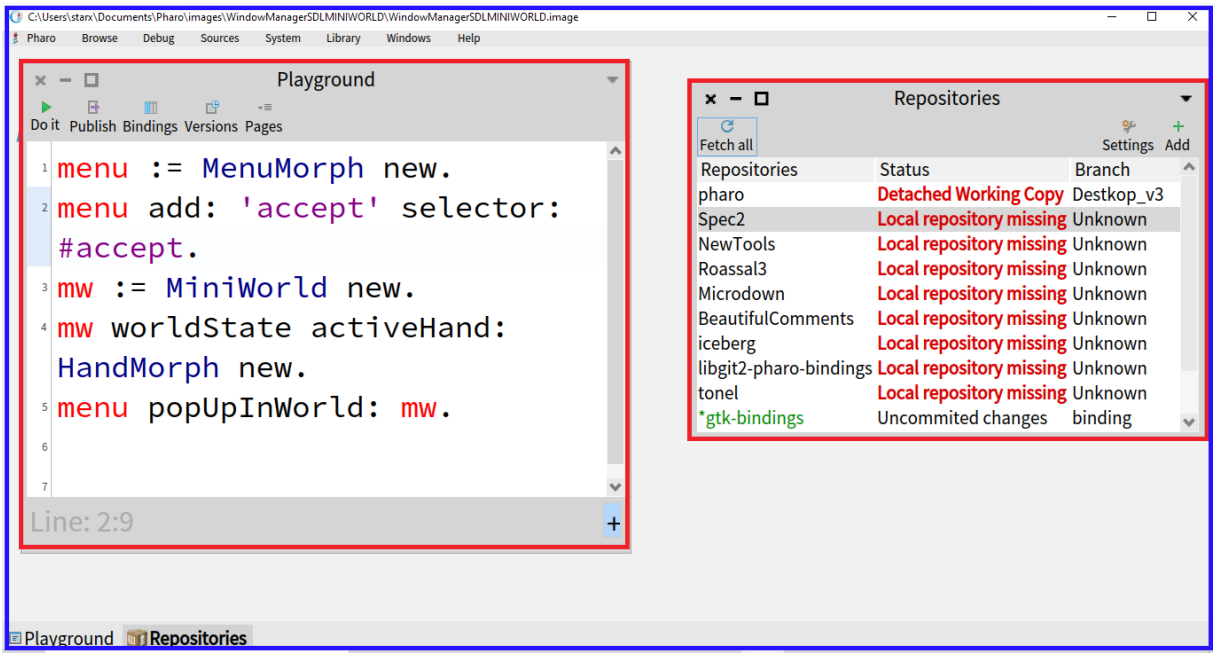

Fig 1 . fenêtre native et fenêtre simulé en Pharo

Avec cette IDE nous n'avons pas la possibilité d'avoir plusieurs fenêtres natives ce qui empêche le travaille avec plusieurs écrans. Nous n'avons pas la possibilité de docker les fenêtre simulés dans la fenêtres natives non plus.

Dans ce contexte,Esteban Lorenzano, ingénieur chez l'équipe Rmod, travaille aujourd'hui sur le développement d'un nouvel IDE sur le moteur graphique GTK. Ce projet à long terme a comme objectif de changer complètement la métaphore de l'IDE et de le moderniser avec un ensemble de composants graphiques standard et bien maintenus, ce qui permettra de mieux répondre aux besoins des entreprises.

Cependant, il faut améliorer l'IDE actuel avant de passer au prochain. Ce qui permettra d'avoir un point de rétrocompatibilité, c'est à dire avoir une compatibilité entre l'ancienne IDE et celui en développement, afin de ne pas devoir tout réécrire ce que l'on à en GTK.

6.3 Architecture existante : Morphic

L'IDE de Pharo est lui-même écrit en Pharo. Il utilise le framework graphique « Morphic ». Ce framework permet de créer des éléments graphiques, ces éléments sont rattachés à un parent, appelé aussi owner, et ont la possibilité d'avoir des enfants, les enfant sont appelez des submorphs.

Cela forme un arbre à l'éxecution. La racine des éléments graphiques d'une fenêtre natives n'a pas de parent. On retrouve à cette racine une instance de WorldMorph. Les fenêtres simulés sont représenté par les Systems Window qui sont des enfants directes de la racine. Les fenêtres simulés pourront correspondre à différent outils de L'IDE Pharo. Les outils de Pharo sont par exemple, le navigateur de code, le gestionnaire de versions et bien d'autres.

Voici un schéma qui représente le framework Morphic.

