Rapport de TAL: Behaviour Trees

Valentin LEBOUVIER

Encadrant: Catherine DEZAN

Table des matières

1 Introduction

Le Behaviour Tree (BT) est un outil qui a été créé pour modéliser les intelligences artificielles dans les jeux vidéos mais il est également utilisé dans la robotique. Le but principal de ce TAL était de voir si il était possible de créer une série de TP pour des L2 informatique en s'appuyant sur cet outil pour apporter un peu de thème et d'intérêt aux TPs proposés.

Dans ce rapport, nous allons tout d'abord décrire ce que sont les BTs, puis voir des exemples de BTs basés sur PacMan et enfin expliquer les TPs que j'ai préparé.

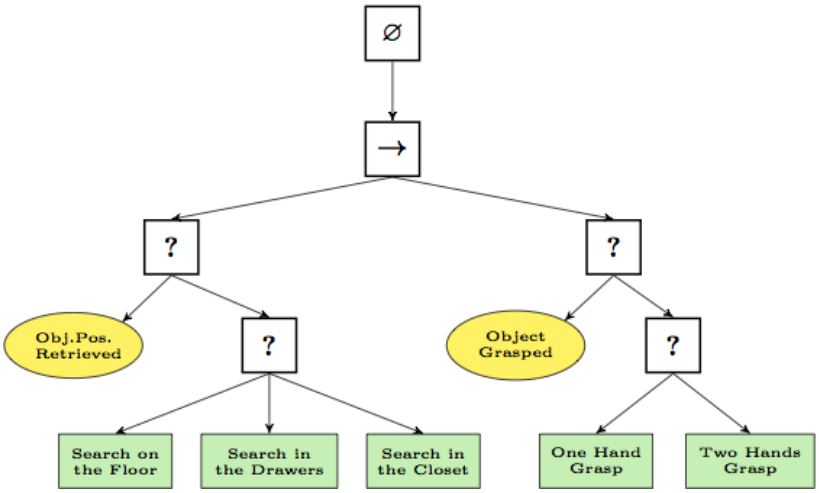

FIGURE 1 – Exemple de BT

2 Les Behaviour Trees (BTs)

Un Behaviour Tree (BT) est un outil permettant à un agent autonome de pouvoir choisir l'action à réaliser. Il est utilisé pour créer des scénarios que suivront ces agents autonomes. Il sert à la fois d'outil de conception et d'éxécution. Le fait qu'il puisse être représenté graphiquement et même manipulé graphiquement dans certains environnements (Unity) en fait un outil très intuitif et rapide de compréhension.

Les BTs sont venus du besoin dans le monde du jeu-vidéo d'un outil de gestion des comportements plus performant que ceux qu'ils utilisaient. Les BTs ont donc été créés dans le but de répondre aux problèmes de mission planning. Ainsi, les BTs condensent certaines forces qui manquaient à d'autres outils. Ainsi ses points forts sont sa modularité et sa réactivité, notions qui seront revues plus loin dans ce document.

2.1 Définition technique

2.1.1 Définition

Un BT est un graphe orienté acyclique dont les noeuds retournent Succès, Échec ou En Cour selon les règles associées à ces noeuds. Le status de chaque noeud est évalué par un *tick* envoyé depuis la racine et qui parcourt le graphe selon les règles de parcours de chaque noeud. Les noeuds ré-évaluent tous leurs enfants (selon leurs règles) à chaque fois qu'ils sont tickés. Il y a de nombreuses définitions de ce que peuvent être les noeuds d'un BT. Il faut aussi noter que dans certains modèles, un quatrième état peut être retourné par un noeud en cas de problème et retourner une exception. La définition que je présente ci-après est celle de la bibliothèque py_{trees} qui sera utilisée lors des TPs avec quelques modifications. Ainsi il y a trois types de noeuds : les noeuds d'éxécution, les composites et les décorateurs.

2.1.2 Les noeuds

1. Les noeuds d'éxécution

   Les noeuds d'éxécution sont les feuilles du BT, ce sont eux qui interagissent avec le modèle.

— Action :

FIGURE 2 – Noeud Action

Le noeud action retourne l'état dans lequel est l'action au moment du *tick*. L'action peut donc être en cours et retourner En Cours. Elle peut être terminée correctement et retourner Succès. Ou alors elle peut s'être terminée incorrectement et retourner Échec.

