

LIMBALLE Pierre - PEREZ Quentin

UFR-ST - Licence informatique 3ème année
Groupe TP 2B

2 janvier 2016



Méthodes et Outils pour la Programmation Mini Projet 2015

Rapport



Contacts : pierre.limballe@edu.univ-fcomte.fr, quentin.perez@edu.univ-fcomte.fr
Professeurs : GREFFIER Françoise, BONNEVILLE François

Table des matières

1	Sujet	2
1.1	Contexte	2
1.2	But de l'application	2
2	Conception	3
2.1	Introduction	3
2.2	Structures de données au sein de l'application	3
2.2.1	Modélisation d'un astre (Etoile ou Satellite)	4
2.3	Structure générale de l'application	4
2.3.1	Emplacement des fichiers	4
2.3.2	Architecture MVC	6
2.4	Les algorithmes intéressants	6
2.5	Points intéressants et supplémentaires du programme	6
3	Conclusion	7
3.1	Bilan	7
3.2	Les acquis	7
3.3	Améliorations futures	7

1 Sujet

1.1 Contexte

Dans le cadre du module : Méthodes et Outils pour la Programmation (MOP) la réalisation d'un projet nous a été demandé afin de mettre en application les connaissances théoriques acquises tout au long du semestre 5.

La conception de l'application a pour objectifs :

- l'utilisation du langage de programmation Java en orienté objets
- l'application du paradigme MVC (Modèle - Vue - Contrôleur)
- l'approche du travail collaboratif par l'utilisation de logiciels de gestion de versions
- la manipulation d'éléments graphiques de la bibliothèque graphique Java Swing

1.2 But de l'application

La finalité de ce projet est la conception Java une application qui réalise une animation graphique en 2D représentant un ensemble de planètes et leurs satellites en orbite autour d'une ou plusieurs étoiles. L'utilisateur doit disposer des fonctionnalités suivantes :

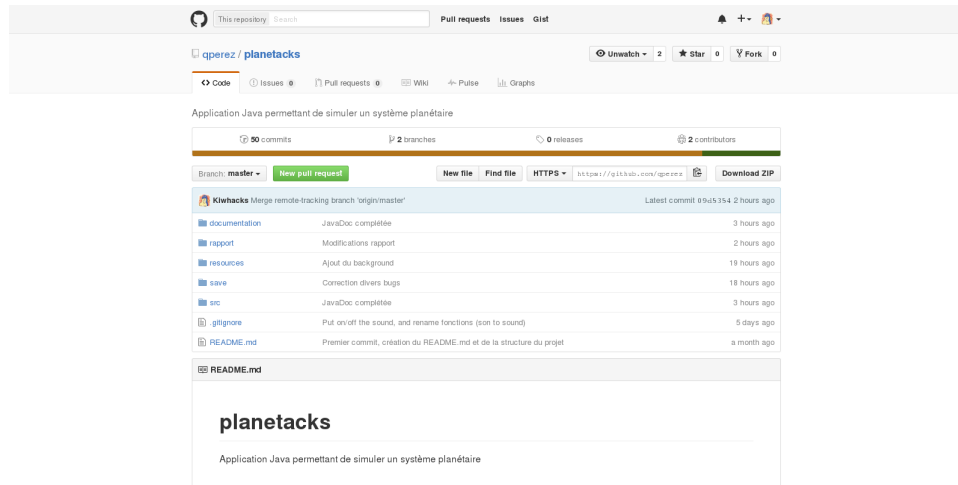
- l'ajout et la suppression d'astres (étoiles ou satellites) dans l'application
- l'affichage des astres dans une fenêtre graphique animée
- la possibilité de sauvegarder un système planétaire

La contrainte d'ergonomie est également importante quant à l'utilisation du logiciel.