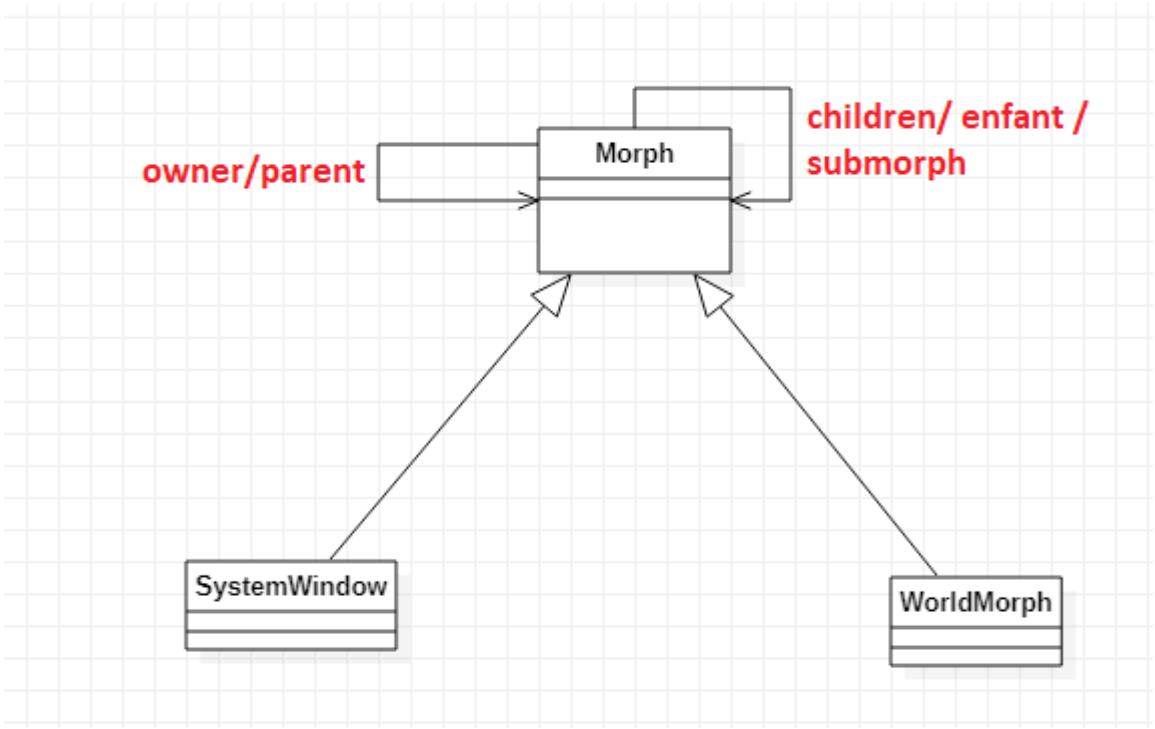

Fig 2 Le framework Morphic

Voici un schéma de comment cela fonctionne :

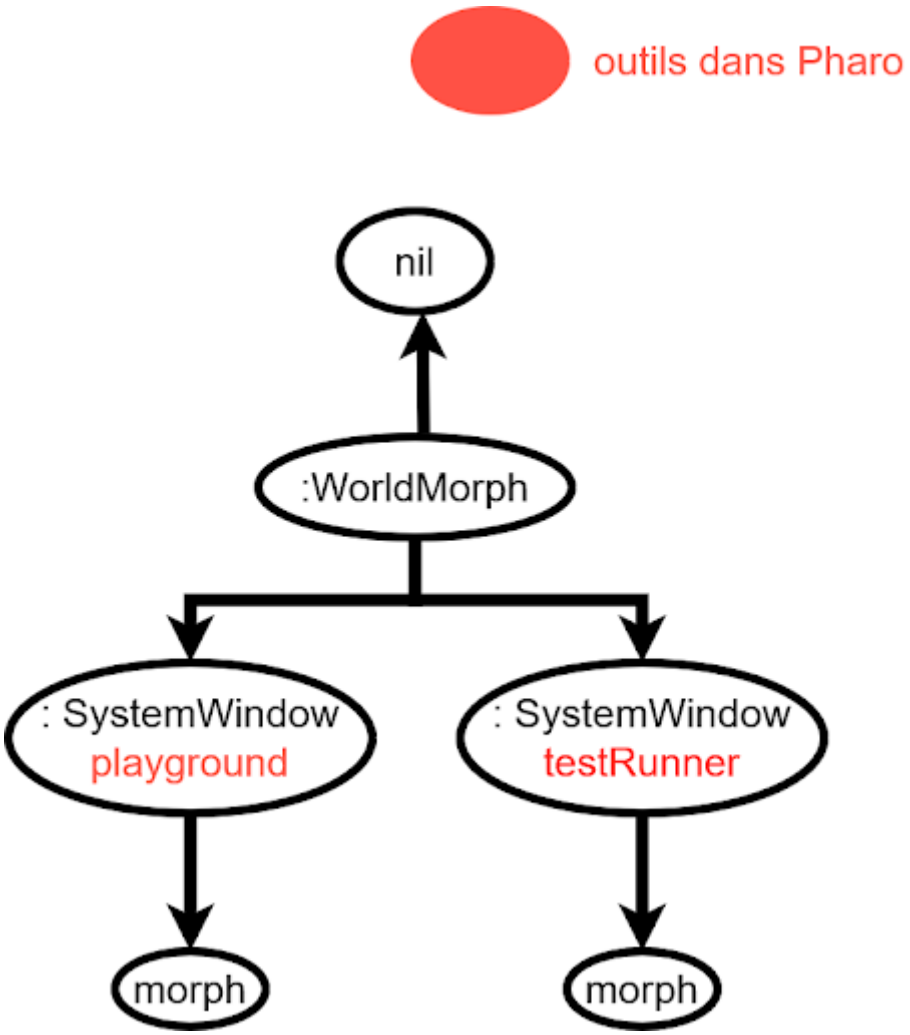

Fig 3. Schéma représentant la disposition des morphs

WorldMorph est la racine des éléments graphique des fenêtres native. Ce dernier connaît ainsi tout ce qui s'y trouve. Un WorldMorph à un WorldState qui permet de définir l'état dans lequel il se trouve. Cela permet de savoir si ce dernier à besoin d'être redessiné ou si un de ces enfants à évolué. Le WorldState à un WorldRenderer qui se charge du rendu graphique. WorldState et WorldRenderer permet de faire un rendu différent en fonction de l'état où l'on se trouve.