— Condition :

FIGURE 3 – Noeud Condition

Le noeud condition retourne l'état du test qui lui a été confié. Ainsi la condition ne peut retourner que Succès si le test est vrai et Échec si le test est faux.

2. Les composites

   Les noeuds composites sont les noeuds de routage du BT, ils décident quelle branche prendre lors de l'évaluation du BT. Les noeuds enfants sont toujours évalués du plus à gauche au plus à droite, sauf pour les noeuds parallèles qui les évaluent tous en même temps. Les noeuds composites les plus courants sont les suivants :

— Séquence :

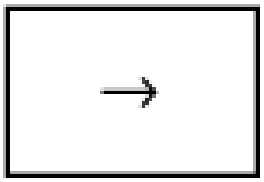

FIGURE 4 – Noeud Séquence

Le noeud séquence n'évalue l'enfant suivant que si le précédent a retourné Succès, sinon il renvoie l'état de l'enfant courant. Ainsi, ce noeud ne retourne Succès que si tous les enfants ont retourné Succès lors de ce *tick*.

— Sélecteur :

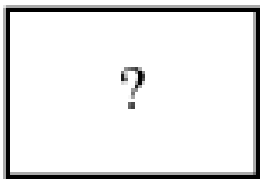

FIGURE 5 – Noeud Sélecteur

Le noeud sélecteur renvoie Succès dès qu'un enfant retourne Succès, sinon il retourne En Cours tant qu'il y a encore au moins un enfant qui retourne En Cours.

— Parallèle :

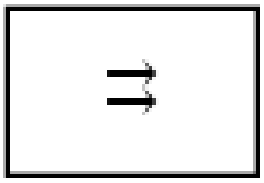

FIGURE 6 – Noeud Parallèle

Le noeud parallèle envoie un *tick* à tous ses enfants en même temps. Le status qu'il renvoie dépend alors du nombre d'enfants qui retourne Succès ou Échec. Les seuils M et N sont donnés à la création du noeud et représentent le nombre d'enfants devant retourner Succès (respectivement Échec) pour que le noeud parallèle retourne Succès (respectivement Échec). Dans le reste des cas, le noeud retourne En Cours.

— Composites avec mémoire :

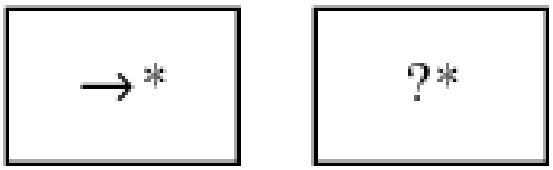

FIGURE 7 – Composites avec mémoire

Les noeuds composites avec mémoire fonctionnent comme leurs équivalents sans mémoire à l'exception prêt qu'ils se rappellent des résultats précédents donnés par les enfants. Ainsi ils ne tickent pas les enfants qui ont déjà été évalués.

3. Les décorateurs

FIGURE 8 – Noeud décorateur

Les décorateurs sont des noeuds ne possédant qu'un seul enfant. Leur but est de modifier le résultat de cet enfant selon les règles que l'on choisit. Pour donner quelques exemples tirés de la librairie :

— Inverter :

Ce décorateur inverse le résultat reçu de l'enfant, ainsi : Succès devient Échec, Échec devient Succès et En Cours ne change pas.

— Condition :

Ce décorateur attend un certain état de l'enfant pour passer à Succès, avant il retourne En Cours.

— EternalGuard :

Ce décorateur met une condition sur l'éxécution d'un sous-arbre. Si la condition devient fausse, le sous-arbre se trouve arrêté.

2.2 Avantages des BTs

2.2.1 Modularité

Pour qu'un BT soit modulaire, sa racine se doit de ne pas renvoyer Succès ou Échec lorsqu'il tourne mais seulement lorsqu'il a terminé tout ce qu'il avait à faire. Cette modularité permet de pouvoir réutiliser ces BTs sans avoir à les redévelopper pour d'autres applications.

