Résumé des commandes du robot

Grille des points d'arrêt du $\mathcal{R}^o\!\!\smile^o t$:	
`dg`	dessine la grille des points d'arrêt du $\mathcal{R}^o\!\!\smile^o t$ (peut être interrompue par Esc, § ou #).
`lg`(n)	définit la nouvelle longueur n des côtés d'un carreau de la grille ((12.0×12.0) pixels par défaut, $n \geq 0.0$).
`lgX`(n)	définit la nouvelle longueur n du côté horizontal (X) d'un carreau de la grille (12.0 pixels par défaut, $n \geq 0.0$).
`lgY`(n)	définit la nouvelle longueur n du côté vertical (Y) d'un carreau de la grille (12.0 pixels par défaut, $n \geq 0.0$).
`lgRT`(r,t)	définit les nouvelles longueurs des côtés d'un carreau de la grille par les coordonnées polaires r et t (la longueur $r \geq 0.0$ de la diagonale du carreau est exprimée en pixels et l'angle t que fait cette diagonale avec le côté horizontal est exprimé en degrés ($0.0 \leq t \leq 90.0$)).
`valLgX`	est égal à la valeur (de type `real`) de la longueur du côté horizontal (X) d'un carreau de la grille (comptée en pixels).
`valLgY`	est égal à la valeur (de type `real`) de la longueur du côté vertical (Y) d'un carreau de la grille (comptée en pixels).
`valLgR`	est égal à la valeur (de type `real`) de la longueur de la diagonale (R) d'un carreau de la grille (comptée en pixels).
`valLgT`	est égal à la valeur (de type `real`) de l'angle (T) de la diagonale d'un carreau de la grille avec son côté horizontal (comptée en degrés).

`valX`	est égal à la valeur (de type `real`) de l'abscisse absolue (X) du $\mathcal{R}\!\mathit{obot}$ (comptée en pixels).
`valY`	est égal à la valeur (de type `real`) de l'ordonnée absolue (Y) du $\mathcal{R}\!\mathit{obot}$ (comptée en pixels), de sorte que `X := valX; Y := valY`; et plus tard `lg(1.0); deplacer(X, Y)`; replace le $\mathcal{R}\!\mathit{obot}$ à sa place initiale sur l'écran).

Pivotements relatifs sur place :	
`phtd`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place d'un huitième de tour à droite.
`pqtd`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place d'un quart de tour à droite.
`p3htd`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place de 3 huitièmes de tour à droite.
`phtg`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place d'un huitième de tour à gauche.
`pqtg`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place d'un quart de tour à gauche.
`p3htg`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place de 3 huitièmes de tour à gauche.
`pdt`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place d'un demi-tour.

Pivotements absolus sur place :	
`pn`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place vers le nord (N).
`pne`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place vers le nord-est (NE).
`pe`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place vers l'est (E).
`pse`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place vers le sud-est (SE).
`ps`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place vers le sud (S).
`psw`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place vers le sud-ouest (SW).
`pw`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place vers l'ouest (W).
`pnw`	fait pivoter le $\mathcal{R}\!\mathit{obot}$ sur place vers le nord-ouest (NW).
`pivoter`(o)	fait pivoter le $\mathcal{R}\!\mathit{obot}$ dans l'orientation indiquée par la valeur de l'entier o modulo 8 égale à `nord` $= 0$, `nordEst` $= 1$, `est` $= 2$, `sudEst` $= 3$, `sud` $= 4$, `sudOuest` $= 5$, `ouest` $= 6$ ou `nordOuest` $= 7$.
`rt`(t)	change le repère et l'orientation du $\mathcal{R}\!\mathit{obot}$ pour qu'il effectue une rotation sur place d'un angle de t degrés (dans le sens trigonométrique si $t > 0$ et dans le sens des aiguilles d'une montre si $t < 0$).
`valOrientation`	est égal à la valeur (de type `integer`) de l'orientation du $\mathcal{R}\!\mathit{obot}$ codée par `nord` $= 0$, `nordEst` $= 1$, `est` $= 2$, `sudEst` $= 3$, `sud` $= 4$, `sudOuest` $= 5$, `ouest` $= 6$ et `nordOuest` $= 7$. Les directions `nord`, `est`, `sud` et `ouest` sont parallèles aux bords de l'écran, le `nord` en haut. Les directions `nordEst`, `sudEst`, `sudOuest` et `nordOuest` correspondent à l'orientation des diagonales d'un carreau de la grille.

