Projet d'ingénieurie simultanée

Babyfoot : Partie Stratégie

Pierre-Benoît Blanc, Gildas Jalon
5 juin 2015

Cistophe Salztmann

Table des matières

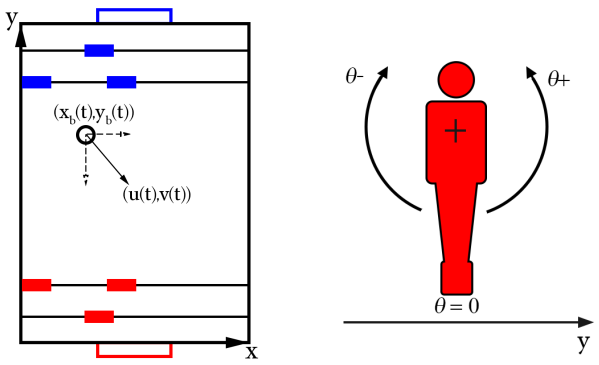
1 Introduction

Ce projet a pour but de réaliser un programme en LABVIEW capable de commander un babyfoot motorisé, le programme devra être capable de se confronter à un joueur humain. Il s'inscrit dans un grand projet comportant :

- Une partie "vision de la balle" devant déterminer la position de la balle et son vecteur vitesse.
- Une partie "position des adversaires" devant déterminer la position longitudinale et angulaire des rangées adverses.
- Une partie "stratégie" dont ce projet fait l'objet.
- Une partie "commandes" devant guider les moteurs linéaires et rotatifs sur la base de positions calculées dans la partie stratégie.

Certains choix de départs et hypothèses doivent être immédiatement mis au clair pour comprendre la suite du rapport :

- Le repère du terrain est placé tel que le zéro se situe dans la coin à gauche du gardient de l'ordinateur (Figure 1.(a)).
- Le choix est fait de considérer le babyfoot comme bidimensionnel sans considérer la position en hauteur du pied de chaque figurine. Chaque figurine est donc modéliser par un rectangle de taille (21,2x10mm) pouvant se déplacer selon l'axe x par translation et l'axe y par projection de la rotation. La hauteur d'une figurine est indirectement prise en compte seulement dans certaines fonctions du programme. Notamment dans le controle de la balle.
- On considère que la trajectoire de la balle est toujours rectiligne. Tous effets due à la rotation de la balle sur elle–même sont négligés.
- La stratégie ne traite pas de l'aspect mécanique dans la dynamique du mouvement des rangées.
- Seuls les goals et les lignes de défenses sont montés dans le cadre de ce projet.
- Les positions de la balle et des lignes sont supposées parfaitement précises.
- La forme du pied des figurines est supposée parfaitement rectangulaire de dimensions (21,2x10mm) sans inclinaison ni concavité des faces.
- Le feedback des commandes n'est pas géré par le programme.
- Toutes les distances sont en mm.



(a) Repère et positions initiales sur x

(b) Coordonée angulaire

FIGURE 1 – Repère

Le programme se structure en deux boucles *while* indépendantes (Figure 2). La première boucle calcule toutes les informations nécessaires au choix du comportement et aux actions de l'ordinateur. La seconde boucle "pioche" à l'aide de variables locales dans les informations pour choisir et éxecuter des actions. Le principe est de mettre à jour continuellement les informations afin de choisir ou d'achever une action efficacement. La structure globale est synthétisée dans une orgnigramme (Annexe 1).

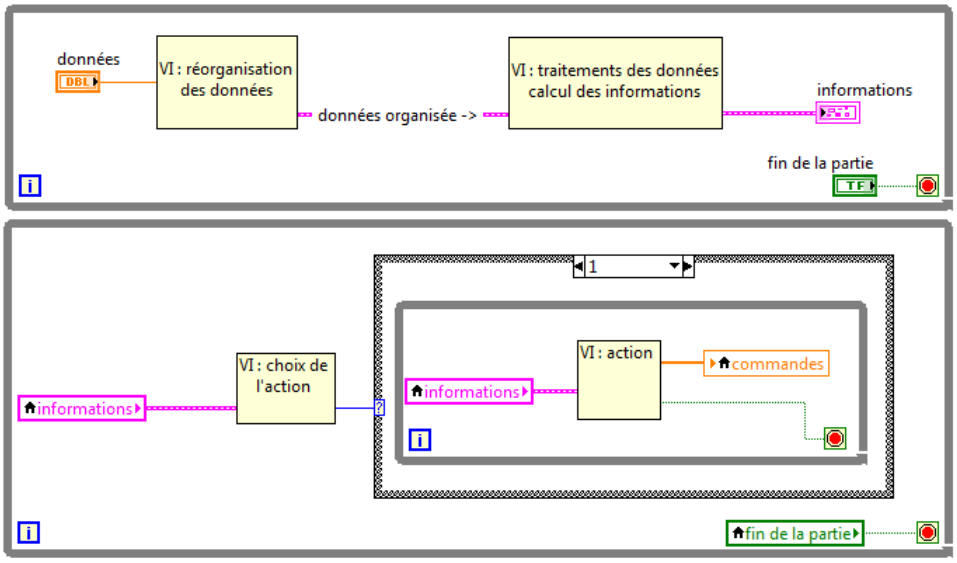


FIGURE 2 – Principe et structure globale du programme

2 Colletcte des données

Pour les deux camps (ordinateur et joueur) les positions reçues sont transformés selon les tableaux suivants :

x_g	θ_g
x_d	θ_d
x_m	θ_m
x_a	θ_a

x_g				
y_g			θ_g	
x_{d1}			x_{d2}	
y_d			θ_d	
x_{m1}	x_{m2}	x_{m3}	x_{m4}	x_{m5}
y_m			θ_m	
x_{a1}	x_{a2}	x_{a3}		
y_a			θ_a	

FIGURE 3 – Données avant et après la réorgatinasion

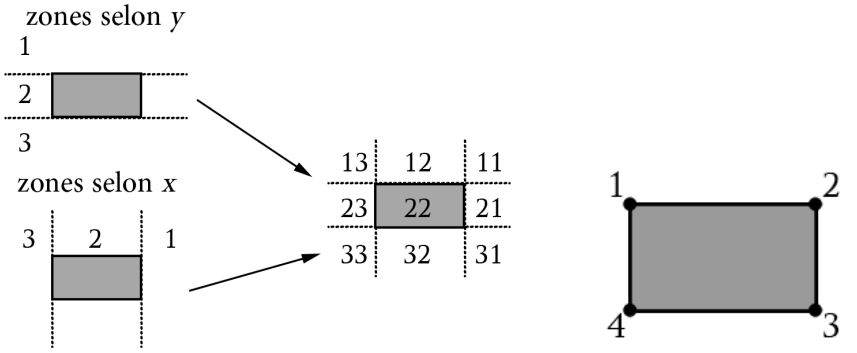
Le but de cette réorganisation des données brutes est de créer un tableau comportant la position de chaque figurine et non seulement celle d'une rangée. Le tableau est aussi organisé de façon à minimiser la quantité d'informations. En effet, les positions communes aux figurines ne sont mentionnées qu'une seule fois. La collecte est implémentée dans le VI *collecte_ données* (Annexe 2).

3 Informations

3.1 Zone de balle et points exposés

Le pied des figurines de babyfoot a 4 faces, toutes ne sont pas exposées à la balle. Il y a 9 zones dans la quelle la balle peut se trouver par rapport à une figurine (figure 4.(a)).

Connaitre ces zones est indispensable :



(a) Zones realtives à une figurine (b) numérotation de coins

FIGURE 4 – Modélisation d'une figurine

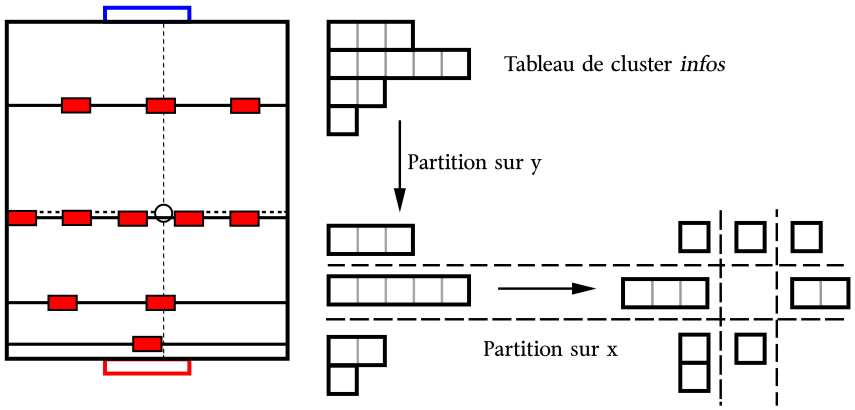


FIGURE 5 – Partitionnement des figurines selon leur position relative à la balle

- Pour connaître les faces exposées et savoir avec quelle face l'ordinateur peut frapper la balle.
- Pour connaître les deux points definissant le secteur, centré sur la balle, que couvre une figurine.
- Pour le calcul des faisceaux (savoir si une figurine en cache une autre).

Par souci de fonctionnalité les zones sont numérotés selon la Figure 4.(b).
Le progamme partitionne l'ensemble des figurines dans les 9 zones en deux étapes (Figure 5). Dans un premier temps il étudie la zone selon l'axe y et scinde le tableau de rangées en 1, 2 ou 3 tableaux en regroupant les rangées pour les quelles la balle est dans la zone 1, 2 ou 3. Puis, pour chaque rangée on parcours les figurines en déterminant leur zone selon l'axe x. On peut ensuite calculer les informations qui en découlent, c'est à dire les points définnissant les faces exposées.
Le VI *zone_ balle_ 1D.vi* est utiliser deux fois pour déterminer la zone selon y puis x, le reste du processus (partitionement) est implémenter dans les VIs *infos.vi* et *zone_ balle_ 2D.vi* (section ? ? ? ?).