2.2.2 Réactivité

La réactivité vient du fait que l'on puisse facilement interrompre une branche par une branche plus prioritaire (plus à gauche). Cela permet de changer de comportement en un *tick*. Certains modèles de BTs permettent également de lancer un signal d'arrêt aux branches qui ont été interrompues.

3 Exemple sur le PacMan

3.1 Contexte

La création des BTs sur le PacMan vient du besoin de tester les BTs sur un exemple assez simple. Et pour pouvoir tester ces BTs, un modèle de PacMan a été développé. La plupart de ces BTs ont été développés et implémentés avec des noeuds séquence. Or il s'est avéré que les noeuds séquence de la librairie sont en fait des noeuds séquence avec mémoire. Cela entraine que certains des premiers BTs ne fonctionnent pas exactement comme prévu mais le comportement reste tout de même proche de ce qui est attendu.

3.2 Un PacMan Pas à Pas

Le premier BT réalisé permet de se déplacer dans le labyrinthe de façon "déterministe" (figure 9). C'est-à-dire que pour chaque direction il vérifie s'il y a un mur, s'il n'y en a pas le PacMan fait un pas dans cette direction. L'ordre des directions ne changeant pas, le PacMan se retrouve donc toujours coincé dans un des coins du labyrinthe.

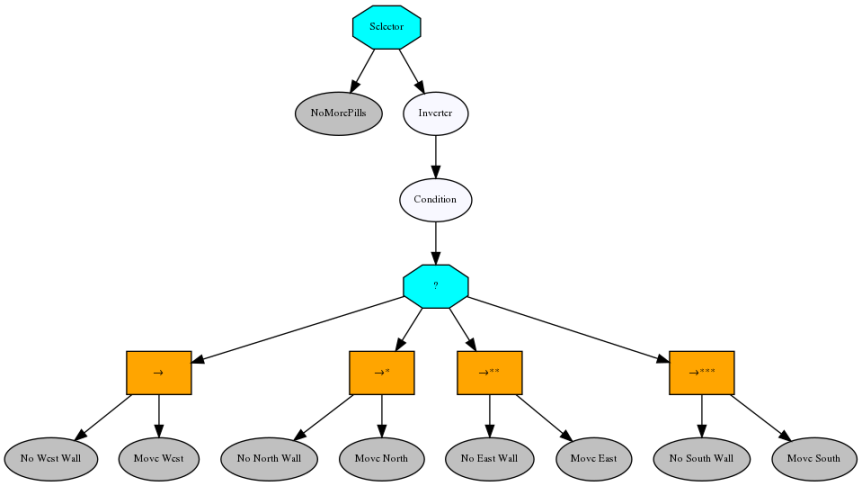

FIGURE 9 – PacMan déterministe

Un autre BT réalisé permet de choisir la direction dans laquelle se déplace le PacMan équiprobablement (figure 11). Ainsi, PacMan a autant de chance de se déplacer dans chacune des cases adjacentes possibles. Pour ce faire, la construction a été réalisée en suivant l'arbre binaire des possibilités de déplacement (figure 10) Et pour chaque feuille de cet arbre binaire, faire un choix parmi les directions disponibles.

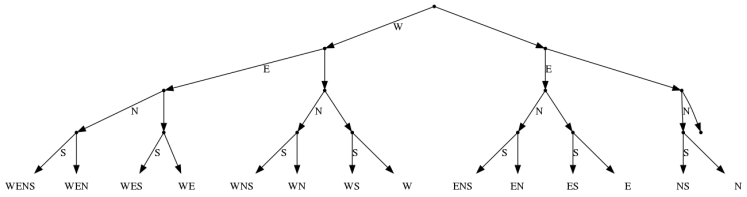

FIGURE 10 – Arbre des possibilités de déplacement

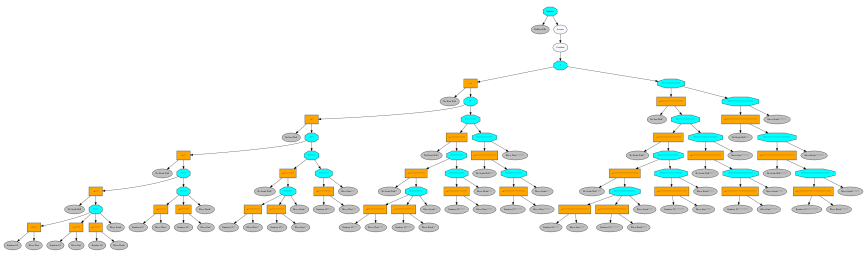

FIGURE 11 – PacMan équiprobable

Ces BTs possèdent une entête (figure 12) permettant de filtrer les résultats indésirables (Succès) et de ne laisser remonter que les Échecs. Pour ce faire, il y a un décorateur Condition qui se déclenche lors de la réception d'un Échec. Cela renvoie donc un Succès qui est alors inversé pour récupérer un Échec.

