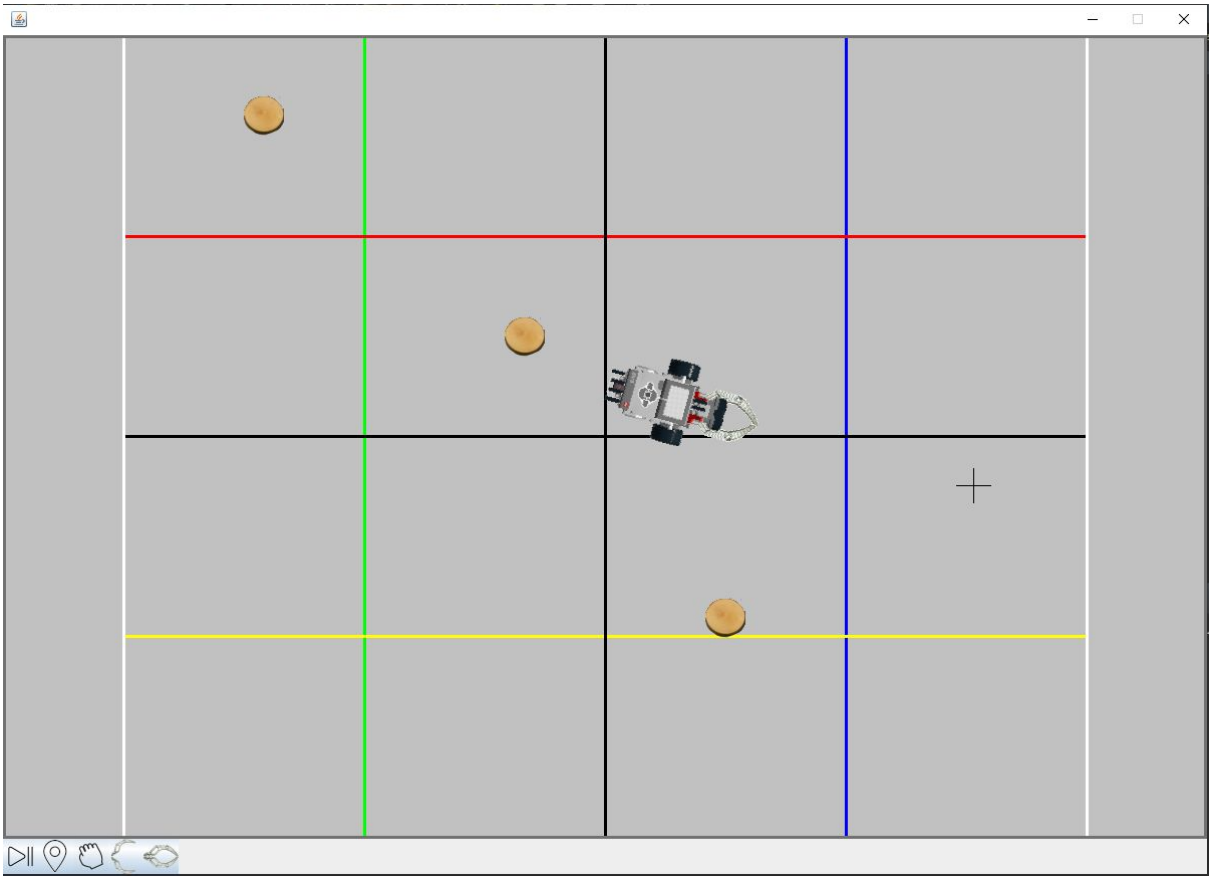
PLAN DE DÉVELOPPEMENT

Projet de modélisation graphique d'un robot LEGO ramasseur de palets



Aperçu de la fenêtre de l'appli, à ce jour.

FLEURY Pierre 11716306
GATTACIECCA Bastien 11808782
L3 MIASHS

10/12/2020

PREAMBULE

Ce plan de développement n'a pour but que de présenter l'organisation des différents paquetages, classes et méthodes ainsi qu'une documentation succincte. Il n'a pas pour but d'être exhaustif sur l'ensemble des éléments mais simplement de documenter les principales fonctions. Ainsi ce document ne sera pas mis à jour au fur et à mesure des modifications, contrairement au diagramme (UML) de classes.

Le plan de développement est utilisé si l'utilité d'une classe ou d'une méthode vous semble peu claire. De manière générale il permet de mieux comprendre la relation entre les classes du diagramme UML qui, lui, ne comporte aucune documentation (mais des précisions sur l'accessibilité, la visibilité, les relations d'héritage, d'implémentation, etc) d'où leur complémentarité.