3.2 Possession et controle

Les critères de possession et de contrôle sont deux informations indisepensables à la bonne adapatation du comportement de l'ordinateur. C'est grâce à eux qu'il est possible d'opter pour une attitude plutôt defensive ou attaquante. Ces deux critères s'inscrivent dans le processus de détermination du comportement de l'ordinateur. En effet le comportement est étroitement lié avec la proximité et la maitrise de la balle. Le choix a donc été de distinguer les deux informations afin de mieux adapter le comportement de l'ordinateur.

Possession : *Une figurine possède la balle si elle est en mesure de la contrôler.*
Ceci implique que soient pris en compte la position de la balle et sa vitesse. De façon générale quand la balle est à proximité d'une figurine, lcette dernière est plus en mesure de la contrôler. Mais si la balle passe à grande vitesse de la figurine celle-ci n'est plus en mesure de le faire. Chaque figurine possède donc une zone de possession en forme de rectangle dont la longueur des côtés sont inversement proportionnelles à la vitesse de la balle. Quand la balle se trouve dans une zone de posession de l'adversaire l'ordinateur opte pour une attitude déffensive cherchant à fermer les couloirs. Quand la balle se trouve dans une zone de possession de l'ordinateur, ce dernier cherche à controler la balle pour effectuer des actions d'attaque.

Contrôle : *Une figurine controle la balle quand elle est en contact avec elle.*
Le calcul se fait simplement sur la base du rayon de la balle et de la position des 4 points de la figurine. Ce critère permet de savoir qui est le dernier à avoir touché la balle et si la balle passe d'une figurine à une autre. Ces informations seront nécessaire à la bonne exécution de certaines actions. L'information concernant le contrôle est mémorisée dans plusieurs variables différentes. L'une se trouve dans le cluster d'information de la balle, elle garde la valeur indiquant qui contrôle pendant une certaine durée. Les autres variables se trouvent dans le cluster d'informations de l'ordinateur et du joueur et indique quelle figurine dans quelle rangée contrôle la balle et sont modifiées immédiatement.

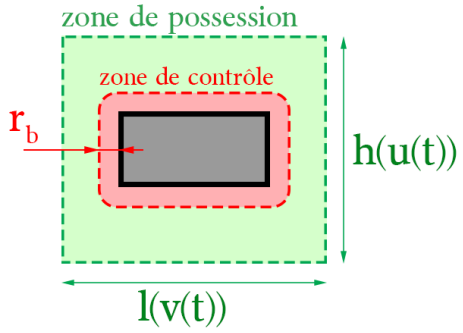


FIGURE 6 – Zones de possession et de contôle

La nécessité d'un critère de contrôle est indispensable pour adapter le comportement de l'ordinateur. Si le critère de controle est mal pensé, l'ordinateur risque d'éxecuter des mouvements de tire alors qu'il ne maîtrise pas la balle. La façon de calculer le critère de controle est donc discutable. En effet le programme recoit uniquement de données spatiales. Même ci ces données étaient exactes par rapports aux positions réelles, le contact avec la balle reste difficile à établir sur la seule base de données spatiales. Il est possible de définir un critère suplémentaire pour l'ordinateur uniquement qui permettrai de savoir si la balle est bloquée sous une figurine.

3.3 Distance à la trajectoire

Lorsque la balle parcours le terrain elle doit parfois être stoppée dans sa course. Par exemple quand la balle se dirige vers les buts de l'ordinateur, il faut impérativement la stopper. La distance à la trajectoire est donc directement utilisée dans le calcul des commandes en position des moteurs linéaires.

La distance à la trajectoire *est définie comme la distance nécessaire pour placer une figurine donnée sur la trajectoire de la balle afin de l'intercepter.*
Cette information est calculé pour chaque figurine de chaque rangée. De plus une information complémentaire est associée à chaque rangée : le numéro de case de la figurine la plus proche de la trajectoire.

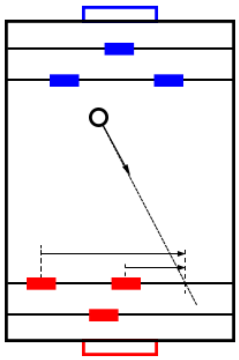


FIGURE 7 – Distance à la trajectoire.png

3.4 Directions de tir

Ici le but est de découvrir tout les chemins possibles que la balle peut effectuer pour se diriger vers les rangés dans un sens vers le goal du joueur et les lignes de l'ordinateur et dans l'autre, le goal de l'ordinateur et les lignes du joueur.
Pour trouver ces trajectoires il est impératif de trouver les angles de directions de chaque trajectoire. Du à l'utilisation d'un système de coordonnée cartésien, pour trouver un angle il faut utiliser les tangentes (x/y).

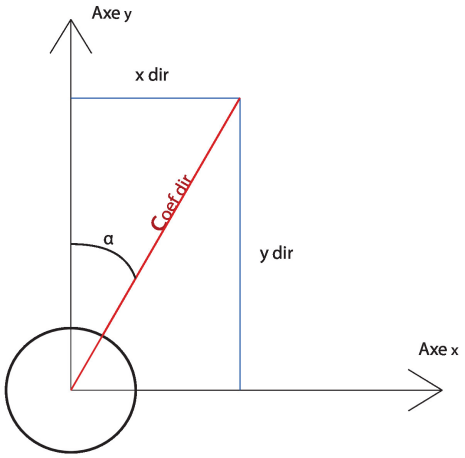
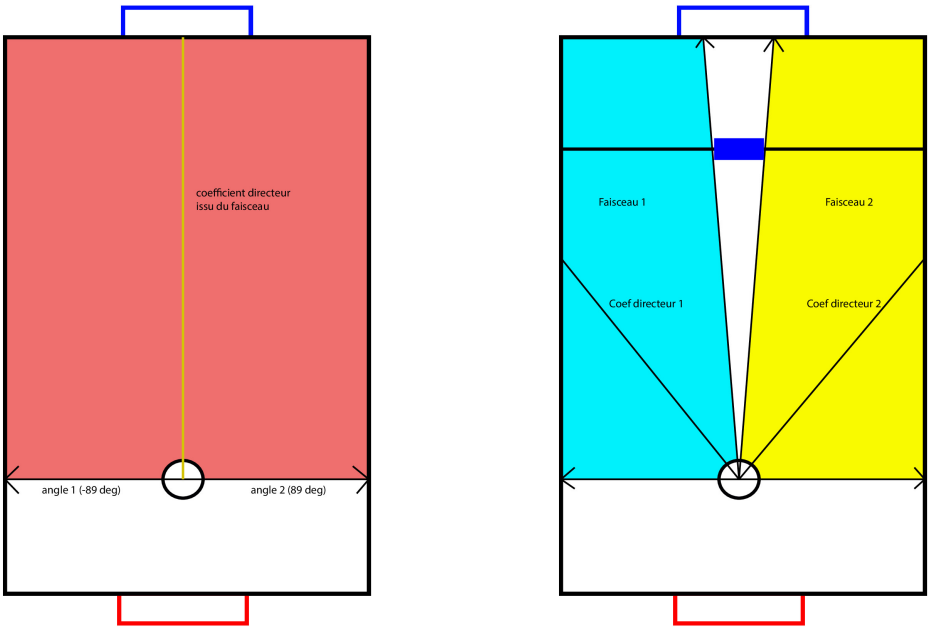


FIGURE 8 – Repère cartésien centré sur la balle

Pour ce faire, considérons que la balle est libre d'aller dans n'importe quelle direction contenu entre $-\pi/2$ et $\pi/2$. A partir de cela il va falloir trouver les endroits où la balle sera bloquée par des obstacles (figurines du joueur ou limite du terrain). Nous connaissons la positions des figurines adversee et grâce au VI des faces exposés, nous pouvons réduire ce segment impossible à franchir par deux points correspondant à ses deux extrémités .
En imaginant qu'il n'y ait sur le terrain qu'une seule figurine contrôlée par le joueur, la balle n'a que deux champs de libre que le programme va ensuite réduire en deux coefficients de direction

qui correspondent au milieu de chaque champs.



(a) Initialisation du faisceau

(b) Séparation du faisceau du à un obstacle

FIGURE 9 – Evolution des faisceaux

Cela est effectué pour chaque obstacle. Chaque possibilité de passe/interception pour chaque rangé de figurine contrôler par l'ordinateu, pour ensuite finir par les buts est sauvegarder.

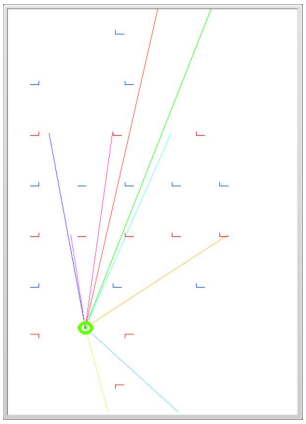


FIGURE 10 – représentaton des directions possible de la balle

Sur cette image, les joueurs en bleu sont contrôlés par l'ordinateur tandis que les rouges sont controlés par le joueur.