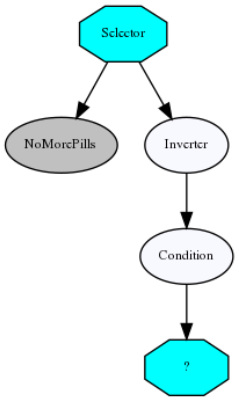

FIGURE 12 – Entête PacMan

Le modèle (figure 16) réalisé pour permettre l'éxécution de ces premiers BTs est des plus simples. Il est composé principalement de tests pour savoir s'il y a des murs autour du PacMan, et de fonctions faisant se déplacer PacMan d'une case.

3.3 Du Pas à Pas au Continu

Pour mieux simuler le comportement d'un PacMan, le modèle a été modifié pour accueillir des fantômes et faire en sorte que les déplacements soient continus (figure 17). Les BTs ne déplacent donc plus le PacMan mais lui disent dans quelle direction se diriger. Ainsi il y a deux fonctions qui envoient périodiquement des ordres au modèle, une dans le BT qui actualise les directions de déplacement et une dans le modèle qui éxécute ces déplacements. Cela a obligé à l'utilisation de threads, les deux fonctions étant bloquantes.

L'objectif avec ce nouveau modèle était de passer à un niveau d'abstraction plus haut pour le contrôleur et de se débarasser du contrôle direct des directions. Pour ce faire les fonctions chase, gotoPill et flee ont été ajoutées ainsi que le principe de vision des autres personnages. Le PacMan voit tous les fantômes (car il est supposé voir l'écran) alors que les fantômes ne voient PacMan que s'il passe dans le même couloir et se rapellent de la dernière position connue du PacMan en cas de perte de vision. Pour éviter que le PacMan ne se fasse poursuivre à l'infini, les fantômes oublient la dernière position de PacMan au bout de 10 pas. Pour pouvoir implémenter une recherche du plus court chemin pour gotoPill, un graphe du labyrinthe a été généré.

Plusieurs versions de BTs pour simuler une partie de Pacman ont été créés pour en arriver au BT suivant (figure 13). Malheureusement je n'ai pas réussi à trouver de façon rapide de faire fuir le PacMan, ainsi si il se trouve à moins de 10 cases d'un fantôme, il "panique" et se déplace aléatoirement.

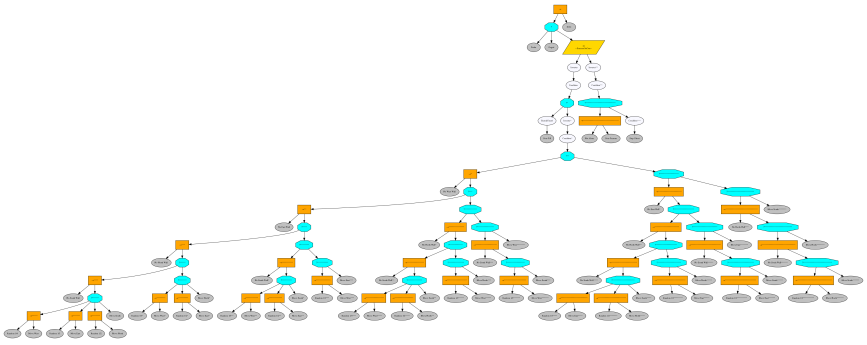

FIGURE 13 – BT continu

3.4 Un PacMan Avec Interface

Pour rendre le PacMan plus lisible, une interface tkinter lui a été donnée. L'interface est passée d'un affichage textuel (figure 14) à une fenêtre tkinter (figure 15). Cette interface demandant également une fonction bloquante pour fonctionner, elle a été d'abord mise dans un thread. Or tkinter ne supporte pas très bien de ne pas tourner dans le thread principal, ainsi l'interface s'éxécute maintenant dans le thread principal et le contrôleur dans un autre thread.