Crayon :	
`bc`	baisse le crayon pour dessiner lors des déplacements.
`lc`	lève le crayon pour éviter de dessiner lors des déplacements.
`dc`	dessine une petite croix (de l'épaisseur du crayon), sans déplacer le $\mathcal{R}^{o}_{b}{}^{o}t$.
`dp`	dessine un point (de l'épaisseur du crayon), sans déplacer le $\mathcal{R}^{o}_{b}{}^{o}t$.
`ec`(n)	définit la nouvelle épaisseur n du crayon (1 pixel par défaut).
`valEc`	est égal à la valeur (de type `integer`) de l'épaisseur du crayon.
`valBc`	est égal à la valeur booléenne `true` (vrai) si et seulement si le crayon est baissé, sinon à `false` (faux).

Déplacements (avec ou sans tracé selon que le crayon est levé ou baissé) :	
`av`	fait avancer le $\mathcal{R}^{o}_{b}{}^{o}t$ tout droit jusqu'au prochain point d'arrêt sur la grille.
`vd`	fait tourner le $\mathcal{R}^{o}_{b}{}^{o}t$ par un virage d'un quart de cercle à droite.
`vg`	fait tourner le $\mathcal{R}^{o}_{b}{}^{o}t$ par un virage d'un quart de cercle à gauche.
`avf`(n)	fait avancer le $\mathcal{R}^{o}_{b}{}^{o}t$ tout droit n fois (sans rien faire si $n = 0$ et en reculant si $n < 0$).
`tl`(l)	change le repère (mais pas l'angle des directions NE-E) et fait avancer le $\mathcal{R}^{o}_{b}{}^{o}t$ par une translation de l pixels dans la direction où il se trouve (si $l > 0$) ou dans la direction opposée (si $l < 0$).
`tracer`(X,Y)	trace un segment de droite (si le crayon est baissé) de la position courante du $\mathcal{R}^{o}_{b}{}^{o}t$ au point de coordonnées (X,Y) et déplace le $\mathcal{R}^{o}_{b}{}^{o}t$ en ce point.
`valLgTl`	est égal à la valeur (de type `real`) de la longueur du segment tracé par la dernière translation par `av` ou `tl` (comptée en pixels).

Positionnement au centre et sur les bords du cadre :	
`ce`	place le $\mathcal{R}^{o}_{b}{}^{o}t$ et *le centre de la grille* au centre de *l'écran*, sans changer l'orientation du $\mathcal{R}^{o}_{b}{}^{o}t$ et sans rien dessiner.
`eh`	place le $\mathcal{R}^{o}_{b}{}^{o}t$ sur la grille en haut, sans changer sa position verticale, ni son orientation et sans rien dessiner.
`eb`	place le $\mathcal{R}^{o}_{b}{}^{o}t$ sur la grille en bas, sans changer sa position verticale, ni son orientation et sans rien dessiner.
`ag`	place le $\mathcal{R}^{o}_{b}{}^{o}t$ sur la grille à gauche, sans changer sa position horizontale, ni son orientation et sans rien dessiner.
`ad`	place le $\mathcal{R}^{o}_{b}{}^{o}t$ sur la grille à droite, sans changer sa position horizontale, ni son orientation et sans rien dessiner.

Repère cartésien :
`translater`(X,Y) translate le *Robot* de X carreaux de la grille horizontalement (à droite si $X > 0$ et à gauche si $X < 0$) et Y carreaux de la grille verticalement (en haut si $Y > 0$ et en bas si $Y < 0$), sans changer son orientation et sans rien dessiner.
`deplacer`(X,Y) déplace le *Robot* au point de coordonnées (X, Y) sur la grille, sans changer son orientation ni rien dessiner, l'origine du repère cartésien étant placée au centre de la grille.

Test booléen de l'orientation du *Robot* :	
`tn`	teste si le *Robot* est orienté au nord (N).
`tne`	teste si le *Robot* est orienté au nord-est (NE).
`te`	teste si le *Robot* est orienté à l'est (E).
`tse`	teste si le *Robot* est orienté au sud-est (SE).
`ts`	teste si le *Robot* est orienté au sud (S).
`tsw`	teste si le *Robot* est orienté au sud-ouest (SW).
`tw`	teste si le *Robot* est orienté à l'ouest (W).
`tnw`	teste si le *Robot* est orienté au nord-ouest (NW).

Test booléen de la position du *Robot* :	
`tb`	teste si le *Robot* est au bord du cadre (un déplacement du *Robot* en avant par `av` le faisant sortir du cadre).