Voici un schéma résumant le fonctionnement de WorldMorph :

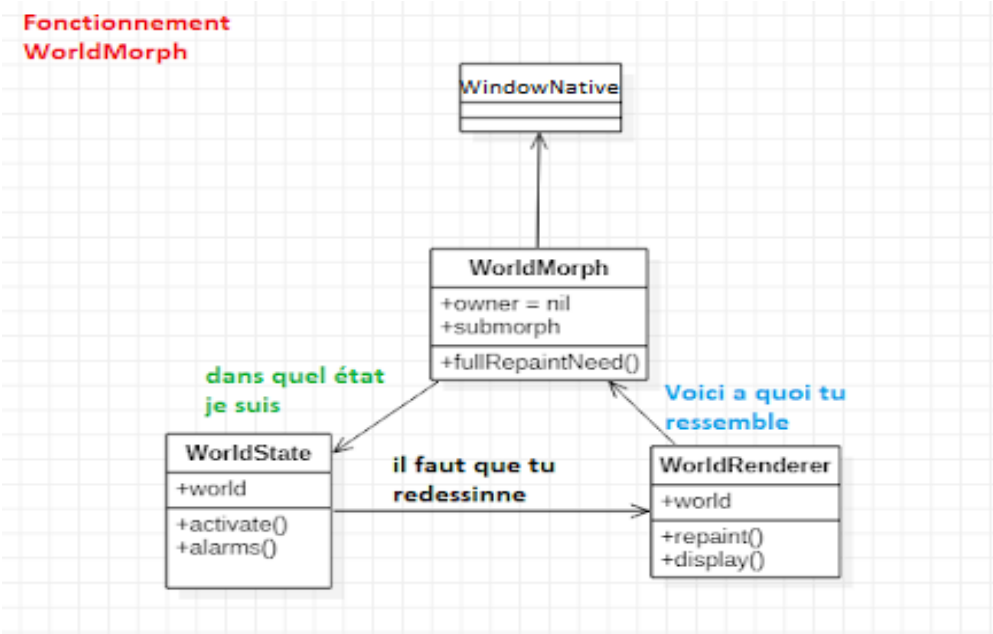

Fig 4, Fonctionnement de WorldMorph

Le backend de la fenêtres natives utilise le moteur graphic «SDL» .Le but est de passez à un backend GTK. On se retrouve avec 2 backends.

Mais on se retrouve avec un problème, WorldMorph est un singleton, il ne possède qu'une unique instance, le World.World est donc un singleton qui ne marche que pour un type de fenêtre native. Une solution rapide est de dupliqué le World en modifiant seulement les endroit ou l'on parle des backends. Mais cela est une mauvaise pratique car nous obtenons beaucoup de répétition de code .

6.4 Objectifs

Mon objectif est donc d'étendre L'IDE avec plusieurs fenêtres natives qui nous permettent de pouvoir utilisez plusieurs écran sans changer la métaphore de l'IDE. Le tout permettant une rétrocompatibilité entre l'IDE actuelle et le prochain, c'est à dire être la passerelle entre l'ancien et le nouveaux IDE. Cette rétrocompatibilité permet aussi de ne pas devoir réimplémenter GTK par exemple.

Voici un schéma qui résume ce que l'on veut :

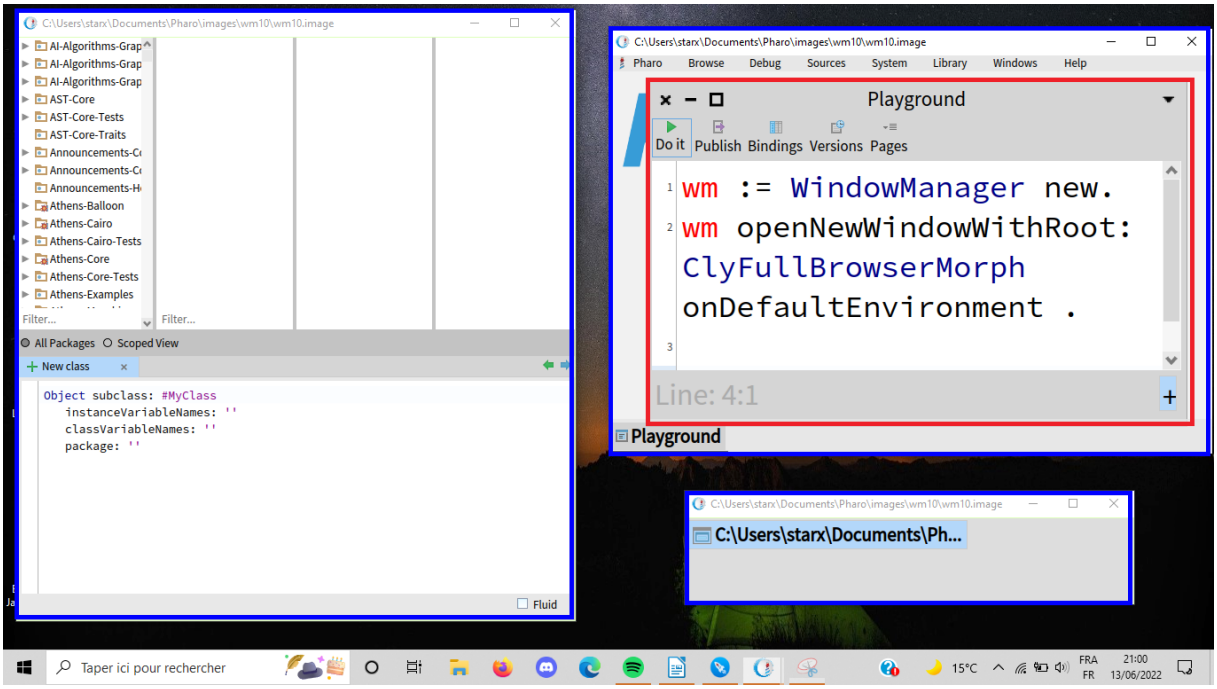

Fig 5. ce que l'on veux / a changé

Pour répondre à mon objectifs j'ai établie un cahier des charges :

- avoir plusieurs fenêtres native
- trouver le design pour ouvrir ces nouvelles fenêtres natives
- pouvoir rouvrir les fenêtres natives dans le même état qu'on les à sauvegarder avant de fermer l'IDE.
- pouvoir gérer le tout de manières programmatique
- que le tout soit testé

7 Contributions

7.1 Overview

La solution que je propose est de créer des fenêtres natives qui auront chacun un World minimale, et ces fenêtres seront gérées par un gestionnaire de fenêtre native.