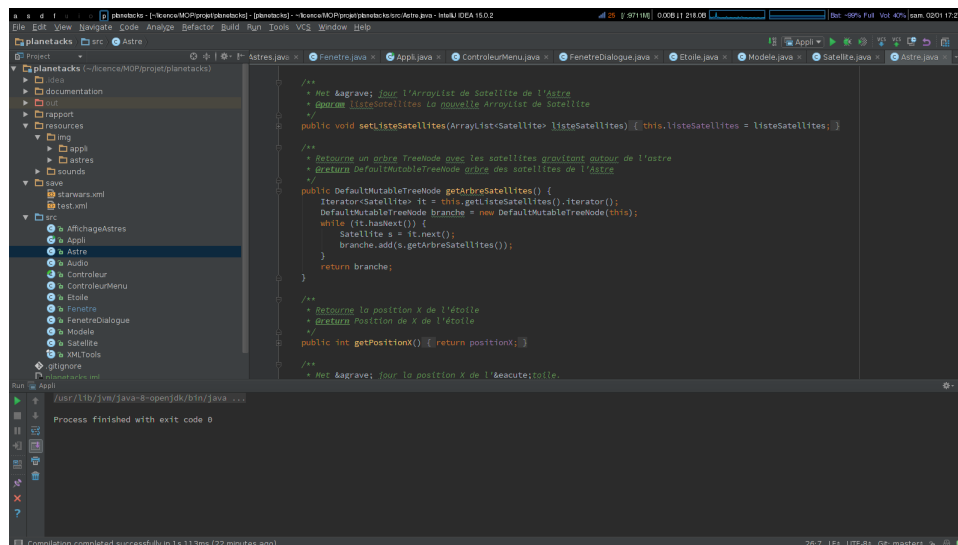
2 Conception

2.1 Introduction

Planethacks est le nom choisi pour l'application. Du fait d'un travail en binôme, l'utilisation d'un logiciel de gestion de version fut privilégiée. Nous avons pour cela utilisé Git, en lien avec le serveur d'hébergement GitHub (sources GitHub de Planethacks : <https://github.com/qperez/planetacks>).



L'IDE (*Internal Development Environment*) IntelliJ IDEA fut utilisé durant toute la phase de développement et de tests et ce afin d'accélérer et faciliter le développement de l'application ainsi que son débogage.



2.2 Structures de données au sein de l'application

La définition de classes permettant de structurer les données de notre programme fut la première étape de conception. Ainsi 3 classes nous permettent de modéliser un astre :

- Astre (classe abstraite)
- Etoile
- Satellite

Planethacks repose sur une conception basée sur le paradigme MVC. Ce paradigme impose l'utilisation à minima de 3 classes (Modèle - Vue - Contrôleur) permettant de séparer les données

de l’affichage et des interactions utilisateur.

Dans un soucis de respect du modèle MVC nous disposons donc des classes suivantes :

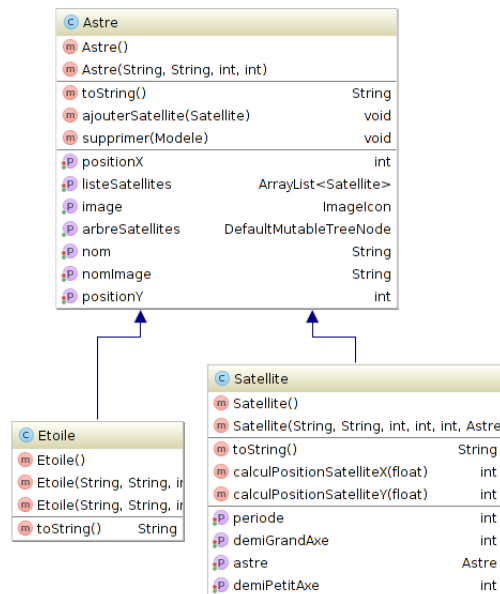
- Fenetre
- Modele
- Controleur (classe abstraite)
- ControleurMenu
- AffichageAstres

Afin de réaliser la fonctionnalité de sauvegarde une classe spécifique contenant les outils nécessaires fut créée : XMLTools.

Le point d’entrée de l’application est assuré par la classe Appli.

2.2.1 Modélisation d’un astre (Etoile ou Satellite)

Le diagramme UML ci-dessous représente la modélisation choisie pour un astre :

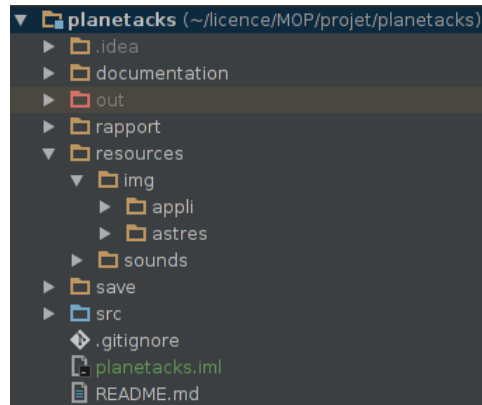


Les méthodes *calculPositionSatelliteX(float)* et *calculPositionSatelliteY(float)* présentes dans la classe *Satellite* permettent de calculer la position en X et en Y du *Satellite* en fonction du temps. La méthode *ajouterSatellite(Satellite)* permet de mettre en orbite un satellite autour d’un astre. Enfin, la méthode *supprimer(Modele)* permet de supprimer récursivement un astre et ses satellites à partir du modèle donné.