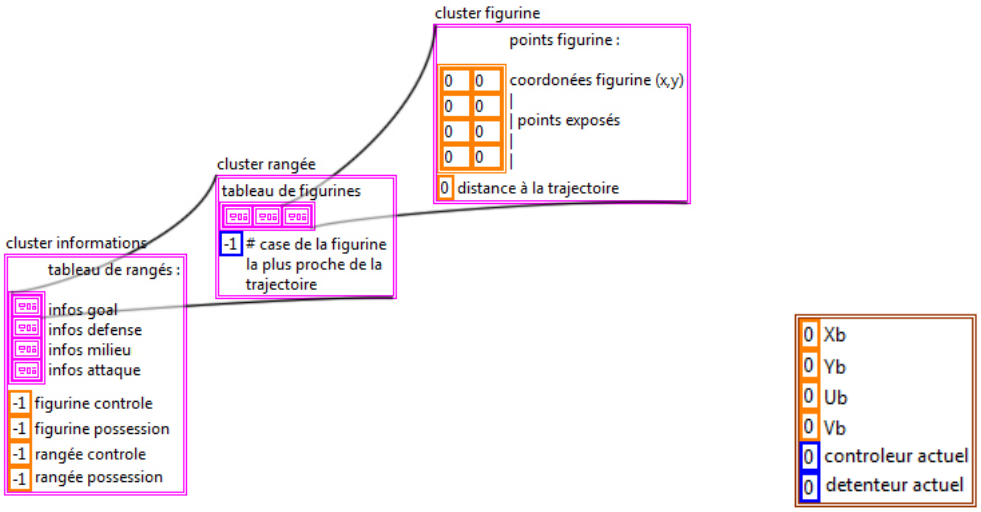
En regardant la complexité du VI *faisceau.vi*, on est en droit de se demander s'il est vraiment nécessaire et indispensable au fonctionnement général du programme. En effet il demande à lui seul une durée de 31 millisecondes et il s'exécute deux fois par boucle. Les résultats obtenus par ce VI ne seront en effet utilisés que dans certains cas. Il devient vraiment utile pour faire de la contre-attaque, c'est-à-dire que l'ordinateur ne contrôle pas encore la balle mais sait déjà quelles sont ses possibilités pour réagir et n'aura pas forcément besoin de la contrôler avant de tirer. Ou alors ce programme peut être utililser pour passer la barrière des milieux où généralement personne

ne contrôle vraiment la balle et le but est de la récupérer par la ligne des avants. On pourrait aussi imaginer, afin de déstabiliser l'adversaire, faire une sorte de passe à 10 sur tout le terrain. On pourrait aussi utiliser ce VI afin d'optimiser le placement des joueurs en fonction de la balle. Il s'agit là de stratégie plutôt avancé et qui n'a pas forcément grand intérêt sur un baby-foot à deux rangées mais dans un futur où les quatre lignes seront présentes, ce VI deviendra certainement nécessaire au bon fonctionnement du baby-foot.

Un des gros points qui peut être soulever est est-ce qu'on a besoin d'autant d'informations en quasi-continu pendant toute la durée du jeu ? Une des pistes à observer serait de déterminer à quel moment ce VI serait le plus utile et ne l'appeler qu'à ce moment là. Ce qui fait la lourdeur de ce programme est l'utilisation de shift register et de tableau dynamique. On pourrait par exemple limiter la recherche d'ouverture à deux maximum pour les buts et 3 pour les passes et interception lors des phases de possession de la balle par l'ordinateur. Ainsi en créant des tableaux fixes on pourrait diminuer le temps d'éxecution. Une autre piste serait de n'utiliser ce programme que pour un seule coté à la fois, quand l'ordinateur possède la balle, on regarde par où la balle peut revenir si elle se fait intercepter par l'adversaire afin de boucher rapidement les ouvertures. Et faire l'inverse lorsque le joueur la possède.

3.5 Organisation de l'information

L'organisation de l'information est rendue difficile par l'incapacité du langage LABVIEW à supporter les tableaux bidimensionels non rectangulaires alors que le nombre de figurines par ligne n'est jamais le même. Il a donc été décidé de communiquer l'information par des tableaux de cluster, orffrant le double avantage de pouvoir contenir des tableaux de tailles différentes mais aussi d'autres informations complémentaires. Les informations concernant la balle ont aussi été organisées dans un cluster.



(a) Cluster information pour le joueur ou l'ordinateur

(b) Cluster balle

FIGURE 11 – Organisation de l'information

Cette organisation a l'avantage de mettre beaucoup d'information à disposition des actions. Cependant les différents niveaux de cluster imbriqués dans des tableaux rendent l'accès aux informations laborieuse. Par ailleurs il s'avère que toutes les informations ne sont pas utilisées, il est donc possible d'alléger les clusters *infos*. De plus les mêmes informations sont calculées pour le joueur et pour l'ordinateur, or seules quelques informations concernant le joueur sont nécessaires. Les valeurs que peuvent prendre les variables dans les cluster sont explicités dans la documentation (Annexe 2).

3.6 Liste des VI d'informations

Les VI sont explicités en détails dans la documentation du programme.

main_strategie.vi est le VI principal qui comporte tous les autres VI et dont la structure est expliquée plus haut. Il n'est éxecuté qu'une seule fois.

Collecte_données.vi Ce VI reçoit les données bruts et les traite afin d'en tirer des données mieux organisées. Il est executé deux fois à chaque boucle, une fois pour le joueur et une fois pour l'ordinateur.

infos.vi Ce VI reçoit les données organisées et renvoie un cluster *informations* comportant toutes les donnés sur les rangées et les figurines. Ce VI est executé 2 fois à chaque boucle, une fois pour les informations du joueur et une fois pour les informations de l'ordinateur. Il appelle d'autres sous-VI : *zone balle 2D.vi*,*faisceau.vi* et *points figurines pour affichage.vi*

dernier controleur_détenteur de la balle.vi Ce VI modifie les informations propres à la balle. Le dernier contrôleur ou le contrôleur actuel et le possesseur actuel. Ce VI est éxecuter une fois par boucle et n'appelle aucun sous-VI.

zone balle 1D.vi Détermine dans quelle zone se trouve la balle par rapport à une figurine ou une rangée de figurine. Il ne comporte pas de sous-VI

zone balle 2D.vi Détermine dans quelle zone se trouve chaque figurine en faisant appelle au VI *zone balle 1D.vi*, mais aussi de nombreuses informations comme la possession, le contrôle et la distance à la trajectoire. Il appelle les VI *zone balle 1D.vi*, *affine bornée.vi*

affine bornée.vi Ce VI qui n'est rien d'autre qu'une fonction affine avec des extremums. Elle est utilisé pour calculer la largeur et la hauteur d'une zone de possession en fonction de la vitesse. L'axe des abscisses prend donc la dimension d'une vitesse et l'axe y coressepond à la largeur obtenue.

faisceau.vi Ce VI permet de ressortir les directions que la balle peut prendre sans être interceptée par les lignes adverses comme expliqué dans la partie 3.4.Direction de Tir. Il est exécuté deux fois : une fois pour l'ordinateur et une fois pour le joueur.

3.7 Liste des VI d'affichage

dessin balle.vi Calcul une liste de points à partir de la position de la balle dans le but de représenter la balle.

points zone de possession.vi calcul la position des 4 points de la zone de possession pour une figurine à partir de la position de la figurine et de la hauteur et de la largeur de la zone de possession calculée.

points figurine pour affichage.vi Construit des tableaux de points comportant les points définissant les faces exposées, et les points des zones de possessions.

4 Actions

Maintenant que les informations sont calculées, il suffit de lire dans les clusters *informations*,les données nécessaires pour adapter le comportement de l'ordinateur et ainsi éxecuter une action.

Une action détermine les mouvements executés par l'ordinateur, c'est dire qu'une action renvoie une commandes destinés à guider les moteurs. Chaque action possède des conditions d'entrée. Quand une action est choisie le programme lance une boucle *while* dans laquelle est implémentée le calcul des commandes (Figure 2). La boucle *while* est stoppée lorsque les conditions de fin de l'action sont remplies.

4.1 Choix et fin des actions

Sur la base des informations on peut déterminer le comportement de l'ordinateur. Cela se traduit par un choix d'action adapté aux circonstances. Ce processus de choix est effectué par le VI *decision.vi*. Une fois choisie, l'action est exécutée en boucle jusqu'à ce que les conditions de fins soient remplies. Les conditions de fins doivent prévoir le succès comme l'echec de l'action. Par exemple si l'ordinateur tire vers les buts et que le tir est intercepté par le joueur, l'action doit être interrompue.

action	conditions d'entrée	conditions de sortie
1. fermer les issues	possession : joueur	possession : ordinateur ou personne
2. intercepter, stopper	vitesse de balle : élevée possession : personne contrôle : joueur	possession : ordinateur ou joueur
3. contrôler la balle	vitesse de balle : faible possession : personne contrôle joueur ou personne	possession : ordinateur ou joueur
4. tirer	contrôle : ordinateur directions : existantes	possession : personne
5. faire une passe	contrôle : ordinateur directions : aucunes vers les buts	possession : ordinateur ou joueur

Remarque : Certaines actions n'ont pas été implémentées pour plusieurs raisons : par manque de temps mais aussi en raison de l'impossibilité de pouvoir expérimenter le comportement du babyfoot avec la balle. Les actions impliquant le contact avec la balle doivent absolument être testées.

4.2 Liste des actions et VI d'actions

fermer les issues.vi Permet de placer les joueurs dans une position défensive prédeterminée afin de minimiser les ouvertures et ainsi faciliter les actions d'interceptions de la balle en cas d'attaque adverse. Il renvoie une position x et un angle θ adapté et constant à la fonction de chaque ligne.

intercepter_ stopper balle.vi Quand la balle est très rapide et qu'il faut avant tout la stopper dans sa course, ce VI positionne la figurine la plus proche sur la trajectoire de la balle pour une rangée donnée. Il renvoit une position x et un angle toujours nul (c'est à dire une figurine droite).

controle_ blocage_ balle .vi Quand la balle est plus lente alors l'ordinateur peut chercher à la contrôler, c'est à dire à la bloquer sous le pied d'une figurine. Ce VI renvoie donc une commande en position sur x et l'angle permettant de bloquer la balle sous le pied d'une figurine donnée.

tir.vi Ce VI n'est choisi que si la balle est maîtrisée. Une fois sous le pied de la figurine, en une position donnée, la balle doit être amenée à une seconde position d'où elle sera tirée. Ce VI revoie donc une liste de positions sur x et sur θ. Les explications plus détaillées sont dans la documantation.

Recherche_ tir.vi Ce VI se situe juste avant le VI *tir.vi*. Il a pour but de sélectionner la mailleure position sur la ligne qui possède laballe ainsi que la direction de tir. Il envoie à *tir.vi* une position x ainsi qu'un coefficient directeur.

faire une passe Faute de temps et de pouvoir tester les VI précédents ce VI n'as pas été implémenter.

5 Commandes, données de sortie

Les données destinées à la commande des moteurs sont des positions (en mm) ou des vitesses (en mm/s). Le traitement de ces données n'est pas géré par ce programme mais dans le projet de la partie "commandes".
Il y a deux types de commandes en positions. Tout d'abord les positions uniques calculées en continue, par exemple quand l'ordinateur doit suivre la balle, la commande est recalculée aussi fréquement que possible. Les trajectoires constituent le second type de commande en position. Une trajectoire n'est calculée qu'une seule fois et délivrée points par points aux commandes. Il est à noter que les positions sont calculées sur le plan bidimensionnel du terrain. La position d'une figurine dans le plan x,y fixe les deux degrés de liberté : la translation et la rotation. Les commandes sont mémorisées dans un cluster et mise à jour par une variable locale en écriture.
Les commandes en vitesses ont pour but de transmetre à la balle une certaine vitesse dans une direction donnée. La vitesse en translation et la vitesse en rotation sont donc couplées pour orienter la balle dans une direction.

