Emmanuel BABALA COSTA & Sybille CRIMET
M1 Informatique Groupe 2

Guide Développeur

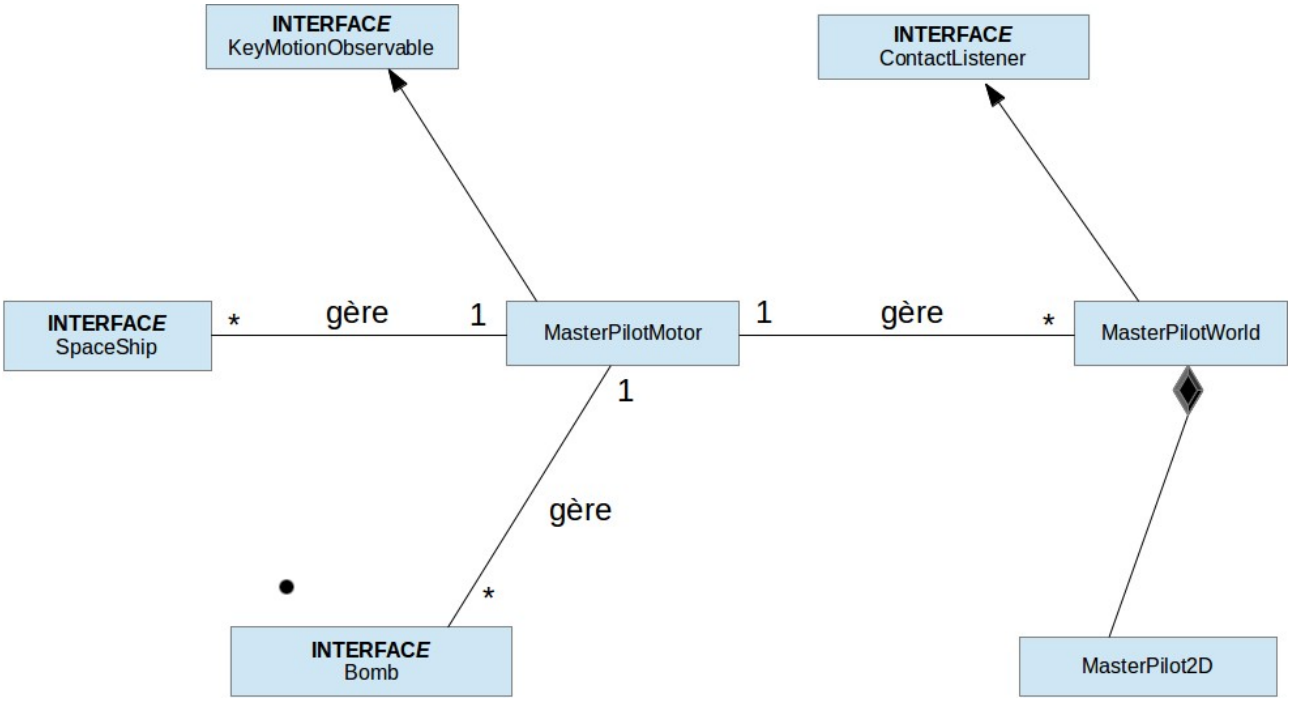
Master Pilot

Emmanuel BABALA COSTA & Sybille CRIMET
M1 Informatique Groupe 2

Sommaire

Emmanuel BABALA COSTA & Sybille CRIMET
M1 Informatique Groupe 2

Diagramme UML général du projet.



Emmanuel BABALA COSTA & Sybille CRIMET
M1 Informatique Groupe 2

Choix d'implémentation

Comme vous avez pu le constater d'après le diagramme UML, le projet est divisé en plusieurs sections. Ces sections correspondent à des packages.

Les packages :

MasterPilot:

Il contient la logique générale du jeu. C'est lui qui charge le jeu depuis la config xml et peuple le monde en fonction, applique la logique des vagues, de la generation des planètes, des bombes et des ennemis.

Il contient une classe interne GameSpec qui se chargera de stocker les specificités du jeu chargé depuis le fichier de configuration.

Bomb:

Contient les fichiers nécessaires à la gestion des bombes et des méga bombes dans le jeu. On gere notamment la collision avec les ennemis, le sac du hero qui ne peut contenir qu'une bombe à la fois.