2.3 Structure générale de l’application

2.3.1 Emplacement des fichiers

Nous avons utilisé une structure de fichiers qui nous paraissait être la plus simple et la meilleure pour le fonctionnement de l’application. Ci-dessous, une représentation de l’arborescence :



Nous pouvons retrouver les dossiers de production suivants, dans lesquels nous manipulons directement les fichiers :

- **src**
→ dossier des sources (.java)
- **ressources**
→ dossier des différents fichiers image ou audio
- **img**
→ dossier contenant toutes les images du projet
 - **appli**
→ dossier des images de l'application en général
 - **astres**
→ dossier des images des astres (.png)
- **sounds**
→ dossier contenant tous les sons du projet
- **rapport**
→ dossier contenant les fichiers permettant de réaliser ce rapport (fichiers $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$)

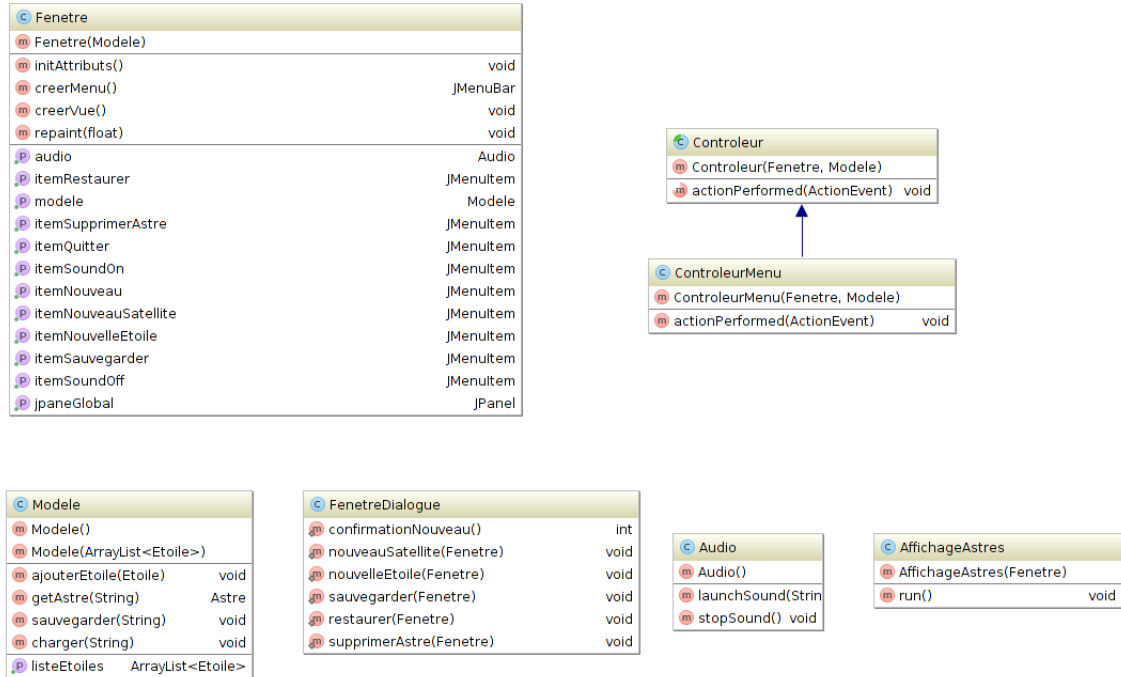
Mais également des dossiers dont le contenu est généré automatiquement :

- **out**
→ dossier des fichiers java compilés (.class)
- **save**
→ dossier des fichiers de sauvegarde générés par l'application (.xml)
- **documentation**
→ dossier contenant la documentation du projet (.html)
- **.idea**
→ dossier contenant les fichiers de configuration du logiciel *IntelliJ IDEA*

Ainsi que des fichiers seuls :

- **.gitignore**
→ fichier permettant d'ignorer l'envoi de fichiers sur GitHub (ex : dossier *out*)
- **README.md**
→ fichier Markdown de présentation du projet sur GitHub
- **planetacks.iml**
→ fichier de configuration du projet pour *IntelliJ IDEA*

2.3.2 Architecture MVC



2.4 Les algorithmes intéressants

Algorithm 1 méthode repaint de la classe Fenetre

```

function REPAINT(t temps)
    Supprimer l'ensemble des composants graphique du JPanel
    for chaque Etoile e de la liste d'étoiles du Modele do
        jlabastr ← nouveau JLabel avec l'image de e
        Afficher jlabastr en fonction de la position  $X_e$  et  $Y_e$ 
        Ajouter le jlabastr au JPanel
        for chaque Satellite s de la liste des satellites de e do
            jlabsat ← nouveau JLabel avec l'image de s
            Afficher jlabsat en fonction de  $s.calculPositionSatelliteX(t)$  et  $s.calculPositionSatelliteY(t)$ 
            Ajouter le jlabsat au JPanel
        end for
    end for
    Appel de la méthode repaint() sur le JPanel
end function
  
```

2.5 Points intéressants et supplémentaires du programme

3 Conclusion

3.1 Bilan

3.2 Les acquis

3.3 Améliorations futures