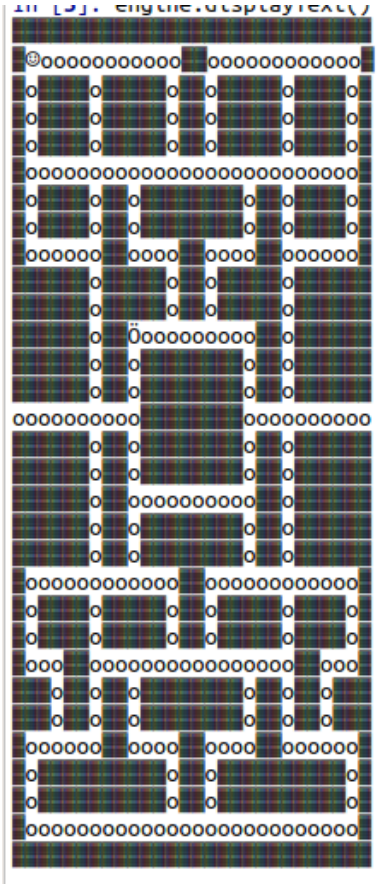

FIGURE 14 – Ancien affichage

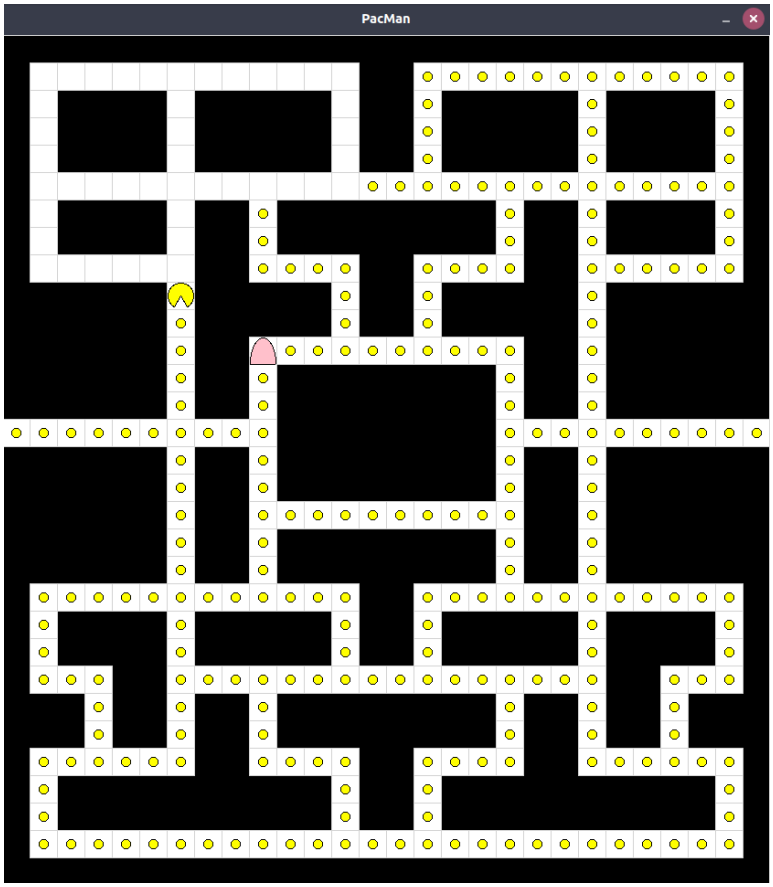

FIGURE 15 – Affichage tkinter

3.5 Un Modèle MVC correct

Pour pouvoir proposer d'utiliser ce modèle dans les TPs, il a été modifié pour correspondre à un modèle MVC (Modèle-Vue-Contrôleur). Ainsi plusieurs pattern Observateur ont été implémentés (figure 17).

