PLAN DE DÉVELOPPEMENT

https://fr.123rf.com/photo_110580265_bangkok-thailand-august-4-2018-robot-controls.html?vti=o7aeg4akz9luvdjvtz-1-80

Initiation à l'intelligence artificielle 2021/2022

Belaikous
Dong
Parouty
Vallois

Sommaire

I/ Introduction **3**
II/ Plan de développement des classes et échéances **3**
ShrekCapteurs et ShrekMoteurs 3
ETATS 3
Shrek2 3

I- Introduction

Le plan de développement développé dans ce document vise à montrer les étapes à suivre pour mener à bien le projet. Il trace les grandes lignes des classes, attributs et méthodes à écrire et tester pour permettre de remplir les objectifs spécifiés dans le cahier des charges.
Concernant l'ordre de production des classes, l'écriture des classes se chevauche pour des raisons de gestion du temps mais l'ordre de commencement des classes a été établi de la manière suivante : les classes sur lesquelles d'autres classes s'appuient sont développées en priorité car contrairement aux autres elles ne dépendent pas d'éléments pas encore développés et peuvent être testées.

II/ Échéances pour les classes et leurs principaux attributs et méthodes

ShrekCapteurs et ShrekMoteurs

ShrekCapteurs : Prend en attribut des instances des classes shrekUltraSonic, shrekCouleurs et shrekTouch qui regroupent respectivement les méthodes relatives au capteur à ultrasons, au capteur de couleurs et au capteur de toucher. L'objectif est d'avoir accès à une classe regroupant les différentes méthodes relatives aux capteurs qui pourra être exploitée pour l'écriture de stratégies de plus haut niveau.
développée en semaine 1

ShrekUltraSonic *développée entre les semaines 1 et 8*
getDistance()
shrekFaceAuMur()

ShrekCouleurs *développée entre les semaines 1 et 8*
getCouleur()

ShrekTouch *développée entre les semaines 1 et 8*
estTouche()
touchDansLePasse()
updateStatus()

ShrekMoteurs: Prend en attribut des instances des classes shrekQuiBouge et shrekPinces. Ces deux classes regroupent sont respectivement les méthodes relatives aux déplacements du robot et aux mouvements des pinces. L'objectif est d'obtenir une classe permettant d'accéder à toutes les méthodes relatives aux capteurs et pouvant être utiles dans l'écriture de méthodes de plus haut niveau.
développée en semaine 1

ShrekQuiBouge *développée entre les semaines 1 et 8*
shrekAvance(double)
shrekEnAvant()
shrekIsMoving()
shrekStop()
shrekTourneGauche(double)
shrekTourneDroite(double)
shrekRotateWhile(double)
shrekMoveWhile()

ShrekPinces *développée entre les semaines 1 et 8*
openwhile(int)
shrekPincesEtat()
closewhile(int)
shrekOuvreLesPincesAvecPlaet()
shrekOuvreLesPincesSansPlaet()
shrekFermeLesPincesAvecPlaet()
shrekFermeLesPincesSansPalet()

ETAT

Cette classe possède comme attributs tous les états définis dans le diagramme d'états présenté dans le cahier des charges. *développée en semaine 8 à 9*

Shrek2

shrek2 : La classe shrek2 prend en attribut des instances des classes shrekMoteurs, shrekCapteurs et ETAT. Il s'agit de la classe qui définit des méthodes et des attributs à l'échelle du robot et non plus à celle de ses capteurs ou moteurs. Les méthodes définies ici s'appuient sur le répertoire de comportements rendus possibles par les méthodes préalablement définies à l'échelle des capteurs. Il y a également un attribut direction qui fait office de boussole, des méthodes relatives à celui-ci sont développées et sont aussi utilisées dans les stratégies de plus haut niveau. *développée entre les semaines 6 et 11*

Quelques méthodes à définir dans shrek2 :
correctionMur() : méthode qui en s'appuyant sur les méthodes de shrekUltrasonic et shrekQuiBouge permet de ne pas entrer en collision avec les murs et d'effectuer une rotation vers la droite systématiquement.
recupereUnPalai()
run() : permet en fonction des états de générer des comportements puis en fonctions des conséquences de ces comportements de basculer vers l'état suivant.
rechercheGraal()
quiMarque(): permet de déposer un dans la zone d'embut en s'appuyant sur les méthodes définies dans shrekCouleurs et shrekMoteurs.
getD() : permet d'accéder à la direction indiquée par la boussole.
retour0() : permet d'opérer une rotation vers ce qui pour la boussole correspond à la direction 0 en s'appuyant sur getD() et les méthodes définies dans la classe shrekQuiBouge.
getEtat() : permet d'accéder à l'état courant.
testLigne(): permet d'avancer et de s'arrêter une fois la ligne d'en but atteinte en s'appuyant sur les méthodes définies dans shrekCouleurs et shrekQuiBouge.
testPinces() : permet d'avancer et de s'arrêter une fois le capteur de toucher activé en s'appuyant sur les méthodes définies dans shrekTouch et dans shrekQuiBouge.
testPremierPlaet(): permet de mettre en œuvre la stratégie pour se diriger vers le premier palet, l'attraper et le déposer dans la zone d'en but adverse.