Voici un diagramme Uml qui reprend cette solution :

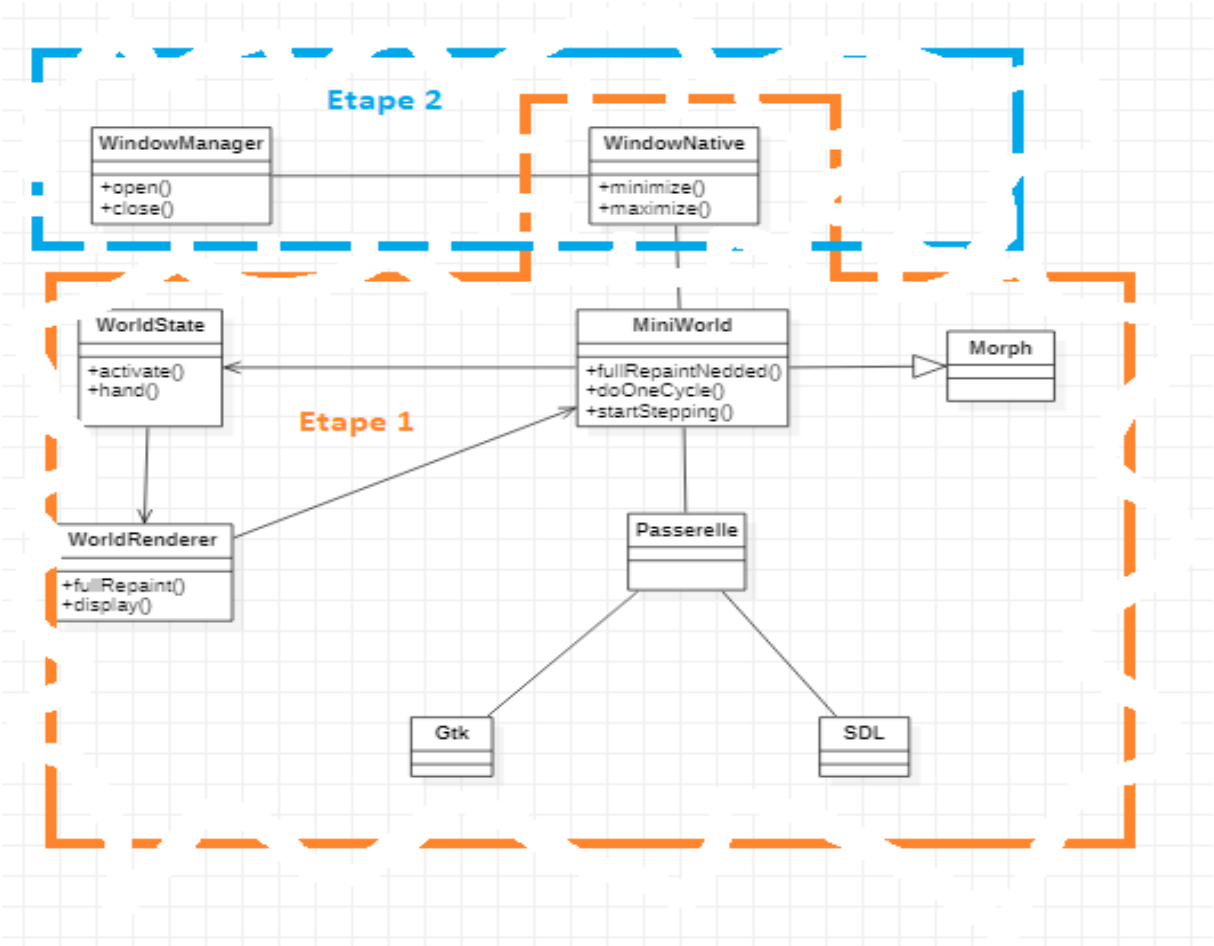

Fig 6 La solution créé afin de répondre au objectifs

Cette solution est découpée en 2 étapes. La première : créer un *MiniWorld* qui sera une instance de Morph. Et la seconde : créé un WindowManager pour gérer les fenêtres natives avec un objet window qui les représentera.

Créer le *MiniWorld* permettra de réglé, le problème de duplication, cela permettra de faire la transition vers le nouvel IDE de manière propre.Afin de géré ces nouvelles fenêtres natives ma solution est de créé un gestionnaire de fenêtre.

Créé un gestionnaire de fenêtre à pour but de créer des bureaux virtuelle qui permettront de géré certaines fenêtres ensemble. Ce gestionnaire doit contenir une barre de tâches comme la fenêtre native par défaut.

Entre le *MiniWorld* et le WindowManager on retrouve l'élément centrale qu'est la Window. La Window représente la fenêtre native.

La Window possède une instance *MiniWorld*, elle connaît qu'elle est l'outil qu'elle contient sone backend qui est utilisé et ces attributs, on entend par attributs taille position icon titre ect...

Le design actuelle nous pose des défis technique, comme la spécification du rendu et des événements de chaque backend avec comme World un singleton.