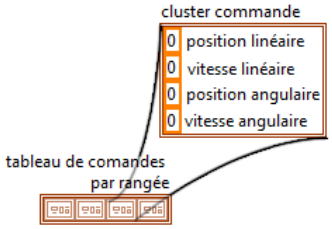


FIGURE 12 – Cluster commande

6 Conclusion

Plusieurs aspects positifs et négatifs ressortent de ce projet. Tout d'abord le but du pojet global était de faire fonctionner les quatres sous projets ensemble. Mais cela fut rendu difficile par l'absence de concertation dès le début du projet. Deuxièmement la stratégie est un programme qui guide un système physique. Il nous a donc été impossible de tester l'interaction entre la balle et des figurines guidées par ordinateur. Ce qui entraine naturellement des difficultés à implémenter des actions, puisque celle-ci ne reposent que sur des hypothèses non vérifiées. Quant au programe lui-même, le principe de choix reposant en grande partie sur les critères de proximité et de la maîtrise de la balle sont assez simples et efficaces. L'adaptation du comportement constitue en effet une part non négligeable d'un tel programe.

Nous tenons à remercier Monsieur Saltzmann qui nous a permis de réaliser ce projet.

7 Annexes

7.1 Organigramme algorithmique

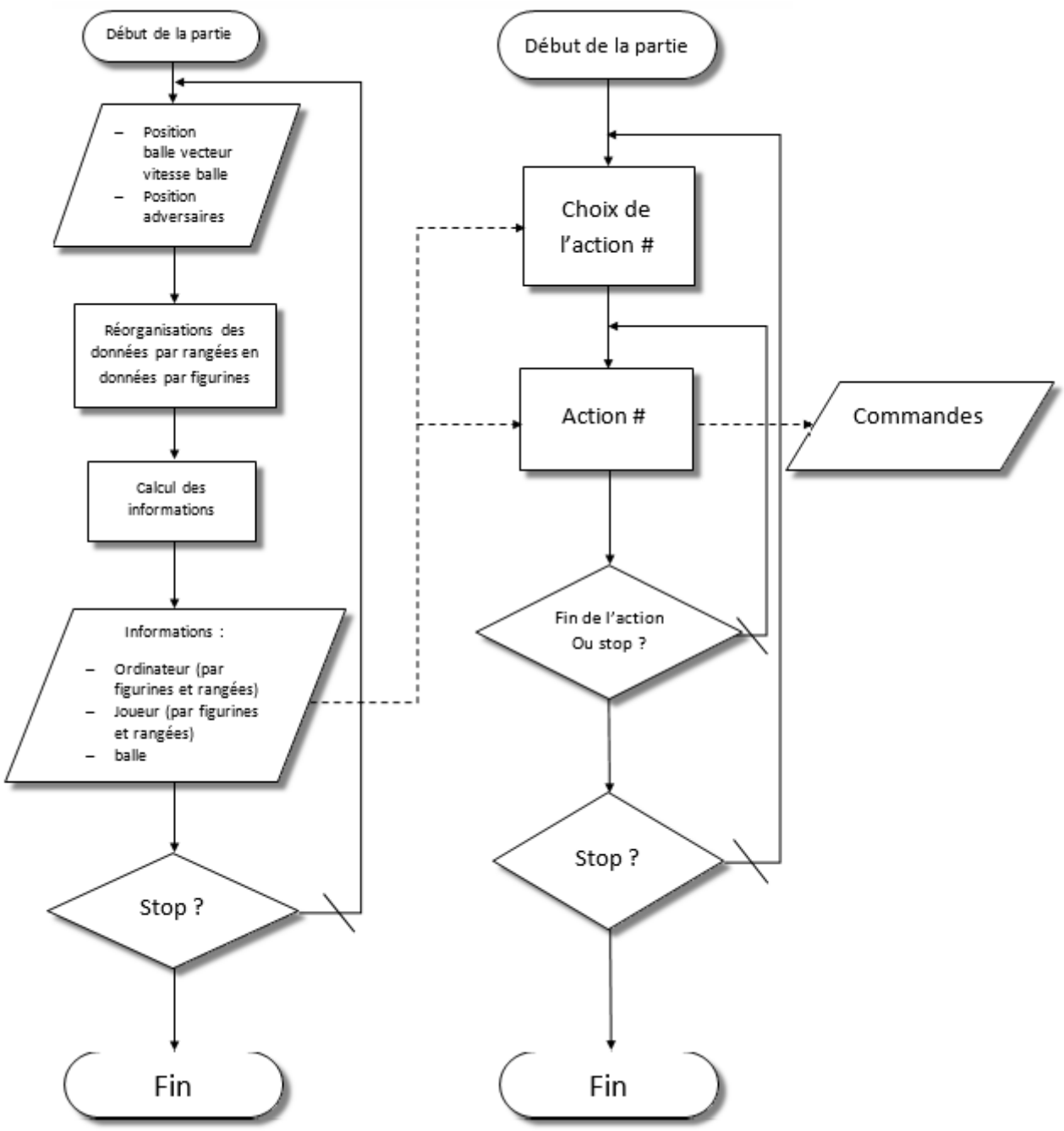


FIGURE 13 – Organigramme algorithmique

- Annexe 2 -
Documentation du programme

Pierre-Benoît Blanc, Gildas Jalon

5 juin 2015

8 Variables

8.1 Données

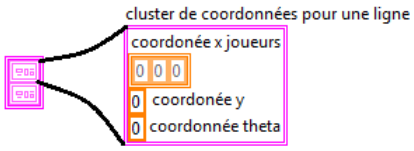


FIGURE 14 – format des coordonnées réorganisées

A chaque rangée correspond une case du tableau. Et dans une case on trouve les valeurs suivantes :

double	*coordonée y*	position de la rangée
double	*coordnnée θ*	angle des figurines d'une rangée
tableau double	*coordonées x joueurs*	contient la position x des figurines d'une rangée

8.2 Informations

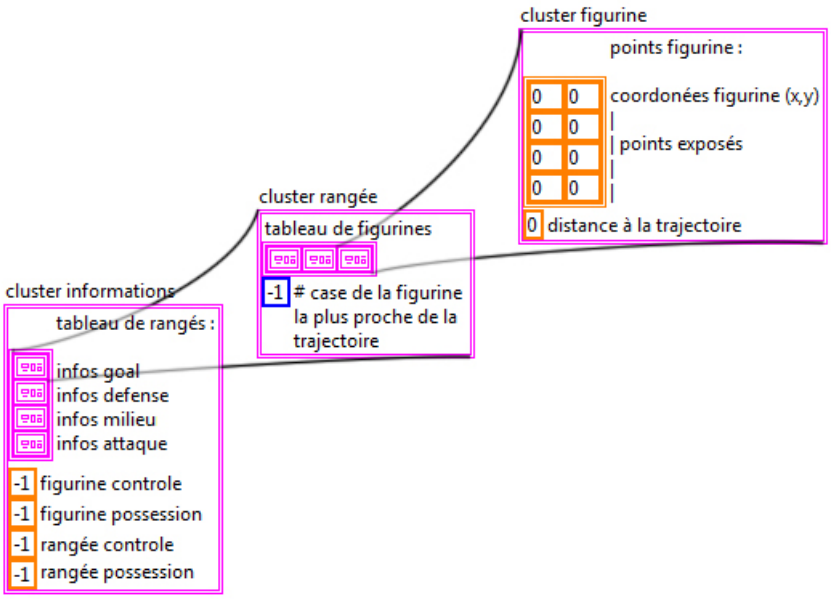


FIGURE 15 – différents niveaux du cluster *informations*

niveau 1 : Informations sur l'ensemble des figurines

double	*figurine contrôle*	$= -1$: aucune figurine ne contrôle la balle
		$\neq -1$: numéro de la case dans le *tableau de figurine*
double	*figurine possession*	$= -1$: aucune figurine ne possède la balle
		$\neq -1$: numéro de la case dans le *tableau de figurine*
double	*rangée contrôle*	$= -1$: aucune rangée ne contrôle la balle

		$\neq -1$: numéro de la case dans le *tableau de figurine*
double	*rangée possession*	$= -1$: aucune rangée ne possède la balle
		$\neq -1$: numéro de la case dans le *tableau de figurine*
tableau de cluster	*tableau de rangées*	

niveau 2 : Informations sur l'ensemble d'une rangée

entier	*figurine la plus proche*	$= -1$: la balle s'écarte de la rangée
	de la trajectoire	$\neq -1$: numéro de case de la figurine la plus proche
tableau de cluster	*tableau de figurines*	

niveau 3 : Informations d'une figurine

double	*distance à la trajectoire*	la valeur prise peut être positive ou négative.
tableau de double	*points figurines*	ligne 0 :coordonnées de la figurines (x,y)
		ligne 1 à 3 : points définissants les faces exposées à la balle

8.3 commandes

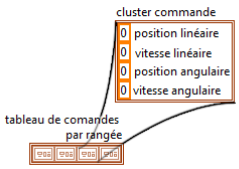


FIGURE 16 – format des commandes

A chaque rangée en jeu correspond une case du tableau.

double	*poosition linéaire*	sur x
double	*vitesse linèaire*	sur x
double	*position angulaire*	sur θ
double	*vitesse angulaire*	sur θ

9 Liste des VI et sous VI

9.1 main_stratégie.vi

Entrées : acunes entrées

Sorties : aucunes sorties

Paramètres :

tableau de booleen *rangées en jeu* TRUE : rangée installée ; FALSE : rangée absente

Explications du VI : Le VI principale se construit en 2 boucles *while*, l'une met à jour les information pour que l'autre puisse choisir et exécuter des actions et délivrer les commandes.