Temps :
`heure`(`H`, `Mi`, `S`) affecte aux variables `H`, `Mi` et `S` trois entiers qui sont l'heure (entre 0 et 23), la minute (entre 0 et 59) et la seconde (entre 0 et 59) présentes.
`date`(`J`, `Mo`, `A`) affecte aux variables `J`, `Mo` et `A` trois entiers qui sont le jour (entre 1 et 31), le mois (entre 1 et 12) et l'année courante.
`delai`(t) attendre pendant un délai de t secondes (t de type `real`).

Interruption de l'exécution du programme :
[Esc] ou [§] ou [#] stoppe le *Robot* au cours de l'exécution de l'une de ses commandes.

Mise au point du programme :	
`st`	stoppe le $\mathcal{R}^{o}_{b}{}^{o}t$ qui s'arrête définitivement en tapant l'une des touches Esc, §, ou #) ou redémarre en enfonçant une autre touche du clavier ou le bouton de la souris. Avant de redémarrer, l'utilisateur a la possibilité de diminuer (avec la touche -) ou d'augmenter (avec la touche +) la vitesse du $\mathcal{R}^{o}_{b}{}^{o}t$ (entre 0 et 10, la vitesse 0 faisant passer en mode pas à pas).
`pp`	exécute la suite du programme en mode pas à pas, à la vitesse 0.
`ex`	exécute la suite du programme en mode exécution normale, à la vitesse 10.
`vt`(v)	exécute la suite du programme à la vitesse v (en pas à pas si $v \leq 0$, avec un maximum de 10 si $v > 10$).
`montrerRobot`	rend le $\mathcal{R}^{o}_{b}{}^{o}t$ visible sur l'écran pendant le dessin (option par défaut).
`cacherRobot`	rend le $\mathcal{R}^{o}_{b}{}^{o}t$ invisible sur l'écran pendant le dessin.
`InterdireSorties`	interdit les sorties du $\mathcal{R}^{o}_{b}{}^{o}t$ en dehors du cadre dans la suite du programme (option par défaut).
`autoriserSorties`	autorise les sorties du $\mathcal{R}^{o}_{b}{}^{o}t$ en dehors du cadre dans la suite de l'exécution du programme.
`valVt`	est égal à la valeur (de type `integer`) de la vitesse du $\mathcal{R}^{o}_{b}{}^{o}t$ (entre 0 (mode pas à pas) et 10).
`valVisible`	est égal à la valeur booléenne `true`, (respectivement `false`) si le $\mathcal{R}^{o}_{b}{}^{o}t$ est visible (respectivement invisible) sur l'écran.
`valAutoriserSorties`	est égal à la valeur booléenne `true` (respectivement `false`) si les sorties en dehors du cadre sont autorisées (respectivement interdites).

Couleur :
Les couleurs sont `blanc` $= 0$, `jaune` $= 1$, `bleu` $= 2$, `violet` $= 3$, `vert` $= 4$, `rouge` $= 5$, `indigo` $= 6$, `noir` $= 7$ (dans l'ordre du plus clair au plus foncé sur un écran ayant au moins 8 niveaux de gris) et `inverse = 8`. Sur un écran noir et blanc, et selon le modèle de l'ordinateur, toutes les couleurs (sauf blanc) apparaissent en noir ou les couleurs sont représentées par des motifs répétés.

`cc`(c)	définit la couleur c du crayon. Si la couleur est `inverse`, les tracés se feront en vidéo inverse (par exemple noir sur fond blanc et blanc sur fond noir).

`peindre` peint de la couleur du crayon la région blanche de surface maximale où se trouve le $\mathcal{R}^{o}_{b}{}^{o}t$ (la peinture peut être interrompue par [Esc], [§] ou [#], le $\mathcal{R}^{o}_{b}{}^{o}t$ continuant ensuite son travail normalement).

`cf`(c) efface complètement le dessin puis remplit l'écran de la couleur de fond c, `inverse` n'étant pas autorisé. Le $\mathcal{R}^{o}_{b}{}^{o}t$ est placé au centre de l'écran dans les conditions initiales (grille (12×12), visible, tourné vers le nord, sorties hors du cadre interdites, crayon noir de taille 1 baissé et exécution normale à la vitesse 10).

`valCf` est égal à la valeur (de type `integer`) de la couleur du fond.

`valCc` est égal à la valeur (de type `integer`) de la couleur du crayon.

`ecranNoirEtBlanc` est égal à `true` si et seulement si l'écran est noir et blanc (sinon `false`).

Écriture dans la fenêtre de dessin du $\mathcal{R}^{o}_{b}{}^{o}t$:

`ecrire`(c) écrit la chaîne de caractères c (entre apostrophes ′ et ′, où toute apostrophe ′ doit être doublée) à droite du $\mathcal{R}^{o}_{b}{}^{o}t$ sans changer sa position ni celle de son crayon.