7.2 Le MiniWorld

La première étape est donc de créé un « *MiniWorld* » qui à pour but d'être générique, et minimale ainsi que de respecter le cahier des charges suivant :

- gère le rendu graphique

- gère les événements, et le dessin de ces derniers

- avoir de bonne performances

7.2.1 Gestion Graphique de L'IDE

7.2.1.1 Gestion de plusieurs backends

Ma solution en ce qui concerne le rendu graphique est d'avoir une passerelle entre le *MiniWorld* est les backends. Cette passerelle et représenté par Root

Voici Donc le diagramme UML :

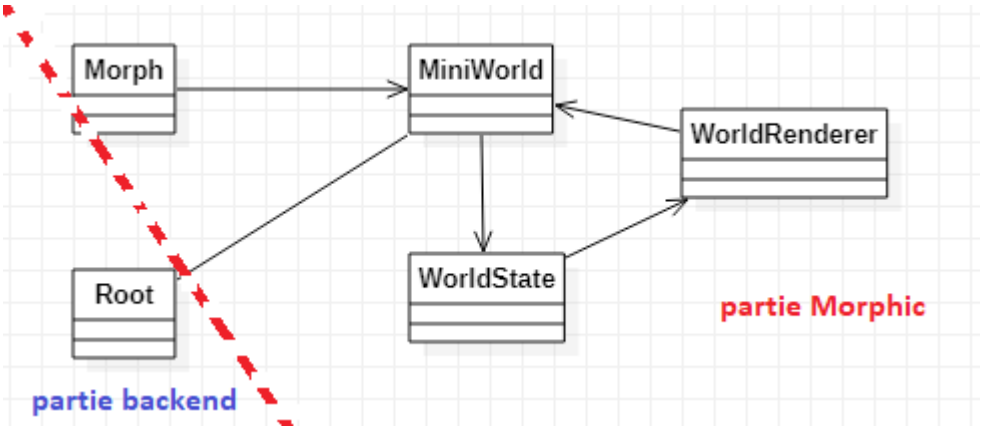

Fig 7 diagramme UML gestion graphique

Tout ce qui est rendu est donc géré par le WorldRenderer en fonction de la demande du Root .

J'ai développé un système avec des backends qui exploite le polymorphisme, cela permet d'avoir une « interface » unique pour des backends différents. Mais cela permet aussi d'ajouter par la suite de nouveaux backends sans devoir modifier l'implémentation actuelle.

Passer par une « interface » permet d'avoir un IDE plus flexible qui accepte plus facilement le changement.

7.2.1.2 Un Rendu Graphique Performant

Une première implémentation du dessin est de dessiner la fenêtre natives en entier tout le temps.Ma solution en ce qui concerne de la performance et de suivre le même modèle que la fenêtre native de base en Pharo c'est à dire, ne dessiner que ce qui subit un changement.

Voici un schéma expliquant cette solution :

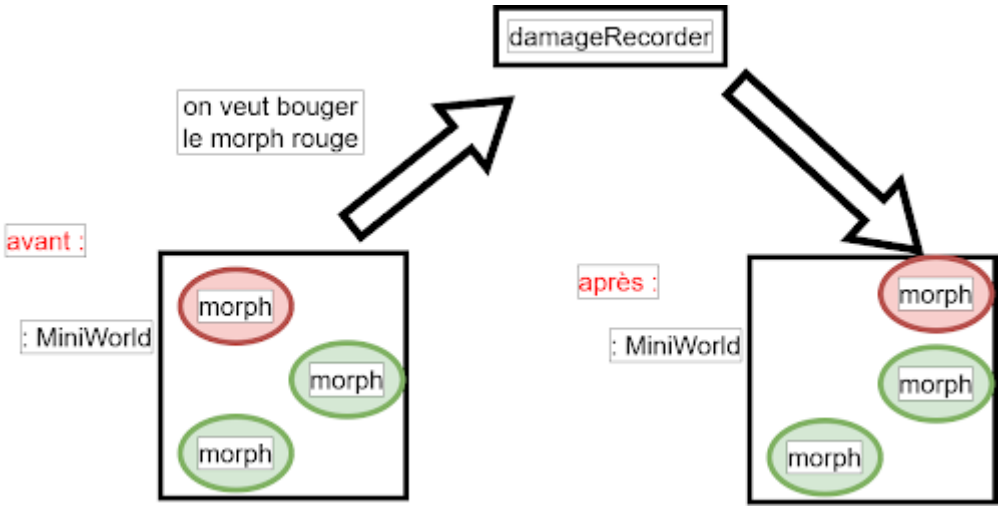

Fig 8 Comment fonctionne le *damageRecorder*

Si un morph de la fenêtre natives nécessite d'être redessiné, le backend va l'envoyer dans un *damageRecorder*, cette structure permet de sauvegarder les éléments modifié nécessitant une mise a jour de l'état graphique.

Le *damageRecorder* est récupéré par le Root, qui est la passerelle entre les backends et le *MiniWorld* qui le gère et l'envoie au *MiniWorld* afin de le traiter.

Ainsi chaque événement est traité par son backend et la reconstruction d'élément par le *MiniWorld*.

Voici un schéma reprenant le chemin du *damageRecorder* :

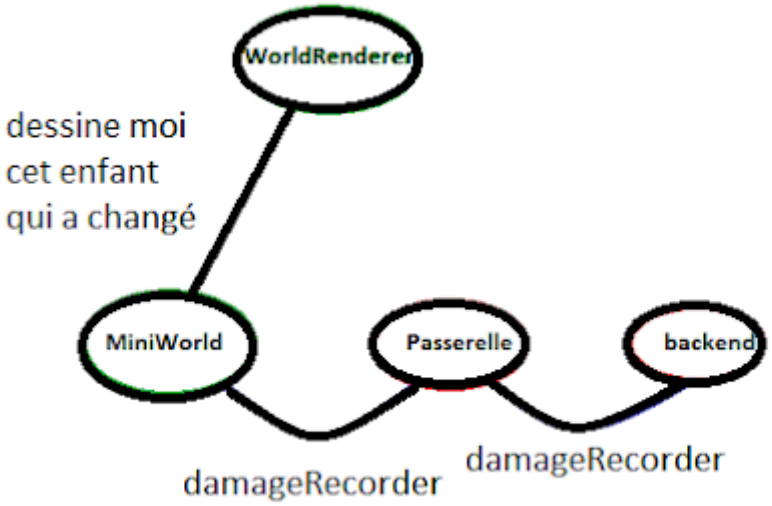

Fig 9 Fonctionnement du dessin du *damageRecorder*

Grâce à cette solution on ne dessine que partiellement et uniquement lorsqu'un changement à lieu. On ne dessine plus en continue on à donc une demande de puissance bien plus faible.