9.2 collecte_entrées.vi

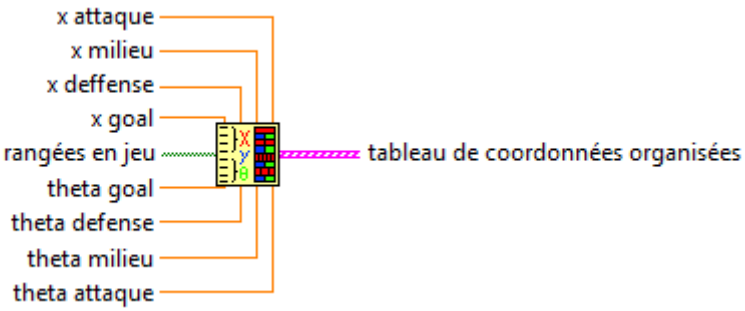


FIGURE 17 – collecte_entrées.vi

Entrées :

double	x_g	$0 \leq x_g \leq 224mm$	position selon x du goal
double	x_d	$0 \leq x_d \leq 418mm$	position selon y de la rangée de défense
double	x_m	$0 \leq x_m \leq 178mm$	position selon y de la rangée du milieu
double	x_a	$0 \leq x_a \leq 238mm$	position selon y de la rangée de l'attaque
double	θ_g	$-\pi \leq \theta_g \leq \pi$	angle du goal
double	θ_d	$-\pi \leq \theta_d \leq \pi$	angle de la ligne de défense
double	θ_m	$-\pi \leq \theta_m \leq \pi$	angle de la ligne du milieu
double	θ_a	$-\pi \leq \theta_a \leq \pi$	angle de la ligne de l'attaque
double	(x_b, y_b)		position de la balle
double	(u_b, v_b)		vecteur vitesse de la balle
tableau de booléen	*rangées en jeu*	TRUE = rangée installée ; FALSE = rangée absente	

Sorties :

tableau de cluster	*coordonnées organisées*

Paramètres :

double *hauteur d'une figurine* (mm).
position des figurines sur la barre (autant de paramètres que de figurines en jeu) (mm).

Explication du VI : Sur la base des positions x et θ de chaque rangée en jeu, ce VI calcule la position de chaque figurine en jeu et créer un tableau de coordonnées contenant le minimum de données nécessaires.

9.3 infos.vi

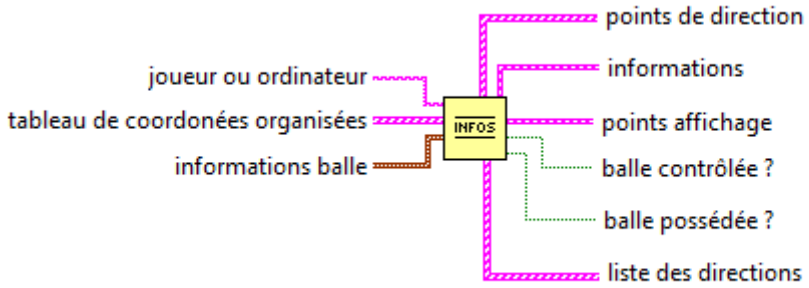


FIGURE 18 – infos.vi

Entrées :

string	*joueur ou ordinateur*	ce string indique simplement pour qui les informations sont calculées.
tableau de cluster	*coordonnées organisées*	
cluster de nombres	*information balle*	

Sorties :

tableau de cluster	*informations*	
booléen	*balle controlée ?*	TRUE = balle controlée ; FALSE = balle non contrôlée.
booléen	*balle possédée ?*	TRUE = balle possédée ; FALSE = balle non possédée.
tableau de cluster	*points de direction*	
tableau de cluster	*liste directions*	
tableau de cluster	*points affichage*	répertorie l'ensemble des points pour XYGraph représentant les faces et la zone de possession des figurine.

Paramètres : aucun paramètre.

Explication du VI : A l'aide du sous vi *zone balle 1D.vi* ce vi partitionne le tableau de *coordonndonées organisées* afin de les traiter séparément en fonction de la position de chaque rangée par rapport à la balle. La recherche des zones se fait dans l'ordre des rangées le long de l'axe y, il y a 3 partitions possibles :

1. Si la balle est trouvée dans une zone 1 cela signifie qu'elle est au dessus de toutes les rangées, dans le cas contraire la balle se trouve dans la zone 2 ou 3 d'une autre rangée.
2. Si la balle est trouvée dans une zone 2 cela signifie que la balle se trouve dans la zone 1 des lignes précédentes et dans la zone 3 des lignes suivantes.
3. Si la balle est trouvée dans une zone 3 cela signifie qu'elle se trouve dans la zone 1 des lignes précedentes et dans la zone 3 des lignes suivantes.

Une fois les coordonées spatiale traitées par le vi *zone balle 2D.vi* on disposes des valeurs nécessaire pour déterminer quelle figurine dans quelle rangée possède et/ou contrôle la balle. Enfin le vi assemble les tableaux de clusters d'informations et les tableaux de cluster de points à afficher calculés dans le vi *zone balle 2D.vi*.

Sous VI :

- *zone balle 1D.vi*
- *zone balle 2D.vi*
- *faisceau.vi*
- *points affichage.vi*

9.4 dernier detenteur_controleur de la balle.vi

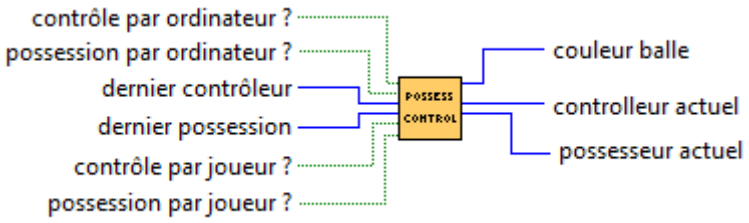


FIGURE 19 – dernier detenteur_controleur de la balle.vi

Entrées :

booléen	*ordi controle ?*	TRUE=l'ordinateur contrôle la balle FALSE=l'ordinateur ne contrôle pas la balle
booléen	*controle joueur ?*	TRUE=le joueur contrôle la balle FALSE=le joueur ne contrôle pas la balle
booléen	*possession ordi ?*	TRUE=l'ordinateur possède la balle FALSE=l'ordinateur ne possède pas la balle
booléen	*possession joueur ?*	TRUE=le joueur possède la balle FALSE=le joueur ne possède pas la balle
entier	*dernier controleur*	0 = personne ; 1 = ordinateur ; 2 = joueur ;
entier	*dernier detenteur*	0 = personne ; 1 = ordinateur ; 2 = joueur ;

Sorties :

entier	*couleur balle*	vert (personne contrôle la balle), rouge (l'ordinateur contrôle la balle), bleu (le joueur contrôle la balle)	destiné à l'affichage
entier	*dernier contrôleur*		
entier	*dernier détenteur*		

Paramètres : La durée d'attente avant de modifier la valeur *dernier contrôleur* à 0 si personne ne touche la balle de nouveau peut se modifier dans le sous-vi *Elapsed Time.vi*.

Explication du VI : Ce VI change la valeur de *détenteur actuel* selon le possesseur la balle. Pour la valeur de *controleur actuel* la valeur est changée en 0 si personne ne contrôle la balle depuis une certaine durée paramétrisalble dans le sous-vi *Elapsed Time.vi*, sinon la variable prend la valeur de 1 ou 2.

9.5 zone balle 1D.vi

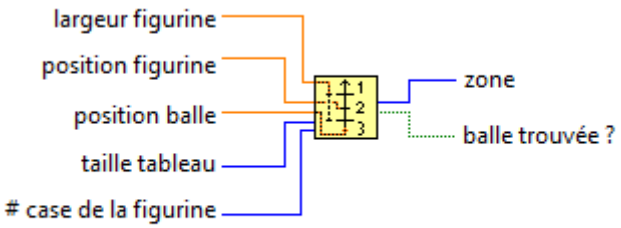


FIGURE 20 – zone balle 1D.vi

Entrées :

double	*largeur figurine*	Il peut s'agir de la largeur ou de la longeur du pied d'une figurine la valeur peut être autre que celle d'une figurine car le vi est aussi utilisé pour la zone de possession.
double	*position figurine*	sur l'axe x ou y
double	*position balle*	sur le même axe que la position de la figurine.
entier	*taille tableau*	tableau contenant les coordonées de figurines où la balle est cherchée.
entier	*case*	numéro de la case du tableau d'une figurine.

Sorties :

entier	*zone*	numéro de la zone 1,2 ou 3
booléen	*balle trouvée ?*	indique si la balle est trouvée (dans une zone 3 ou 2) dans l'ensemble des rangées ou des figurine.

Paramètres : aucun paramètre.

Explication du VI : Ce VI détermine la zone dans laquelle se trouve la balle par rapport à une rangée (sur y) ou une figurine (sur x). Le VI détermine aussi si la balle est définitivement trouvée. En effet ce vi est toujours appelé dans une boucle parcourant les rangée ou les figurine d'une rangée en suivant l'axe correspondant (y pour les rangées et x pour les figurine). Il y a trois zones :

- zone 1 : la balle se trouve devant la rangée ou à droite de la figurine. La balle n'est pas définitivement trouvée.
- zone 2 : la balle se trouve au niveau de la rangée ou en face de la figurine. La balle est définitivement trouvée.
- zone 3 : la balle se trouve derrière la rangée ou à gauche de la figurine. La balle es définitivement trouvée.