`entierEnChaine`(i) est la chaîne de caractères représentant l'entier i en base 10.

`reelEnChaine`(r) est la chaîne de caractères représentant le réel r sous la forme $d.dddd$`E`$\pm ee$ où d est un digit de la mantisse, e est un digit de l'exposant et `E` se lit "fois 10 à la puissance".

`booleenEnChaine`(b) est la chaîne de caractères (′`Vrai`′ ou ′`Faux`′) représentant le booléen b.

`concat`$(c_1,c_2,\ldots,c_n)$ est la chaîne de caractères obtenue en concaténant les chaînes c_1, c_2, ..., c_n les unes derrière les autres.

`policeTexte`(p) choisit la police p du texte (qui dépend du type d'ordinateur utilisé).

`tailleTexte`(t) choisit la taille t du texte (qui dépend du type d'ordinateur utilisé).

`styleTexte`(s) choisit le style du texte (qui dépend du type d'ordinateur utilisé).

`valPoliceTexte` valeur de la police courante du texte.

`valTailleTexte` valeur de la taille courante du texte.

`valStyleTexte` valeur du style courant du texte.

Messages affichés en dessous de la fenêtre de déplacement du Robot :
`message`(c) efface le message dans le cadre prévu en dessous de la fenêtre de déplacement du Robot puis y écrit la chaîne de caractères c.
`marquerUnePause` marque une pause (en faisant clignoter le symbole ↩ dans la fenêtre des messages) en attendant que le bouton de la souris ou qu'une touche du clavier soit enfoncé.
`effacerMessage` efface le message dans le cadre prévu en dessous de la fenêtre de déplacement du Robot.
`lireCar`(`C`) attend (en faisant clignoter le symbole ↩ dans la fenêtre des messages) qu'une touche du clavier ou que le bouton de la souris soit enfoncé, puis lit le caractère correspondant à la touche enfoncée (ou le caractère nul `chr`(`0`) si le bouton de la souris a été enfoncé) et l'affecte à la variable `C` (de type caractère `char`).
`mementoCommandesClavier` retourne une chaîne de caractères rappelant les commandes de pilotage interactif du Robot.
`valMessage` est égal à valeur (de type `string`) du message (de type `string`) inscrit dans le cadre sous la fenêtre de déplacement du Robot ou la chaîne vide '' s'il n'y a aucun message.

Lecture et écriture dans la fenêtre texte de Pascal :
`modeTexte` fait apparaître la fenêtre "Texte" de Pascal (pour réaliser les entrées-sorties avec les instructions `write`, `writeln`, `read` et `readln`). La réapparition de la fenêtre de dessin du Robot est automatique à la prochaine utilisation d'une commande du Robot.

Impression et conservation du dessin du Robot :
`definirFormatImpression`(f) définit le format f = `reduction` = 0, `standard` = 1 ou `agrandissement` = 2 de la prochaine impression.
`definirTitreImpression`(t) définit la chaîne de caractères t (entre apostrophes ' et ', où toute apostrophe ' doit être doublée) comme le titre pour la prochaine impression.
`imprimer` imprime le dessin dans le format et avec le titre (en dessous du dessin) tels qu'ils ont été dernièrement définis (`standard` et '' par défaut).
`valFormatImpression` est égal à la valeur (de type `integer`) du format d'impression.
`valTitreImpression` est égal à la valeur (de type `string`) du titre d'impression.
`copier` copie le dessin sur disque (dépend de l'ordinateur utilisé).

Sons :

`bip` émet un son bref.
`biiip` émet un son long.
`son`(f,t) émet un son de fréquence f (comprise entre 15 et 15000) pendant t secondes (f et t de type `real`, peut être interrompu par Esc, § ou #).

Musique :

`metronome`(n) règle le métronome à n battements par seconde (`largo` = 50, `larghetto` = 63, `adagio` = 71, `andante` = 92, `moderato` = 114, `allegro` = 144, `presto` = 184, `prestissimo` = 204).

`nuance`(n) règle la nuance n (`pianissimo` = 1, `piano` = 2, `mezzo_piano` = 3, `un_poco_piano` = 4, `sotto_voce` = 5, `mezza_voce` = 6, `un_poco_forte` = 7, `mezzo_forte` = 8, `forte` = 9, `fortissimo` = 10).