Un diagramme UML est **fonctionnel**, d'où la nécessité de le mettre à jour à chaque itération dans le cadre de notre gestion de projet *agile*, tandis que la documentation formelle du plan de développement ne s'y prête pas vraiment.

SOMMAIRE

0. Hiérarchie

Paquetage	Classes	Documentation
agent	Objects	Classe qui décrit les attributs de base d'un Objet. Un Objet est un élément qui se situe dans l'environnement, comme un Palet ou un Robot. Ces deux types possèdent des caractéristiques en commun qui sont factorisées dans cette classe abstraite. On va trouver la position (x,y) sur le plateau ainsi que l'angle de l'objet. On définit deux méthodes abstraites update et paint(Graphics g) pour le polymorphisme. Il fallait trouver un nom pour cette classe suffisamment abstrait pour que cela corresponde à un Robot ET à un Palet. A ne pas confondre avec la classe Object de java.lang !!
	Palet	Classe qui décrit les caractéristiques d'un Palet. Les palets sont les objets de l'environnement qui doivent être capturés par les pinces des robots pour être marqués. Ils sont ronds et possèdent tous la même taille.
	Robot	Classe qui décrit les caractéristiques d'un Robot. Un Robot est un Objects dans l'environnement. Un Robot possède un ensemble de UpdatableDevice qui correspondent à ses composants. Ils sont de deux types : les Motor qui lui servent à interagir dans l'environnement et les Sensor qui lui permettent de récupérer de l'information de cet environnement.
devices	Motor	Classe qui décrit les fonctionnalités du moteur moyen. Ce moteur est utilisé pour ouvrir ou fermer les pinces. L'axe du moteur fait tourner une vis sans fin qui fait tourner deux roues dentées latérales (a pour effet l'ouverture symétrique des deux pinces). La différence de ratio fait que les pinces s'ouvrent assez lentement. Le moteur a pour caractéristiques d'avoir très peu d'inertie (il atteint la vitesse max quasi immédiatement) et tourne assez vite. On opte alors pour des constantes qui indiquent le sens du mouvement du moteur uniquement.
	BigMotor	Classe qui décrit les fonctionnalités du gros moteur. Ce moteur est utilisé pour déplacer une roue du robot. Un robot en possède 2, et un stabilisateur fixe à l'arrière. Le moteur a pour caractéristiques d'avoir de l'inertie (il prend un certain temps à atteindre la vitesse souhaitée) et de tourner moins vite qu'un Motor simple. On opte alors, en plus des variables décrites dans Motor, pour deux variables Double qui indiquent la force du moteur actuelle comprise entre deux constantes et celle que le moteur doit atteindre.
	Sensor	Classe abstraite mère de tous les capteurs. Les capteurs permettent au robot de récupérer de l'information de leur environnement. Notre robot dispose de trois capteurs qui permettent d'évaluer la distance (capteur à ultra sons SonicSensor), de reconnaître une couleur au sol (capteur de couleur ColorSensor), et de savoir si un objet se trouve dans le balancier (capteur tactile TouchSensor). Précisons que vu de dessus, sur le plan (x,y), les trois capteurs se trouvent aux mêmes coordonnées.
	ColorSensor	Classe qui décrit les caractéristiques du capteur de couleur. Le capteur se situe face au sol pour récupérer une couleur (uniquement).
	SonicSensor	Classe qui décrit les caractéristiques du capteur à ultrasons. Le capteur peut calculer une distance en comptant le temps qu'a mis l'onde sonore pour partir et revenir, sachant la vitesse de l'onde. Dans notre cas on fait partir un point de la position du capteur dans la direction du robot jusqu'à ce qu'il rencontre un Objects ou alors un des bords du plateau. On rappelle que le capteur a une portée de 5-255cm et qu'on travaille avec une échelle de (1cm = 4px).
	TouchSensor	Classe qui décrit les caractéristiques du capteur tactile. Concrètement, un