Common:

Contient les ressources part&gées par plusierus packages différents notamment tout ce qui concerne les collisions.

Graphic:

Contient tous les fichiers nécessaires pour gérer l'affichage du monde et des objets dans ce monde.

Parser:

Contient les fichiers relatifs au parsing des fichiers xml correspondant à la description des différents niveaux.

Emmanuel BABALA COSTA & Sybille CRIMET
M1 Informatique Groupe 2

Ship :

Correspond aux vaisseaux du jeu qui implementent toutes la mêmes interface spaceShip. Un TIE, Cruiser ou autre ennemi est simplement vu comme un SpaceShip et, par polymorphisme, on peut appeler la méthode doMove() qui appellera celle de la bonne classe, sans se soucier du type.

Le hero c'est l objet principal du jeu. Il implemente en plus de l'interface spaceship, le KeyMotionObserver afin d'etre notifier de l'action sur une touche. En fonction de la touche, il appelera la bonne action : espace pour tire, b pour tire une bomb, espace pour tire etc

Star:

Contient les fichiers relatifs à la création des planètes

world:

C'est le monde du jeu. Il possède plusieurs manager pour les différents objets qui le composent : ennemies, planetes, bombes et hero. Ces managers permettent au monde de connaître les objects présent, c'est pourquoi il a la responsabilité d' appeler les méthodes de la classe MasterPilot2D (méthode de dessin des formes des objects du monde).

Dans sa méthode draw il va itérer sur chaque object du monde present dans jbox et pour chacun d'eux appeler draw shape qui en fonction de leur forme appelera la bonne m éthode de MasterPilot2D pour les dessiner.

Afin d'etre à l'écoute des collisions, cette classes implementera aussi le ContactListener de jbox qui fournit les méthodes beginContact et endContact qui sont appelées respectivement en début et en fin de collision. C'est dans ces méthodes que l 'on récupère l'instance de l'object qui est entré en collision pour y appliqué une logique du comportement a suivre. Ces deux méthodes prennent en paramètre deux object du monde, contact A et Contact B qui entrent en collision, on se sert alors de leur methode getFixture pour réccupérer le champs userData qui possède le comportement à adopter de chaque objet du monde en cas de collision. Ce sont tous des implementations de l'interface UserSpec.Ensuite, la méthode onCollide effectue le comportement desire en cas de collision.

On réccupère la sensibilité au sens jBox2D ce qui permet de savoir si un objet est sensible aux collisions ou non. Si il est sensible alors on le dessine. La methode beginContact permet l'activation du bouclier lorque l'objet est sensible aux collisions et la récuperation des items.

Dans la méthode endContact, on vérifie via la méthode isDestroyable de userSpec si l'object doit être supprimé ou non. Si tel est le cas on le place dans la liste des objects à suprrimer du monde. C'est le moteur du jeu qui se chargera de les retirer ultérieurement du monde. Ceci à cause du fait que jbox2D bloque le monde pendant un callBack de collision.

La méthode getDestroyBody va renvoye la liste des objects du monde à detruire. Cette liste est remplie soit durant un callback de collision dans la méthode endContact, soit durant l'appel de la méthode elle même qui va intérroger le manager des traits du vaisseau et des balles tirées pour récupérer les objets qui doivent être retirés

Main;

Contient le fichier main qui permet le lancement du jeu.