Ainsi notre *MiniWorld* correspond au cahier des charges, il est peu gourmand en performance, il est générique et minimal car il délègue principalement au backend.

7.2.2 La gestion d'événements

Chaque backend produits des événements différents. Tout ces événement doivent être transformé en événement Morphic. Ma solution est donc d'utiliser le polymorphisme afin que les événements peuvent être compris par Morphic. Ainsi tout les types d'événement sont géré par une instance de *MiniWorld*.

7.3 La gestion des fenêtres

Afin de gérer les fenêtres natives il faut :

- avoir la possibilité d'ouvrir / de fermer des nouvelles fenêtres natives

- avoir la possibilité de sauvegarder l'état de toute les fenêtres natives.

- pouvoir rouvrir les fenêtres natives à l'ouverture

Lorsqu'on veut créer une fenêtre native, on stocke certaines informations afin de pouvoir la gérer de manière programatique, tout le rendu graphique passe par le *MiniWorld* ainsi en le gardant on peut garder la main sur la fenêtre.

7.3.1 Ouverture / Fermeture

Dans les fenêtres simulés de Pharo on peut retrouver deux type de fenêtres, celle normal auquel on peut interagir comme on le souhaite, et celle modal qui bloque le système tant que l'on a pas fait une action spécifique. Ces fenêtres modal s'ouvre de base sur une fenêtre simulé. Il faut donc pouvoir ouvrir des fenêtres basiques et des fenêtres plus complexes.

Ma solution est que lorsque l'on veux ouvrir une fenêtre natives, on demande au WindowManager, en lui précisant ou non le backend que l'on veut utiliser. Lorsqu'on lui demande d'ouvrir une fenêtre native on lui fournis l'outil qu'on souhaite utiliser. Lorsque le système veux ouvrir une fenêtre modal il devra demander dans quel World il se trouve afin de s'ouvrir au bonne endroit.

Voici un schéma de comment fonctionne une ouverture de fenêtre.

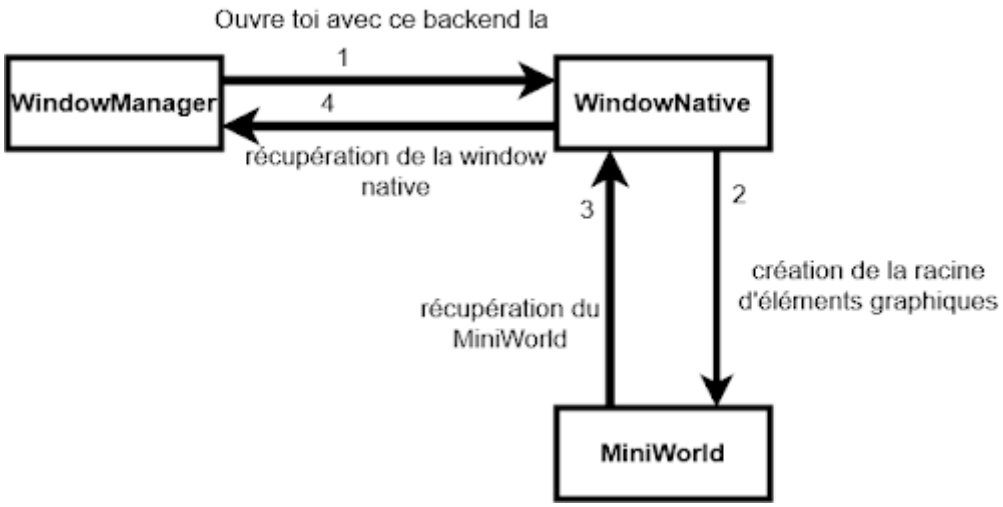

Fig 10 Comment fonctionne l'ouverture

Les instances de Window sont sauvegardé dans une Liste affin de pouvoir les appelées pour les gérer de manière programmatique et de garder un ordre.

Voici un exemple de comment les gérer de manière programmatique :

wm := Windowmanager new.

wm openNewWindowWithRoot : (ClyFullBrowserMorph onDefaultEnvironment) .

wm allWindow first minimize

Cela permet que la fermeture se passe de la même façon que de l'ouverture. Le WindowManager reçoit l'événement de la fermeture, puis il envoie à l'instance de Window identifié la demande de fermeture, par la même occasion il la retire de la liste.