		balancier se situe dans les pinces du robot, lorsqu'un objet appuie sur celui-ci, le bras du balancier appuie sur le capteur tactile. On a choisi de ne pas coder les mouvements du balancier et de réduire l'information à "si les bornes d'un objet rencontrent la barre du balancier, alors celui-ci est déclenché". Les objets qui sont présents sur le plateau et qui sont susceptibles de trigger le capteur sont des sous-classes de Objects : on distingue les Palets et les Robots.
	UpdatableDevice	Interface qui permet de décrire un composant comme étant un composant du Robot et d'être mis à jour à chaque update générale.
environnement	FenetrePrincipale	Classe qui décrit la fenetre en tant que JFrame et ses composants. On ajoute à cette JFrame un PlateauGraphique, sous-type de Plateau et qui est lui sous-type de JPanel. La fenêtre doit ne pas être resizable, on lui enlève donc ses "décorations" (la barre d'attribut d'une fenêtre classique Windows) et son Layout.
	Plateau	Classe abstraite qui définit les caractéristiques d'un Plateau. Cette classe présente des constantes utiles comme les dimensions du plateau, ainsi qu'une BufferedImage pour avoir accès aux pixels de l'image ! On va pouvoir définir des méthodes de classe pratiques, comme contains(Point p). Une telle description d'un Plateau ne suffit pas pour pouvoir être instancié tel quel. Ainsi, on utilise un PlateauGraphique, sous-classe de Plateau pour pouvoir créer une instance "jouable" qui va pouvoir se rafraîchir pour mettre à jour tous les objets qui la composent.
	PlateauGraphique	Classe qui décrit une instance "jouable" de Plateau. Ainsi, un PlateauGraphique est capable de se rafraîchir pour mettre à jour (appel itératif des méthodes update() et paint()) tous les objets qui la composent. C'est dans un PlateauGraphique que l'on va instancier les différents Objects présents sur le plateau.

1 Agent

1.1 Objects

Attributs	x, y, angle : double	Les coordonnées et l'angle de l'Objects.
Méthodes	update()	Met à jour les caractéristiques de l'Objet à chaque itération.
	paint(Graphics)	Met à jour les caractéristiques graphiques de l'Objet à chaque itération.
	setX(double)	Redéfinit les coordonnées x.
	setY(double)	Redéfinit les coordonnées y.
	setAngle(double)	Redéfinit l'angle.
	getX() : double	Récupère les coordonnées x.
	getY() : double	Récupère les coordonnées y.
	getAngle() : double	Récupère l'angle.

1.2 Palet

Attributs	img : Image	L'image d'un palet.
Méthodes	getBounds() : Polygon	Pour récupérer les "bounds" (les bornes) du palet, on crée un objet Polygon. Le polygone est un octogone régulier, car il semble que 8 côtés sont assez pour évoquer les bords arrondis d'un palet.

1.3 Robot

Attributs	img : Image	L'image d'un robot.
	composants : UpdatableDevice[]	Liste des composants du robot. Moteurs & Senseurs.
	positionCapteurs: Point2D.Double	Position des capteurs, rappelons que tous les senseurs se trouvent, sur le plan (x,y), au même point ! (vu d'en haut
	positionPinces: Point2D.Double	Position du point de rotation des pinces.
	pp : PairePince	Une paire de pinces pour notre joyeux robot :) Les pinces ne sont pas un composant du robot. Elles sont dirigées par le petit moteur Motor

Méthodes	toString() : String	Représentation textuelle de l'état du robot.
	getPositionCapteurs(): Point2D.Double	Retourne la position des capteurs.
	getPositionPinces(): Point2D.Double	Retourne la position des pinces.
	getComposant(byte): UpdatableDevice	Récupère le composant à l'indice en paramètre. Cet indice est une constante vu que les composants du robot sont non modifiables (et on instancie les composants du robot dans un ordre précis)

2. Devices

2.1 Motor

Attributs	POSITIVE_TURN: byte	Constante mouvement moteur sens horaire.
	NEGATIVE_TURN: byte	Constante mouvement moteur sens anti-horaire.
	NULL_TURN: byte	Constante sans mouvement moteur.
	currentState: byte	Indique l'état courant du moteur {POSITIVE_TURN, NEGATIVE_TURN, NULL_TURN}.
Méthodes	updateDevice()	
	getState(): byte	L'état courant du moteur.
	setState(byte)	Accesseur à currentState pour redéfinir l'état du moteur. Si currentState n'est pas une des constantes d'état du moteur, lève une IllegalArgumentException

2.2 BigMotor

Attributs	FINAL FORCE_MOTEUR_MIN: double	Constante de la force moteur minimum.
	FINAL FORCE_MOTEUR_MAX: double	Constante de la force moteur maximum.
	currentForceMoteur: double	La force moteur actuelle, comprise entre FORCE_MOTEUR_MIN et FORCE_MOTEUR_MAX inclus.
	reachForceMoteur: double	La force moteur à atteindre.
Méthodes	updateDevice()	Pour le moment l'incrémentation/décrémentation de la puissance moteur est linéaire (+-0.01UA). Si la force moteur actuelle est inférieure à la force moteur à atteindre, on incrémente ; sinon on décrémente. On ajoute un palier de [-0.005;0.005] pour avoir une zone d'arrêt sûre. Si la force

		moteur actuelle est égale à la force moteur à atteindre à +/- 0.005, alors on dit que currentForceMoteur = reachForceMoteur.
	getForce(): double	La force moteur actuelle.
	setForce(double)	La force moteur à atteindre.