9.6 zone balle 2D.vi

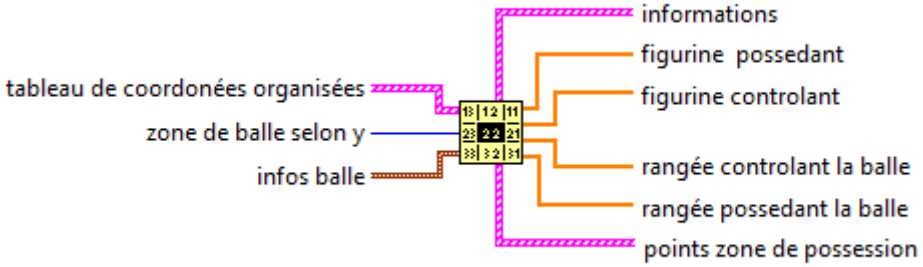


FIGURE 21 – zone balle 2D.vi

Entrées :

tableau de cluster	*coordonnées organisées*	
entier	*zone de balle selon y*1, 2 ou 3	
cluster	*infos balle*	

Sorties :

tableau de cluster	*informations*	
tableau de cluster	*points zone de possession*	liste des points des zones de possession des figurines (destinés à l'affichage)
tableau de double	*figurines possedant*	chaque case est associée à une figurine, -1 = la figurine ne possède pas la balle n = la n^{ieme} figurine d'une rangée possèdant la balle (commence à 0)
tableau de double	*rangée possedant*	chaque case est associée à une rangée, -1 = la rangée ne possède pas la balle n = la n^{ieme} rangée possède la balle (commence à 0)
tableau de double	*figurines contôlant*	chaque case est associée à une figurine, -1 = la figurine ne contrôle pas la balle n = la n^{ieme} figurine d'une rangée contrôle la balle (commence à 0)
tableau de double	*rangée contrôlant*	chaque case est associée à une rangée, -1 = la rangée ne possède pas la balle n = la n^{ieme} rangée possède la balle (commence à 0)

Paramètres :

double	*largeur du pied (sur x)*	
double	*longueur du pied (sur y)*	
double	*dixstance de blocage*	distance sur y entre l'axe d'une rangée et le pied d'une figurine pour maintenir la balle sous le pied.
double	*espacement des figurines d'une rangées*	
double	*diamètre de la balle*	
double	*marge zone de contrôle*	
double	$u_{bmax}, u_{bmin}, v_{bmax}, v_{bmin}$	entrés du VI *affine bornée.vi*

Explication du VI : Sur la base des coordonées spatiales reçues dans *coordonnées organisées* le VI calcule la plupart des informations. Dans le tableau *coordonnées organisées* une paire de case est associée à une rangée. Il peut s'agir d'une portion des rangée selon la partition effectuée dans le VI supérieur *infos.vi*. Les informations calculées sont les suivantes :

- Des points déffinissant les faces exposées à la balle pour chaque figurine
- La distance à la trajectoire pour chaque figurine
- La figurine la plus proche de la trajectoire dans chaque rangée
- Des tableaux destinés à déterminer quelle figurine de quelle rangée possède et/ou contrôle la balle.

9.7 affine bornée.vi

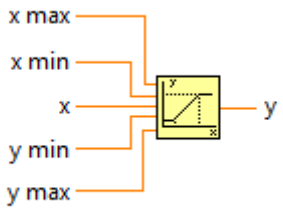


FIGURE 22 – affine bornée.vi

Entrées :

double	x_{min}	(paramètre d'une fonction affine bornée)
double	x_{max}	(paramètre d'une fonction affine bornée)
double	y_{min}	(paramètre d'une fonction affine bornée)
double	y_{max}	(paramètre d'une fonction affine bornée)
double	x	valeur en abscisse

Sorties :

double	y	valeur en ordonnée

Paramètres : Aucun paramètre

Explication du VI : Ce VI est l'implémentation d'une fonction affine bornée en deux points $(x_min; y_min)$ et $(x_max; y_max)$. Dans le programme ce VI est utilisé afin de déterminer la taille de la zone de possession en fonction de la vitesse de la balle. La largeur de la zone de possession (axe x) deépend de la vitesse v_b et la longugueur (axe y) dépend ed la vitesse u_b. Dans les deux cas la largeur diminue quand la vitesse augmente. Les largeur et longueur minimales (quand les vitesses hotizontales et verticales sont maximales) sont égales aux dimensions de la figurine. La largeur maximale est égal à l'espacement entre les figurine et la longueur maximale est égale au double de la distance de perte de contact avec la balle.

9.8 faisceau.vi

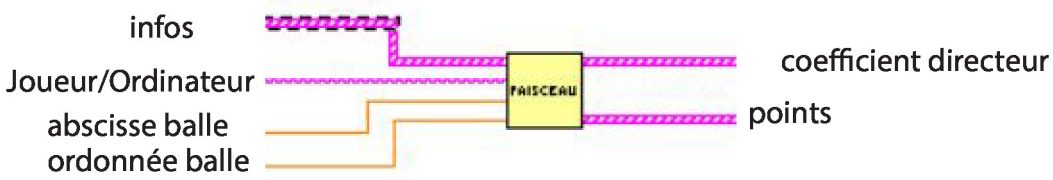


FIGURE 23 – faisceau.vi

Entrées :

tableau de cluster	*Informations*	Seul les lignes situées entre la balle et l'un des goals composent ce clus
string	*Direction*	Donne le sens de la recherche, soit vers les buts du joueur() soit vers c
double	*abscisse de la balle*	
double	*ordonnée de la balle*	

Sorties :

tableau de cluster	*Coefficient directeur*	Cluster composé dún string donnant la destination du coefficient et un
tableau de cluster	*points*	Cluster composé de tableaux de dimension 2 contenant les coordonées

Paramètres :

entier	*distance de sécurité*	Permet de calculer les diférents coefficients en étant sûr que la balle pa
tableau de doubles	*initialisations des bords du terrains*	

Explication du VI : Ce VI se divise en quatre parties. La première consiste à transformer le cluster d'information reçu par le VI "zone balle 2D" et le transformer en un tableau de double qui pourra être utilisé pour effecteur une recherche d'obstacle. Si le cluster d'entrée provient de "infos Joueur", il est tout d'abord inverser pour que la ligne du goal soit la première lue. Ensuit, une décomposition ligne par ligne du Cluster est faite afin de prendre pour chaque figurine, le couple (x,y) des coins de la face de la figurine exposée au ballons. Pour la coordonnée x, 2 cm sont ajouté comme coefficient de sécurité.

De part et d'autre de chaque ligne, un couple de deux points [($\pm$ 10000, ordonnée de la ligne) ; (bord du terrain, ordonnée de la ligne)] est inséré. Cela à pour but de détruire les directions qui partiraient hors du terrain. Cela nous permet aussi de modéliser les buts.

A partir de ce moment là, la recherche de faisceau peut s'effectuer. En soustrayant les coordonnées de la balle à tout les points du tableaux, le centre du repère est déplacé au centre de la balle. Un tableau contenant les deux coefficients limite de la balle est initialisé. Le programme va lire deux lignes du tableau et va transformer les coordonnées de chaque en un coefficient directeur en utilisant la relation

$coef = \arctan(\alpha) = x \div y$
Les deux coefficients directeur obtenus (appelons les faisceau «figurine») représentant les différents obstacles vont être comparé à chaque couple de coefficient directeur représentant les zone de liberté de la balle (faisceau «balle»). Ainsi trois situations se présentent :

- **le faisceau «figurine» est compris dans le faisceau balle :** Dans ce cas-là, le faisceau «balle» n'a d'autre choix que d'être scindé en deux. Nous obtenons ainsi deux faisceaux «balle». Pour cela nous insérons tout simplement entre le faisceau «balle» étudié, les nouveaux coefficients trouvés.
- **un seul coefficient du faisceau figurine est compris dans le faisceau «balle» :** Ici, nous devons remplacer le coefficient directeur de la balle qui est dans les faisceau « figurine »par le coefficient directeur de la figurine qui se situe dans le faisceau «balle». Pour cela nous insérons le coefficient directeur de la figurine dans le tableau de faisceau entre les deux coefficients directeurs de la balle puis nous supprimons coefficient directeur de la balle inutile.
- **Le faisceau «figurine» est hors du faisceau de la «balle»**

Ce qui fait la force de ce programme, c'est qu'il ressort les différents coefficients directeur pour atteindre chaque ligne contrôlé par l'ordinateur. Pour se faire il faut tout d'abord savoir combien de lignes la balle devra traverser pour arriver jusqu'au but. En connaissant la taille du cluster infos il est très aisé connaître le nombre de lignes. Ensuite, il est nécessaire d'avoir un marqueur permettant de connaître la fin d'une ligne, il s'agit de l'abscisse du bord de chaque ligne, en l'occurence le nombre "10000". A chaque fois que le programme lira cette valeur, il décrémentera le nombre de lignes restantes et enverra un string désignant la ligne de figurine de l'ordinateur qui peut intercepter ou recevoir la balle. Ce string sera la première partie du cluster de sortie "coeffients directeurs". Chaque coefficient directeur sera la moyenne des angles des deux extrémités des faisceaux trouvés jusqu'à cet instant.

Il y a cependant un cas particulier, lors du passage de la ligne des défenseurs à la ligne du gardien du côté du joueur, il n'y a pas de ligne contrôler par l'ordinateur. Dans ce cas là il ne faut pas créer, lorsque l'on est à la fin de la ligne des défenseur, de nouveau cluster. Cependant comme nous sommes dans une boucle condition, nous sommes obligés de retourner une valeur. Nous avons décider qu'il s'agirait d'un cluster constant.

Par ailleurs comme cette boucle condition se trouve dans la première boucle for qui lit chaque figurine, le cluster constant est créer à chaque nouvelle figurine lu.

Nous ressortons de la boucle for avec un cluster contenant les différents clusters de lignes plus les cluster nuls.

Nous obtenons ainsi de nombreux clusters parasites qu'il va falloir détruire. C'est ce qui se passe dans les deux boucle for suivantes. La première retrouve ces cluster parasites et stocke leur position dans un array. Le deuxième le lit l'array à l 'envers et supprime les clusters correspondant se situant aux positions stockées dans l'array.

Dans cette dernière partie, nous nous occupons de créer les deux types de cluster de sorties pour la suite du programme. Nous avons décider d'avoir deux types de sorties : la première serait les coefficients directeurs qui permettent de projeter la position de la balle sur toute sa direction. La deuxième serait les points se situant sur chaque ligne contrôlées par l órdinateur.