7.3.2 La TaskBar

La TaskBar est un élément centrale de la fenêtre natives par défaut, c'est dans celle-ci que l'on voit apparaître les fenêtres simulé et sur laquelle on peut effectuer des action comme minimiser, maximiser, fermer, ect. Les fenêtres simulés sont ajoutés à une taskbar par le World, via des items impléméntés directement dans la classe SystemWindow

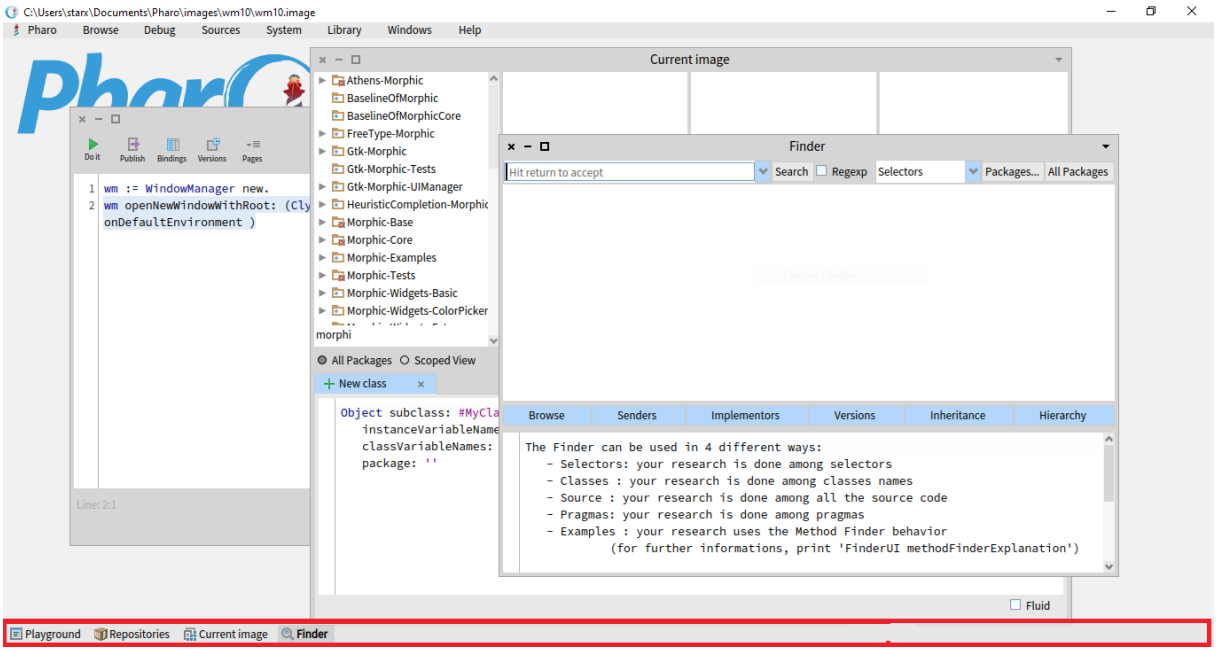

Figure 11, La taskbar dans Pharo

On retrouve cette taskbar encadré en rouge dans la figure 14.

Ma solution est d'ajouter une taskBar de la même manière que celle par défaut, cependant au lieux d'avoir des fenêtre simulés nous auront des fenêtres natives. En ce qui concerne les items, pour ne pas répéter le code ma solutions est d'utiliser un trait, un trait est un concept qui permet d'ajouter un ensemble de méthodes à une classe.

A chaque création de fenêtre natives, le WindowManager ajoute un item à cette taskbar en interrogeant la Window pour obtenir sont item.

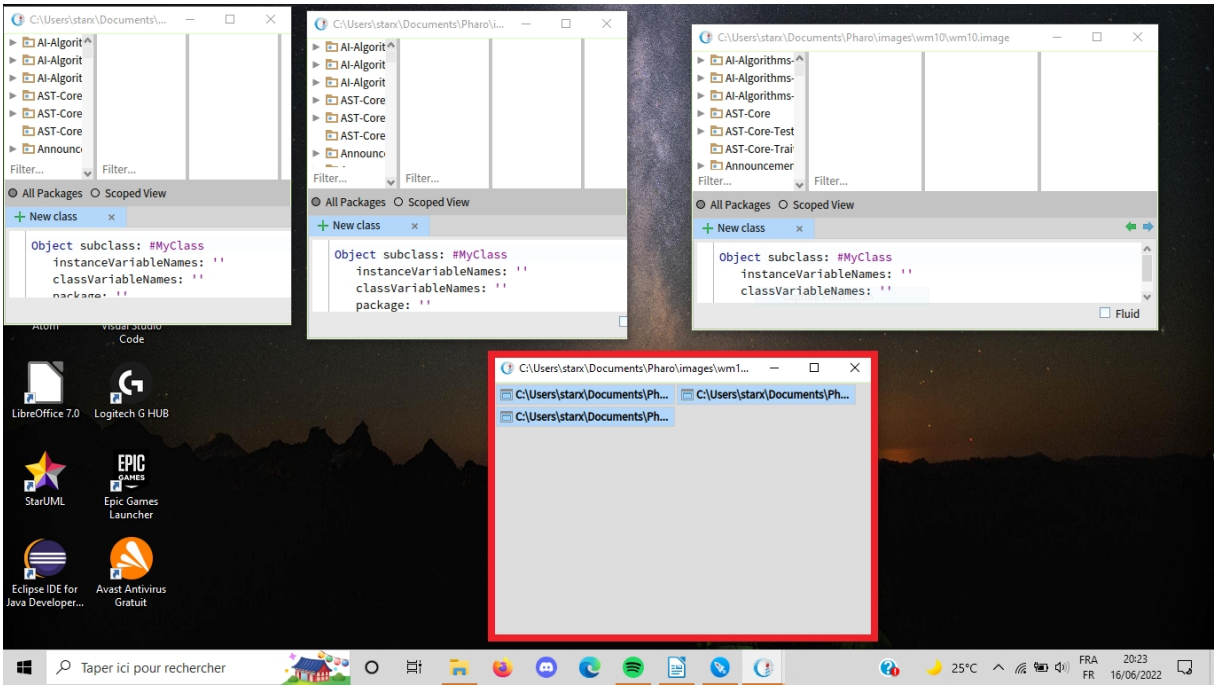

Fig 12 TaskBar de la fenêtre native maintenant

Dans la figure 15 on retrouve la taskbar dans la fenêtre native qui est encadré en rouge.