4 TPs

4.1 Rappel Contexte

Ces TPs seront donnés lors du cours de python lors du second semestre de L2 informatique. Le but des TPs étant de familiariser les étudiants au language et à la Programmation Orientée Objet. Les BTs ne sont là que pour apporter une structure plus intéressante et attirante pour les élèves grâce à un thème qui touche à l'Intelligence Artificielle, les jeux vidéo et la robotique. Les trames de ces

4.2 TP1 : Graphes & BTs

Ce premier TP (figure 18) sert aux étudiants à se familiariser avec les graphes et à les manipuler en tant qu'objets python. Le TP se déroulerait comme suit :

- Les étudiants ont un rappel sur les graphes et les parcours en profondeur
- Les étudiants construisent une structure Noeud et écrivent la fonction qui leur permet de faire un parcours en profondeur
- Les étudiants sont présentés aux BTs et sont conviés à créer un exemple simple avec py_{trees}

4.3 TP2 : Tkinter

Ce second TP (figure 19) permet aux étudiants d'apprendre comment réaliser une interface avec tkinter. Le TP se déroulerait comme suit :

- Les étudiants ont un cours sur tkinter (Tk, Frames, Widgets (Boutons,...))
- Les étudiants créent une première fenêtre toute simple (Tk & un bouton pour fermer la fenêtre)
- Les étudiants ont un petit cours sur les Canvas
- Les étudiants mettent en place un Canvas pour afficher le background du PacMan

4.4 TP3 : PacMan (MVC)

Ce dernier TP (figure 20) permet aux étudiants de découvrir ce qu'est le pattern MVC ainsi que des utilités plus poussées du Canvas Le TP se déroulerait comme suit :

— Les étudiants sont présentés au PacMan via diagramme UML et on leur explique comment le modèle et le contrôleur fonctionnent
— Les étudiants ont ensuite à réaliser la vue avec des indications du TP

5 Conclusion

En ce qui concerne les BTs, ce sont des outils puissants de conception de scénarios pour des environnements où la prise de décision doit être rapide.

En ce qui concerne les TPs, un délai d'une semaine m'a été accordé pour finir l'écriture des TPs. Les documents en annexe ne sont que les trames plus ou moins complètes des futurs TPs.