Nous initialisons un cluster composé de deux tableaux correspondant à la position intiale de la balle et sa position finale. Chaque tableau est de dimension 2 et représente la coordonnée x et la coordonnée y. Grâce à un shift register, chaque nouveau point créer sera ajouter au cluster précédent. il sort à la fin, un cluster composé de cluster formé par deux tableau de dimensions 2. Par ailleurs un autre tableau est initialisé dans lequelle les différents coefficients directeurs seront stockés. Ce tableau est réinitialisé à chaque changement de ligne. Une fois ce tableau complété, il sera assembler avec un string dans un cluster afin de connaître les destinations de ces directions. Pour la créations des points, nous utilisons la relations des droites affines avec la coordonnée y est l'antécédent tandis que x est l'image de la fonction. En effet comme le repère est centré sur la balle la coordonée y des différente donnent aussi les distance à parcourir dans cette direction.En connaissant aussi le coefficient directeur de la trajectoire désiré (CD), il est alors aisé d'btenir coordonnée en x du point d'arrivé :

$$xligne = CD \times yligne + xballe$$

Après cela, le programme vérifie si chaque destination se situent dans la limite du terrain, si c'est le cas, le programme va empiler le cluster au cluster principal et va faire de même avec le tableau de coefficients directeur. Dans le cas contraire le programme ne stocke rien.

9.9 decision.vi

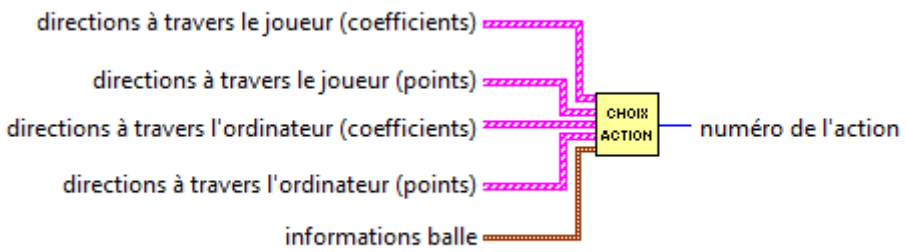


FIGURE 24 – decision.vi

Entrées :

tableau de cluster	*direction à travers le joueur coeff*	
tableau de cluster	*direction à travers l'ordinateur coeff*	
tableau de cluster	*direction à travers le joueur points*	
tableau de cluster	*direction à travers l'ordinateur points*	
cluster	*balle*	

Sorties :

entier	*numéro de l'action*	1, 2, 3, 4 ou 5 selon l'action choisie

Paramètres :

double	*vitesse limite*	vitesse distinguant les vitesses lentes et rapides

Explication du VI : Sur la base des informations calculées ce VI choisit l'action adéquat au circonstance du jeu selon les règles dans le tableau suivant.

9.10 intercepter_stopper balle.vi

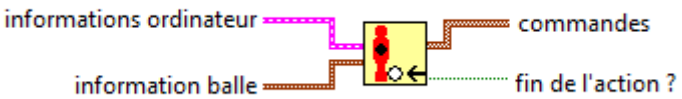


FIGURE 25 – intercepter_stopper balle.vi

Entrées :

tableau de cluster	*informations ordinateur*	
cluster	*iniformation balle*	

Sorties :

tableau de cluster	*commandes*	
boolléen	*fin de l'action ?*	TRUE = l'actoin doit être stopper FALSE = l'action ppeut être continuée

Paramètres :

double	*vitesse limite*	vitesse distinguant les vitesses lentes et rapides
double	*rayon balle*	

Explication du VI : Le tableau d'informations et le cluster de la balle fournissent les informations nécessaires pour placer la figurine la plus proche de la trajectoire sur la trajectoire de la balle. Ce VI est choisi quand la vitesse de la balle est rapide, donc par manque de précision l'ordinateur préfère ne pas bloquer la balle mais la stopper simplement. La position angulaire renvoyée est donc 0. On a donc les commandes suivantes pour une rangée :

$$x_{commande} = x_i + D \qquad \theta_{commande} = 0$$

avec x_i la position initiale de la rangée et D la distance à la trajectoire.

9.11 Fermer_les_issues.vi

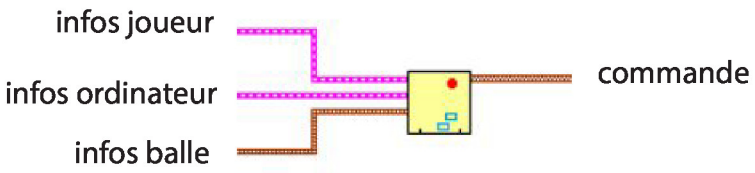


FIGURE 26 – fermer_les_issues.vi

Entrées :

tableau de cluster	*informations ordinateur*	
tableau de cluster	*informations joueur*	
cluster	*iniformation balle*	

Sorties :

tableau de cluster	*commandes*	
boolléen	*fin de l'action ?*	TRUE = l'actoin doit être stopper FALSE = l'action ppeut être continuée

Explication du VI : Ce Vi a pour but de faire en sorte de laisser le minimum d'ouverture à partir du moment ou l'advesaire contrôle la balle et mais n'a pas encore tirer. Ce programme ne gère pour l'instant que le cas où il n'y a que les lignes de défense et de but. En effet, la manière de gérer l'ouverture des issues diffèrent énormément en fonction du nombre de ligne se situant entre la balle et le goal.
Dans notre cas la stratégie est relativement simple, une des figurines de la ligne de défense va suivre la balle dans sur l'axe x. La figurine du goal va faire de même mais avec un offset afin de couvrir le maximum de surface.

Les angles des figurines : Une bonne méthode de défense est de poster les figurines de la ligne de défense en retrait, c'est-à-dire avec un angle de -30 degré tandis que la figurine du goal, quant à elle, se mettra en avant avec un angle de 30 degré. Ainsi chaque ligne à une fonction bien particulière :

- la défense, étant face à la balle, c'est elle qui a le plus de chance de l 'intercepter. Par sa position angulaire, la figurine va plutôt ralentir la balle et la décaler afin que celle-ci tape le fond du terrain pour pouvoir ensuite la contrôler plus facilement.
- le goal quant à lui, est plutôt une solution de dernier secours. En effet, en étant en avant, la balle aura plus tendance à rebondir sur la figurine, elle ne perdra que peu de vitesse lors de l'impact. De plus en étant le plus éloigné des buts, l'angle nécessaire pour que la balle soit correctement dévié diminue fortement.

Ensuite il faut savoir de quel côté va se placer le goal par rapport à la figurine de la défense qui va suivre la balle. Pour choisir, il faut regarder quelle est la figurine adverse qui contrôle la balle. S'il s'agit de la figurine du goal, la position du goal de l'ordinateur est peu importante. Cela est dû au fait que la marge est de manœuvre de la ligne du goal est contenu dans ses buts. Donc l'angle

maximum que peut effectuer la balle pour arriver au but adverse en partant du goal est plutôt petit (environ 20 degré). On peut donc considérer que les tirs en provenance du gardien seront majoritairement droit et peuvent être facilement arrêter par la ligne de défense.
Par contre, si c'est la ligne de défense qui possède la balle, la situation se complexifie. Il faut d'abord regarder quelle figurine contrôle la balle. En effet, le champ d'action d'un figurine de la défense va d'un bord du terrain jusqu'au bout des buts.

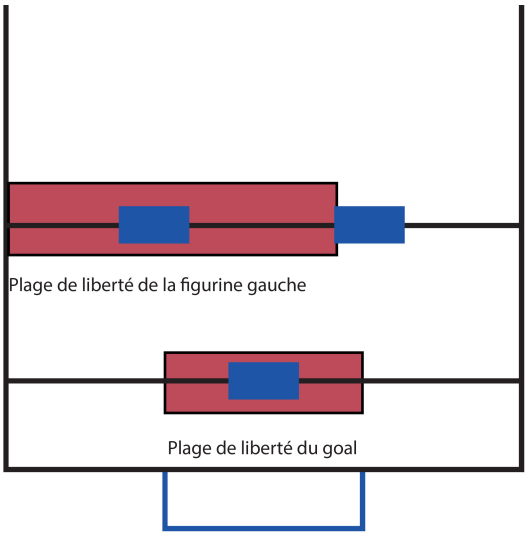


FIGURE 27 – plage de mouvement des deux lignes de la défense

Ainsi afin d'optimiser le tir, il est généralement fait de tel sorte que la balle part d'une extrémité du terrain et se décale vers le milieu du terrain au niveau d'une ouverture afin de créer un décalage entre la ligne de défense et la ligne d'attaque de l'adversaire. Afin de rendre encore plus difficile d'arrêter ce tir, il est généralement croisé. La manière la plus commune de le faire est de laisser rouler la balle le long de la ligne de défense et d'attendre que celle-ci passe devant le joueur qui va effectuer son tir. Ainsi avec le temps nécessaire pour armer le tir, la balle a continuer de se se décaler et de cette manière, lors de l'impact, la balle partira avec un angle dans la direction du décalage. Par conséquent, la méthode pour placer le goal est de le mettre du côté opposé à la figurine qui possède la balle. De cette mainère, les deux figurines chargées de fermer les issues vont former un segment qui sera plus ou moins dans la même direction que la trajectoire d'une tiré avec un angle.

Un dernier point important est le changement de figurine de la ligne de défense. Une règle essentielle en défense est de ne jamais changer de joueur lorsque la balle est proche d'une figurine de l'adversaire. Le changement de figurine s'effectue lorsque l ádversaire le fait lui aussi. Cela laisse une petite durée durant laquelle nous savons que la balle ne pourra pas être tirer. Ce moment est lorsque la figurine adverse dépasse la moitié du terrain. A ce moment là, la figurine n'a que le choix entre tirer ou faire la passe. Comme nous connaissons la zone de possession d'une figurine nous pouvons savoir si le joueur à l'intention de tirer ou de faire une passe. Si la balle sort de cette zone est qu'elle va dans la direction de l'autre figurine le changement peut s'effectuer.