7.3.3 StartUp / ShutDown

Une fonctionnalité de l'IDE est l'ouverture des fenêtres dans l'état où on les as quitté. Il faut donc sauvegarder les fenêtres simulés et les natives.

Ma solution est que cette fonctionnalité est prise en charge par le WindowManager, ce dernier sauvegarde les attributs des fenêtres avant la fermeture et les ouvres avec ces attributs après.

Shutdown : on conserve en un premier temps les outils de chaque fenêtres native ouvertes ce qui permet de conserver le contenant principal. On sauvegarde ces outils dans une liste.

On conserve ensuite les attributs des fenêtres natives, les attributs de la fenêtre sont rassemblés dans une instance de la classe *OSWindowAttributes*. Cette classe contient la taille, position, maximisé, minimisé etc. On conserve ces attributs dans une liste afin que chaque attribut soit à la même place que l'outil associé à sa fenêtre native.

Une fois le tout sauvegardé on peut fermé ces fenêtres.

Startup : on récupère les outils et attributs sauvegardé. On demande au WindowManager d'ouvrir des fenêtres natives avec ces attributs.

Voici le code de comment cela fonctionne :

```
wm := WindowManager new.
wm openNewWindowWithRoot: (ClyFullBrowserMorph onDefaultEnvironment ).
wm saveAllRoot .
wm saveAllAttributes .
wm deleteAll .
wm startupOpen
```

8 Conclusion

Pour conclure, lors de ce stage j'ai contribué à l'ajout d'outils dans l'IDE Pharo via les sprint. J'ai contribuer à la création de fenêtres natives et à commencer la migration entre l'ancien IDE et le nouveaux.

Mes outils sont utilisés par plusieurs personnes depuis peux, cela permet de pouvoir faire avancer la solution plus vite en découvrant de nouveaux besoins et problème dans le code existant.

J'ai eu la chance de découvrir beaucoup d'outils dans Pharo grâce au présentation de l'équipe, « Workshop » et aux réunions avec Guillermo.

Mon stage ne s'arrête pas ici, il faudra faire par la suite l'intégration avec le framework *`spec`*, faire l'intégration avec un nouveau schéma de rendu vectoriel. Puis faire du benchmark afin de valider la performance de mes travaux .

9 Bilan

Lors de ce stage, j'ai acquis beaucoup de savoir. Ces acquis sont dans la continuité de mon projet professionnelle est scolaire. Devenir un développeur au service des autres afin de les aider. Ainsi avec L'équipe Rmod j'ai appris beaucoup de chose vis a vis du Génie Logicielle qui est mon souhait pour la poursuite d'étude afin de perfectionner mes bonnes pratiques et de connaître de nouvel technologie, des nouveaux langages. Cela ma permis aussi d'aprendre plusieurs façon de penser, de résoudre des problèmes. Tout ce que j'ai appris je peux ainsi maintenant le réutilliser en fonction du besoin.

9.1.1 Dévellopement

Au niveaux du développement, j'ai consolidé mes acquis, comme par exemple avec les design patterns ou bien encore le polymorphisme et j'ai appris de nouvelles choses. Voici mes principaux acquis au niveau développement.

Templates methodes: Lors de mon implémentation j'ai découvert les templates méthodes, ce patron de conception ma était utiles pour éviter la duplication de code lorsque les méthodes pour chaque backend se ressemblé énormément, à différence d'une ou deux ligne.

TDD : Lors de mon implémentation j'ai pratiqué le Test driven development (TDD), cela ma permit de m'assurer que la méthode que je voulais définir faisait exactement ce que je voulais.

Mock : Lors de mes tests j'ai eu le besoin de faire un mock d'une fenêtre afin de tester cette dernière. Ainsi cela ma permit d'approfondir dans cette méthode de test rencontré en COO.

9.1.2 Projet open source

Ce stage fut aussi ma première approche dans un projet open source. Cela ma appris à développez avec une communauté. Ainsi j'ai appris :

Intégration continue : j'ai appris à quelle point une intégration continue est importante cela nous permet de coder chacun de notre cotés et vérifier que l'on ne casse le code de son voisin

Pull Request : j'ai réaliser ma première pull request, ceci permet d'apporter une modification à un gros projet tout en ayant un avis extérieur d'expert.

Issues : J'ai écrit pour la première fois des issues, cela permet de transmettre à l'équipe de développement qu'un bug existe quelque part et comment faut il faire pour le reproduire.

9.1.3 Équipe de recherche

Faisant mon stage dans une équipe de Recherche, j'ai appris beaucoup via le coté Humain que ce soit dans les « Sprint » qui se déroule tout les dernier vendredi de chaque mois. Toute l'équipe se réunit afin de résoudre des issues en faisant du Pair Programming. Cela permet de rencontrer de nouveaux membres de l'équipe avec qui nous ne somme pas habitué à travailler, de nouvel culture, et différente façon d'aborder un problèmes. Via ceci j'ai appris

Pair Programming : le Pair Programming est un moyen de programmer à 2 sur une seul machine afin de résoudre un problème, cela ma permit d'apprendre à expliqué ce que je ressent face à un problème, avoir un lexique scientifique.

Anglais : Rmod est une équipe multiculturelle, ainsi la langue universelle qu'est l'anglais est très important. Lors de présentation ou de meeting j'ai pratiqué l'anglais ce qui ma permis de m'améliorer drastiquement

10 Bibliographie

Inria & son eco-systéme . URL: https://www.inria.fr/fr/innovation-numeriqueecosysteme (visité le 06/10/2022).

RMoD. URL: https : / / rmod . gitlabpages . inria . fr / website/ (visité le 06/10/2022)