6 Annexes

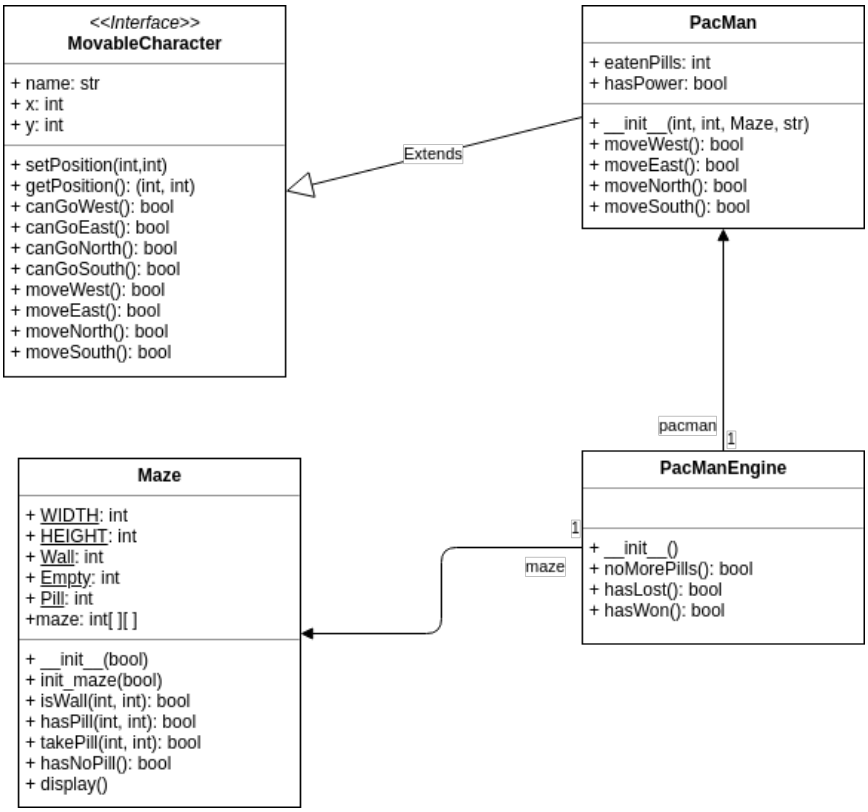

FIGURE 16 – UML PacMan Pas à Pas

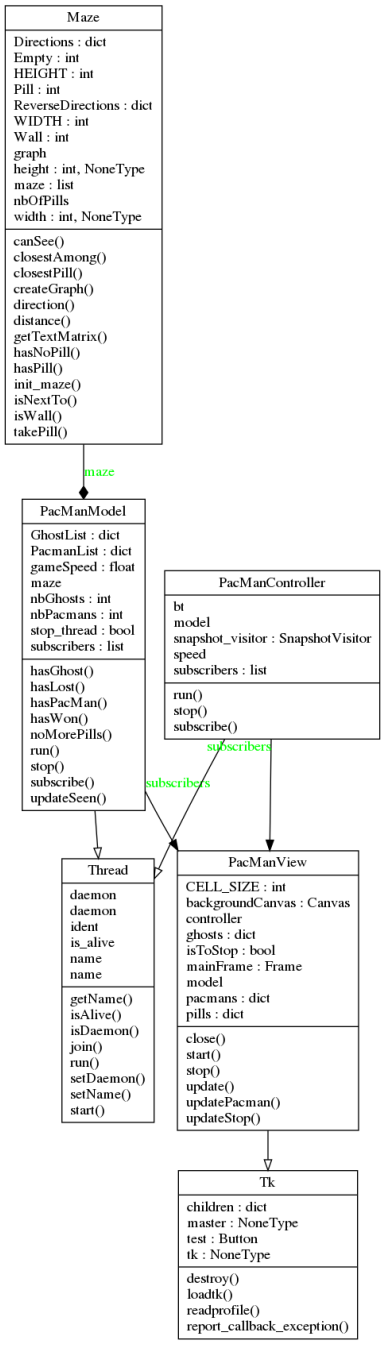

FIGURE 17 – UML PacMan MVC

TP1 : Graphes et Behaviour Trees

Valentin LEBOUVIER

17 juin 2019

1 Rappels sur les graphes :

Rappels sur les graphes : (non-)orientés, (a)cycliques
Parcours largeur, hauteur ...

2 Graphes simples :

Noeuds (init(name), addChild, addChildren, parcourProfondeur)
Avec affichage du nom du noeud lors de la visite en profondeur

3 Behaviour trees :

Présentation BT orienté IA/ jeux vidéo
Présentation des différents types de noeuds & fonctionnement global
Construction exemple simple avec py-trees

FIGURE 18 – TP1

TP2 : Tkinter

Valentin LEBOUVIER

17 juin 2019

1 Présentation Tkinter

Tkinter est un outil de représentation graphique facile à prendre en main et disponible dans la librairie standard Python.
Présentation Application (Tk)
Présentation Widgets(Button,Canvas)

2 Ma première fenêtre

Pour créer votre première fenêtre, il vous suffit d'importer tkinter avec la commande :

```
import tkinter as tk
```

Il vous suffit alors de créer une nouvelle application en créant un objet de type `tk.Tk` et de le lancer avec la méthode `.mainloop()`.
/ !\ Prenez bien note de lancer la `.mainloop` en dernier, car il est bloquant.
Vous pouvez ensuite commencer a modifier les paramètres de la fenêtre

- `.title(str)` : Change le titre de la fenêtre
- `.geometry(str)` : Change la taille de la fenêtre et son emplacement sur l'écran

Pour tout renseignement supplémentaire, vous pouvez utiliser `help()` dans la console.

3 Les canvas

Les canvas servent principalement à contenir des dessins. Chaque élément du Canvas pourra ensuite être déplacé dans le Canvas.
Pour dessiner dans le canvas vous pouvez utiliser les commandes suivantes :

- `create_arc()`
- `create_line()`
- `create_oval()`
- `create_rectangle()`
- `create_polygon()`

FIGURE 19 – TP2 page 1/2

TP3 : MVC

Valentin LEBOUVIER

17 juin 2019

1 MVC

(Explication du MVC)

2 Un PacMan revisité

Lors de ce TP vous aurez à réaliser une vue associée à cet ensemble modèle/controlleur : (diagramme de classe modèle/controlleur)

1

FIGURE 20 – TP3