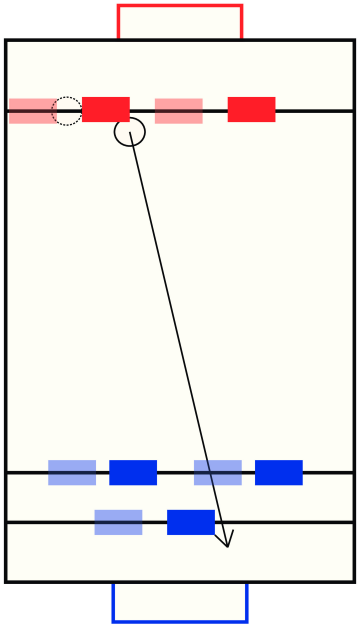


FIGURE 28 – suivi de la balle lors d'un dácalage vers la droite

9.12 controle_blocage balle.vi

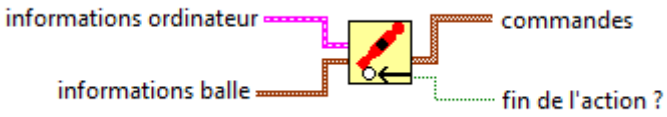


FIGURE 29 – controle_blocage balle.vi

Entrées :

tableau de cluster	*informations ordinateur*	
cluster	*iniformation balle*	

Sorties :

tableau de cluster	*commandes*	
boolléen	*fin de l'action ?*	TRUE = l'actoin doit être stopper FALSE = l'action ppeut être continuée

Paramètres :

double	*vitesse limite*	vitesse séparant les vitesses lentes et rapides
double	*rayon balle*	
double	*vitesse limite*	
double	*hauteur d'une figurine*	

Explication du VI : Le tableau d'informations et le cluster de la balle fournissent les informations nécessaires pour placer la figurine la plus proche de la trajectoire sur la trajectoire de la balle. Ce VI est choisie quand la vitesse de la balle est lente, l'ordinateur va donc chercher à bloquer la balle sous le pied de la figurine. Ceci implique que la figurine soit inclinée d'un certain angle. Les commandes envoyées vallent donc :

$$x_{commande} = x_i + D + d\frac{u_b}{v_b} \qquad \theta_{commande} = \arcsin(\frac{d}{h})sign(v_b)$$

Avec x_i la position initiale de la rangée, D la distance à la trajectoire, d la distance correspondant à l'angle pour bloqué la balle, h la hauteur d'une figurine, u_b et v_b les composantes du vecteur vitesse de la balle.

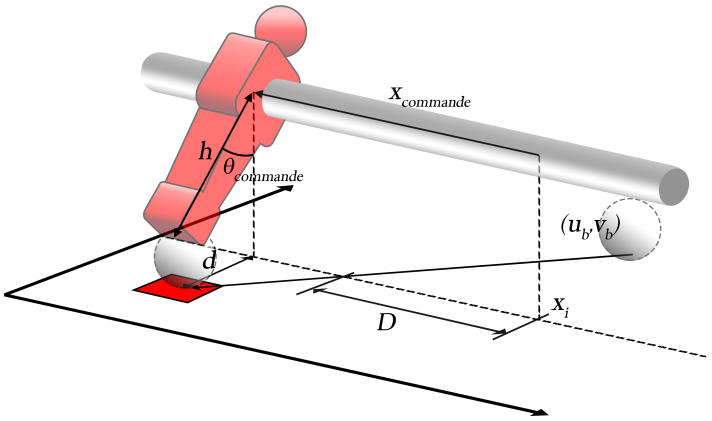


FIGURE 30 – Blocage de la balle en position finale

9.13 tir.vi

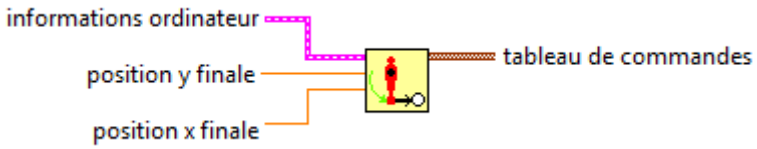


FIGURE 31 – tir.vi

Entrées :

tableau de cluster	informations ordinateur
double	position x finale
double,	position y finale

Sorties :

tableau de cluster	tableau de commandes

Paramètres :

double	pente initiale $-\pi \leq \theta_g \leq \pi$

Explication du VI : Ce VI est choisi si la balle est controlée par l'ordinateur, c'est à dire si elle se trouve dans la zone 22. On suppose alors que la balle est corretement bloquée sous le pied d'une figurine. Le VI reçoit une *position x* et une *position y* déterminant un point depuis lequel il est possible de tirer dans un couloir. Le VI calcul donc une trajectoire pour la figurine afin de guider la balle jusqu'à ce point. La trajectoire est ajusté sur celle d'un cercle dont la tengeante au point initiale est paramétrable.

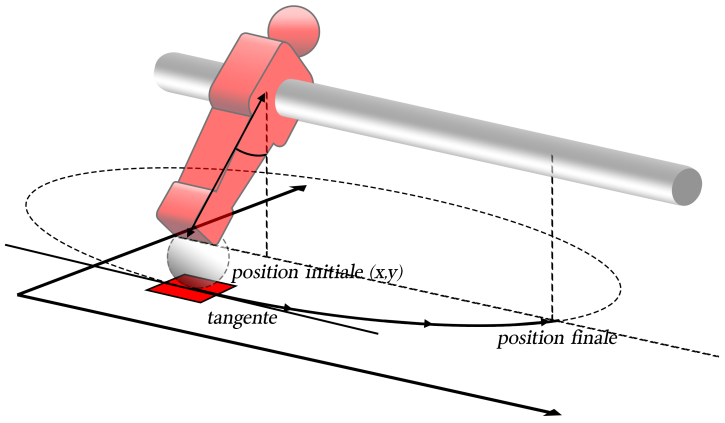


FIGURE 32 – trajectoire tracée par la figurine

9.14 Recherche_tir.vi

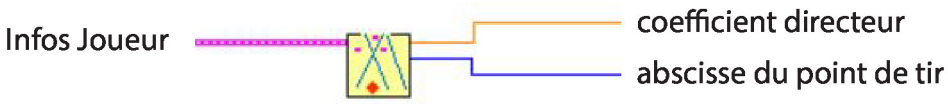


FIGURE 33 – Recherche_Tir.vi

Entrées :

tableau de cluster	*informations joueur*	

Sorties :

double	*coeficient diercteur*	
entier	*absisse du point de tir*	
boolléen	*fin de l'action ?*	TRUE = l'actoin doit être stopper FALSE = l'action ppeut être continuée

Paramètre :

tableau entier	*les abscisses choisies pour effectuer un tir potentiel*	
boolléen	*fin de l'action ?*	TRUE = l'actoin doit être stopper FALSE = l'action ppeut être continuée

Explication du VI : Ce Vi ressemble beaucoup à celui des recherches de faisceaux. Le but de ce VI est de rechercher la meilleur point pour effectuer un tir en direction des buts adverses. Pour cela, nous prenons treize points répartis le long de ligne de défense, et effectuer une recherche de faisceau pour chacun de ces points.

La première partie de ce VI cherche les deux extrémités des figurines de la ligne de défense du joueur correspondant à la face exposée. Ces valeurs sont insérer dans un tableau de constantes afin d'obtenir un tableau similaire à celui du VI faisceau.

Ensuite, le programme initialise un tableau de treize valeurs correspondant aux différentes abscisses où nous allons chercher les possibilité de tir. Le Vi va ensuite exécuter 13 fois la recherche d'obstacle. Les angles maximum et minimum pour chaque faisceau trouvé sont ensuite soustraie entre eux afin de sortir le plus grand faisceau pour chaque point. On transforme ce faisceau en degré grâce à la fonction "arctangente" et en faisant la moyenne de ce faisceau, nous trouvons le meilleur coefficient directeur pour chaque points. On compare cette fois les 13 différences d'angle afin de faire faire ressortir le meilleure point pour attaquer.

9.15 dessin balle.vi

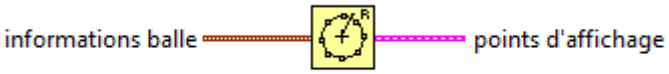


FIGURE 34 – dessin balle.vi

Entrées :

cluster	*balle*	

Sorties :

tableau de cluster	*points balle*	destiné à l'affichage

Paramètres : Aucun paramètre

Explication du VI : Ce VI calcul 30 points à affichés autour de la position de la balle.

9.16 points zone de possession.vi

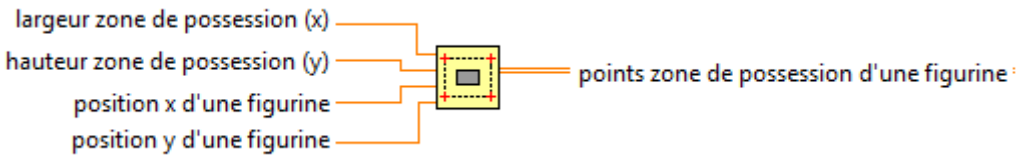


FIGURE 35 – points zone de possession.vi

Entrées :

double	*largeur de la zone de possession* (x)
double	*longueur de la zone de possession* (y)
double	*position* (x) *de la figurine*
double	*position* (y) *de la figurine*

Sorties :

tableau de double	*points zones de possession d'une figurine*

Paramètres : Aucun paramètres

Explication du VI : Ce VI calcul la position des 4 points de la zone de possession d'une figurine.

9.17 points figurines pour affichage.vi

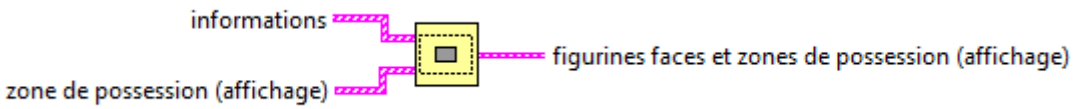


FIGURE 36 – points figurines pour affichage.vi

Entrées :

tableau de cluster	*informations*	
tableau de cluster	*zone de posession figurine*	contient l'essemble des points de chaque zone de possession.

Sorties :

tableau de cluster	*figurine faces et zones de possession*

Paramètres : Aucun paramètre

Explication du VI : La positions des points deffinissant les faces exposées de chaques figurines se trouvent dans le tableau de cluster *informations*. L'ensembles des points deffinisant les zones de possession se trouvent dans le tableau de cluster *zone de possession figurine*. Ce VI rassembles tout ces points dans un nouveau tableau de cluster *figurines faces et zones de possession*. Ce tableau est directement envoyé au *graphxy*.