2.3 Sensor

Attributs	environnement: PlateauGraphique	Un capteur a accès à son environnement.
	robot: Robot	Un capteur donne les informations au robot sur lequel il est installé.
Méthodes	updateDevice()	
	getEnvironnement(): PlateauGraphique	Récupère l'environnement alentour.
	getRobot(): Robot	Récupère le robot.

2.4 ColorSensor

Attributs	BLACK: String	Constante String pour la couleur Noir.
	GREEN: String	Constante String pour la couleur Vert.
	RED: String	Constante String pour la couleur Rouge.
	BLUE: String	Constante String pour la couleur Bleu.
	YELLOW: String	Constante String pour la couleur Jaune.
	WHITE: String	Constante String pour la couleur Blanc.
	GREY: String	Constante String pour la couleur Gris.
	DARKGREY: String	Constante String pour la couleur Gris foncé qui indique les murs en plexiglas.
	pixels: Color[][]	Tableau de couleur récupéré depuis l'environnement.
	currentColor: Color	La couleur (r,g,b) actuellement sous le capteur de couleur.
Méthodes	updateDevice()	
	getColor(): String	Retourne sous forme de chaîne de caractères la couleur perçue par le capteur.

2.5 SonicSensor

Attributs	DISTANCE_MAX: int	Constante de la distance maximum perçue.
	DISTANCE_MIN: int	Constante de la distance minimale perçue.
	distance: int	Distance entre le capteur et la position la plus proche d'un Objects ou d'un bord du plateau
Méthodes	updateDevice()	Calcul de la distance.
	getDistance(): int	Retourne la distance perçue. distance ne peut pas être supérieur à DISTANCE_MAX, le cas est déjà géré dans la boucle.

2.6 TouchSensor

Attributs	TOUCH_RELEASED: boolean	Constante du capteur relâché.
	TOUCH_PRESSED: boolean	Constante du capteur enfoncé.
	state: boolean	Etat du capteur, prend la valeur des deux constantes TOUCH_RELEASED ou TOUCH_PRESSED.
Méthodes	updateDevice()	Vérifie si un Objects se trouve dans le balancier.
	getState(): boolean	Retourne l'état du capteur.

2.7 UpdatableDevice

Attributs	INDEX_BIGMOTOR_LEFT: byte	Attribue un indice au moteur gauche.
	INDEX_BIGMOTOR_RIGHT: byte	Attribue un indice au moteur droite.
	INDEX_MOTOR: byte	Attribue un indice au moteur.
	INDEX_TOUCH: byte	Attribue un indice au capteur de touché.
	INDEX_SONIC: byte	Attribue un indice au capteur de distance.
	INDEX_COLOR: byte	Attribue un indice au capteur de couleurs.
Méthodes	updateDevice()	Méthode abstraite.

3. Environnement

3.1 FenetrePrincipale

Attributs	panel: PlateauGraphique	Le panel sur lequel on dessine dans notre JFrame
Méthodes	getPanel(): PlateauGraphique	Récupère le panel

3.2 Plateau

Attributs	X: int	Dimension x du plateau de jeu.
	Y: int	Dimension y du plateau de jeu.
	curseur: Point	Coordonnées de la souris sur le plateau. Mise à jour que lorsque le curseur est demandé.
	plateau: BufferedImage	Image du plateau dont les pixels sont accessibles en lecture/écriture.
	pixels: Color[][]	Tableau 2D qui contient la couleur de tous les pixels.
Méthodes	contains(double,double): boolean	Méthode de classe qui indique si un point aux coordonnées (x,y) se situe dans le plateau.
	getPixels(): Color[][]	Retourne le tableau de Color de 2 dimensions. Cela permettra au ColorSensor de déterminer la couleur sur le plateau qu'il regarde.

3.3 PlateauGraphique

Attributs	curseurIcon: Image	L'image du curseur lorsqu'on est en mode pointeur pour diriger un ControlledRobot.
	objects: Objects[]	Tous les Objects présents sur le plateau.
	palets: Palet[]	Sous-ensemble de objects, tous les Palets présents sur le plateau.
	robots: Robot[]	Sous-ensemble de objects, tous les Robots présents sur le plateau.
	timer: Timer[]	Timer pour boucler.

Méthodes	update()	Update à chaque appel de actionPerformed
	paint(Graphics)	Repaint à chaque appel de actionPerformed
	getPalets(): Palet	Retourne le tableau de Palet.
	getRobots(): Robot	Retourne le tableau de Robot.
	actionPerformed(ActionEvent)	Permet à update() et paint(Graphics) de boucler ; toutes les 20ms