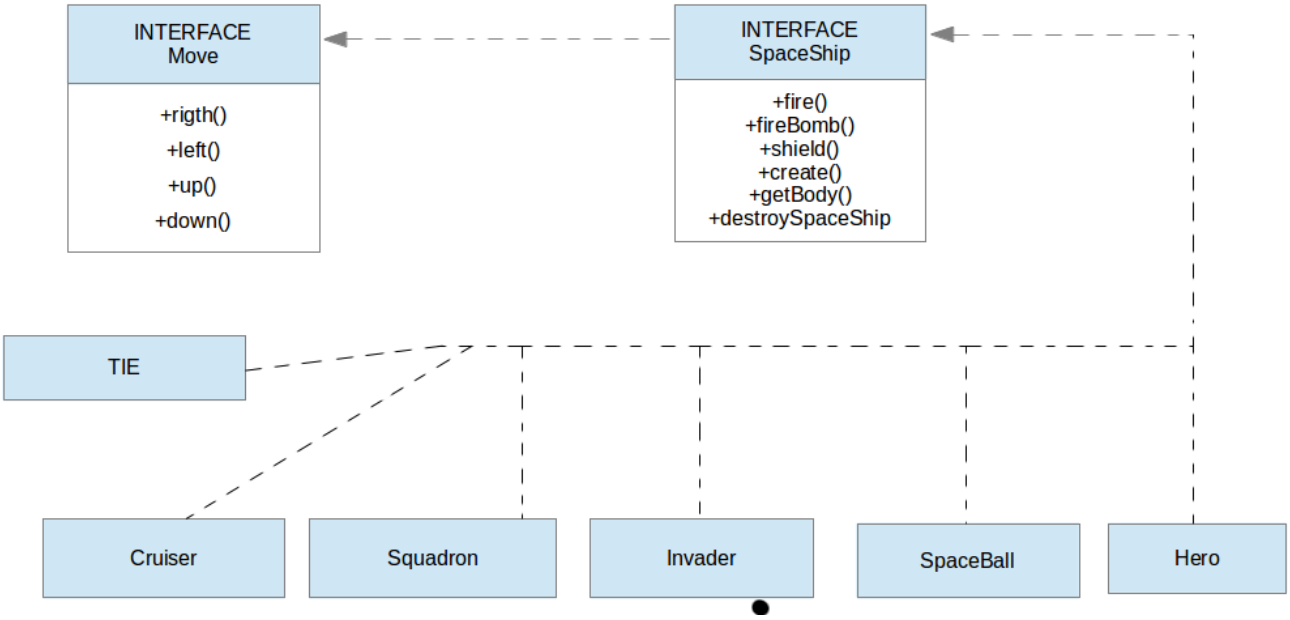
Emmanuel BABALA COSTA & Sybille CRIMET
M1 Informatique Groupe 2

Les Interfaces :

SpaceShip :

toutes les classes qui représentent des vaisseau implémente cette interface puisque qu'elle contient le méthodes générales de tir, de déplacement etc.

Voici comment se présente l'interface :