`note`(h,d) joue une note de hauteur $h = (((o * octave) + n) + a)$ où $o = -3$, -2, ..., 3 est l'octave de la note (`octave` = 12), n = `d0` ou `ut` = 0, `re` = 2, `mi` = 4, `fa` = 5, `sol` = 7, `la` = 9, `si` = 11 est le nom de la note, a = `double_bemol` $= -2$, `bemol` $= -1$, `becarre` = 0, `diese` = 1, `double_diese` = 2 est l'altération de la note et d = `note_carree` = 8.0, `ronde` = 4.0, `ronde_pointee` = 6.0, `blanche` = 2.0, `blanche_pointee` = 3.0, `blanche_en_triolet` = 4/3, `noire` = 1.0, `noire_pointee` = 1.5, `noire_en_triolet` = 2/3, `croche` = 0.5, `croche_pointee` = 0.75, `croche_en_triolet` = 1/3, `double_croche` = 0.25, `double_croche_pointee` = 0.375, `double_croche_en_triolet` = 0.5/3, `triple_croche` = 0.125, `triple_croche_pointee` = 0.1875, `triple_croche_en_triolet` = 0.25/3, `quadruple_croche` = 0.0625, ou `quadruple_croche_en_triolet` = 0.125/3 est la durée de la note.

`silence`(d) marque un silence de durée d = `baton_de_deux_pauses` = 8.0, `pause` = 4.0, `pause_pointee` = 6.0, `demi_pause_pointee` = 3.0, `demi_pause` = 2.0, `soupir` = 1.0, `soupir_pointe` = 1.5, `demi_soupir` = 0.5, `demi_soupir_pointe` = 0.75, `quart_de_soupir` = 0.25, `quart_de_soupir_pointe` = 0.375, `huitieme_de_soupir` = 0.125, `huitieme_de_soupir_pointe` = 0.1875, `seizieme_de_soupir` = 0.0625.

Interaction avec l'utilisateur du $\mathcal{R}\!\underset{\smile}{o}\!\underset{\smile}{o}t$:

`interaction`(`X`, `Y`, `Ch`, `Coordonnees`, `Pivotements`, `Dessin`, `FinSouris`, `Sauver`) retourne les coordonnées (x,y) du $\mathcal{R}\!\underset{\smile}{o}\!\underset{\smile}{o}t$ déplacé par l'utilisateur avec la souris ou les flèches de défilement ↑, ↓, ← ou → au moment où l'interaction avec l'utilisateur se termine.

L'interaction se termine quand le bouton de la souris est enfoncé (auquel cas la variable `Ch` de type `char` prend pour valeur le caractère nul `chr(0)`) ou quand une touche est tapée (le caractère correspondant étant affecté à `Ch`). Les coordonnées (x,y) du $\mathcal{R}^{o}_{b}{}^{o}t$ sont relatives à sa position au moment de l'appel de la procédure et comptées sur la grille. Elles sont affectées aux variables `X` et `Y` de type `integer`. L'interaction peut être interrompue par les touches Esc, § ou #.

– Si la valeur booléenne `Coordonnees` est `true` (respectivement `false`) alors les coordonnées courantes du $\mathcal{R}^{o}_{b}{}^{o}t$ déplacé par la souris ou les flèches de défilement sont (ne sont pas) affichées à l'écran pendant l'interaction.

– Si la valeur booléenne `Pivotements` est `true` alors l'utilisateur a la possibilité de faire pivoter interactivement le $\mathcal{R}^{o}_{b}{}^{o}t$ (touches – (`phtg`), G (`pqtg`), + (`phtd`) et D (`pqtd`)) avant la fin de l'interaction. Dans ce cas l'interaction peut se terminer uniquement en enfonçant le bouton de la souris ou la touche ↩.

– Si les valeurs booléennes `Pivotements` et `Dessin` sont égales à `true` alors l'utilisateur a la possibilité de dessiner avant la fin de l'interaction (touches A (`av`), ((`vg`),) (`vd`), L (`lc`), B (`bc`), P (`dp`), C (`dc`), . (`dg`) et * (changement de la couleur du crayon) en corrigeant les erreurs avec ←). Dans ce cas l'interaction peut se terminer en enfonçant le bouton de la souris ou la touche ↩.

– Si la valeur booléenne `FinSouris` est `true` alors l'utilisateur a la possibilité de terminer en enfonçant le bouton de la souris sinon il doit terminer en enfonçant une touche du clavier.

– Si les valeurs booléennes `Dessin` et `Sauver` sont `true` alors un programme `Dessin_du_robot.p` dont l'exécution reproduit le dessin interactif est engendré sur disque. S'il y a une erreur (disquette vérrouillée,...) alors `dessinDuRobotEngendre` sera égal à `false` après l'exécution de `interaction`.