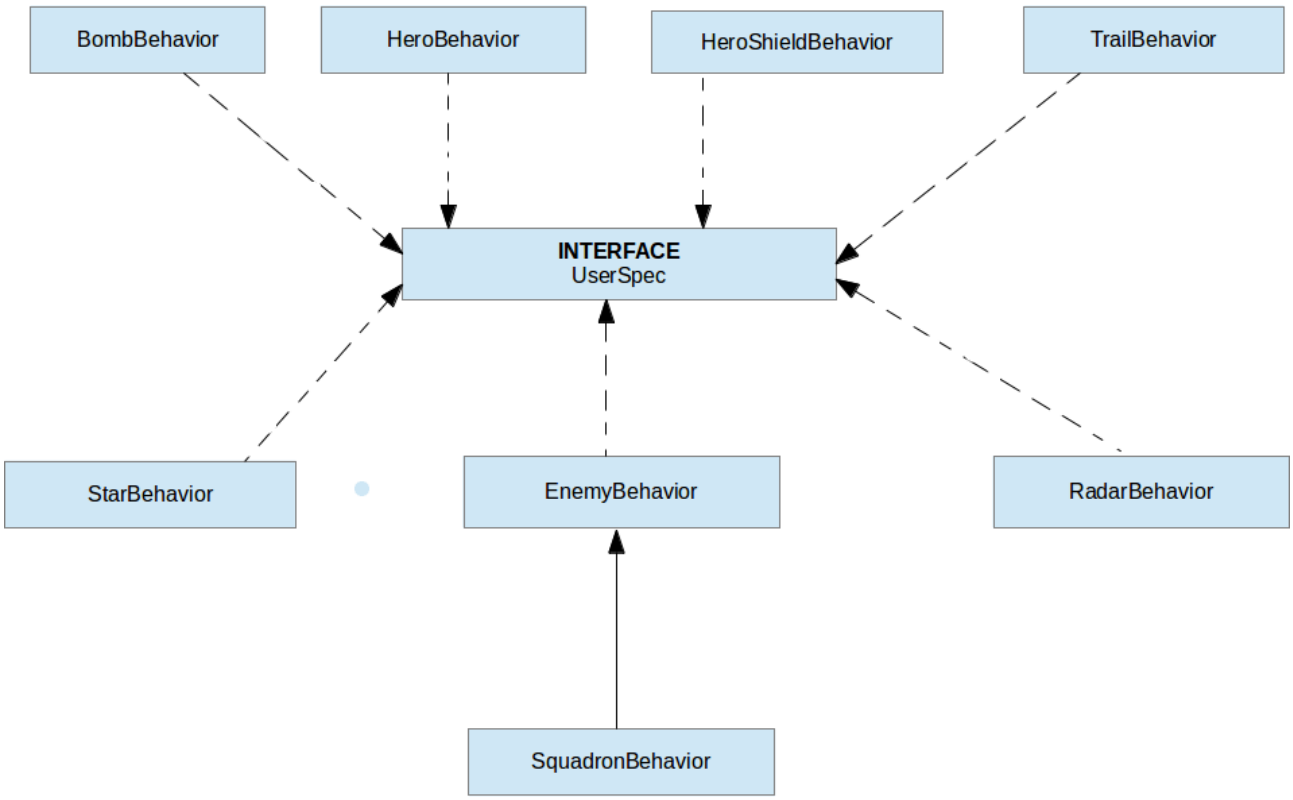


Emmanuel BABALA COSTA & Sybille CRIMET
M1 Informatique Groupe 2

UserSpec :

Heritent de cette interface tous les objects qui sont utilisés dans le champ userdata des fixtures de la librairie jbox2D. Ces objets ont pour suffixe commun bihavior. Cette interface est particulièremet utiles pour gérer les collisions.

Voici la réprésentation des liaisons qui sont faites avec cette interface :

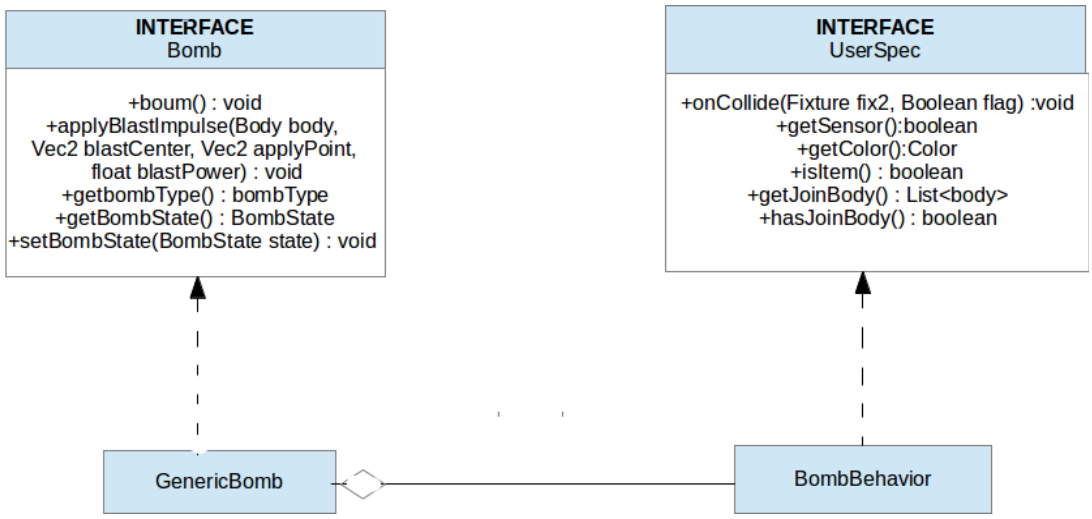


Emmanuel BABALA COSTA & Sybille CRIMET
M1 Informatique Groupe 2

Bomb :

Permet connaître le type de la bombe (implosive ou explosive) .

Voici comment se présentent l'interface :



Améliorations possibles

Nous aurions pu rajouter une barre de vie aux ennemis. Plutôt que d'afficher des formes géométriques, nous aurions pu charger des images pour afficher les planètes, le héro et les ennemis. Nous aurions pu aussi rajouter des spécificités aux planètes c'est-à-dir avoir des planètes de différentes couleurs, de différentes taille et de différente densité.

Conclusion

Ce projet s'est révélé intéressant puisqu'il nous a permi d'appprendre a nous organiser pour travailler en binôme d'une part, et d'autres parts, parce qu'il nous a permi de maitrîser plus largements les notions abordées en cours, notamment en ce qui concerne les lambdas et le design pattern pour avoir un code source plus facilement modulable.