**Note sur SPY**

**Diagramme séquence :**

**Fiche technique système/script/component/prefab :**

EndZoneActionBloc :

C’est un bloc situer en fin de séquence sur les containers. Il contient une drop zone qui permet de déterminer a quel endroit un autre bloc est déposé. Lorsqu’un bloc est déposé sur le drop zone du bloc EndZoneActionBloc, le bloc déposé prend la place du EndZone dans la hiérarchie et le EndZone descend d’un cran.

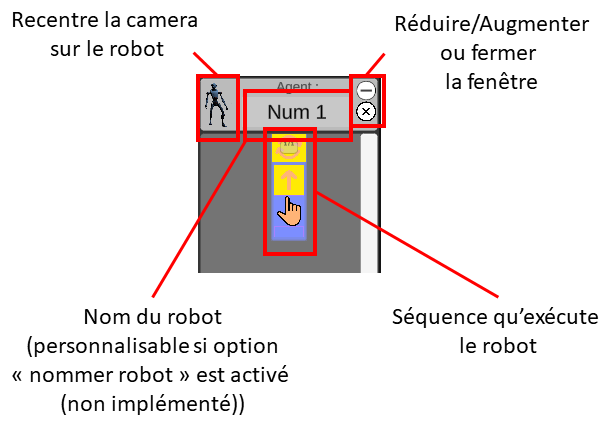
DropZone :

Ce n’est pas un prefab, mais c’est élément souvent utilisé. Sur dropZone est attaché le component « Drop Zone Component ». Ce component permet de signaler quel est le container cible de la drop zone. Soit la drop zone se trouve dans un container et donc la variable « Target » continent le container en cible, soit la drop zone est rattaché à un bloc. Si elle est rattaché à un bloc, la « Target » devient le bloc et la variable « Parent Target » doit être mise à True. Durant l’exécution cela permettra d’identifier que le container n’est pas la « Target » mais le parent de la « Target ».

**Lexique :**

Fenêtre du robot :

Apparait lorsque l’on clique sur le robot, ou lors de l’exécution du code. Permet de visualiser le séquence que le robot va exécuter (lorsque que l’exécution est activé). Permet d’avoir le nom du robot (et pouvoir le modifier si l’option est activé). Permet de centrer la caméra sur le robot.



**Fonctionnalités :**

**Liste des fonctionnalités :**

F1 : Activer l’utilisation de la librairie pour piocher les bocks et créer la séquence d’action du robot. (glisser / déposer ou bien utilisation de la console)

F2 : Activer les blocks d’action que l’on peut utiliser dans le niveau

F3 : Possibilité de glisser / déposer les éléments (venant de la librairie ou présent dans les différentes séquences en construction).

F4 : Initialiser une séquence incomplète pour un robot, le joueur doit la compléter.

F5 : Activer l’historique.

F6 : Niveau avec l’utilisation de boucle " For ".

F7 : Niveau avec l’utilisation de structure de condition " If… Else ".

F8 : Activer le bouton " play " permettant de résoudre la séquence attacher au robot.

F9 : Pouvoir renommer le robot et le container de séquence afin de lier les deux ( si les deux entité on le même nom)

F10 : Parmi une liste d’action, sélectionner la meilleur afin de résoudre le niveau.

F11 : Pauser des questions à l’utilisateur sur la séquence que viens de réaliser le robot

F12 : Parmi une liste de séquence, sélectionner la meilleur afin de résoudre le niveau.

F13 : Mécanique de jeu permettant de peindre le robot afin qu’il puisse passer devant des détecteurs.

F14 : Pouvoir sélectionner des séquences entières au lieu de simple block.

F15 : Séquence incomplète (à compléter avec d’autre séquence contrairement à F4).

F16 : Séquence non optimal (à optimiser).

F17 : Séquence à débuguer.

F18 : Niveau avec l’utilisation de boucle " While ".

F19 : Niveau avec l’utilisation des opérateurs " Et ", " Ou " et " Not ".

F20 : Sélectionner la bonne sortie correspondant à la séquence du robot.

F21 : Avoir plusieurs sortie et édicter un critère de distance pour la création de séquence du joueur.

F22 : Sauvegarder une séquence et l’enregistrer sous la forme d’une icons sélectionnable dans la librairie.

F23 : Activer le mode console

F24 : Sélectionner le mauvais block dans le séquence proposé afin de résoudre le niveau

**Association avec les éléments du jeu :**

**F1 :**

Activation de l’objet " library " qui contient la bibliothèque des actions, opérateurs, éléments et contrôles. Mais si cette fonctionnalité n’est pas activé, alors la librairie n’est pas disponible pour le joueur (utile lors de l’utilisation de fonctionnalité à niveau spécifique comme F10).

Cela ce passe dans le système **LevelGenerator** au niveau de la fonction **activeDesactiveFunctionality()**.

**F2 :**

Lors de la création de niveau, le nombre de bloc adéquate (ou voulue par le créateur du niveau) sont mise à disposition des joueurs. Mais si cette fonctionnalité n’est pas activé, alors aucun bloc n’est à disposition pour le joueur (utile lors de l’utilisation de fonctionnalité à niveau spécifique comme F10).

Cela ce passe dans le système **LevelGenerator** au niveau de la fonction **readXMLLimits()**.

**F3 :**

Activation du Système Drag and drop afin de manipuler les éléments de la librairie et des séquence afin de construire des séquences complètes. Mais si cette fonctionnalité n’est pas activé, alors aucun bloc ne peut être drag and drop par la joueur et donc aucune séquence ne peut être construite(utile lors de l’utilisation de fonctionnalité à niveau spécifique comme F24 ou l’on a besoin d’une séquence prés construite mais pas que le joueur puisse bouger ou supprimer des blocs)).

Cela ce passe dans le système **LevelGenerator** au niveau de la fonction **readXMLLimits()**.

F4 :

Appel d’une fonction qui créera une séquence non complète pour que le robot atteigne la sortie.

**F5 :**

Rend accessible le bouton " NextStepButton " qui permet, quand l’exécution de la séquence est en pause, de l’exécuter étape par étape. Si la fonctionnalité est désactivé, le joueur n’a plus la possibilité de faire effectué la séquence du robot pas à pas.

Cela ce passe dans le système **LevelGenerator** au niveau de la fonction **activeDesactiveFunctionality()**.

**F6 :**

Lors de la sélection des niveaux, le système ne sélectionnera que les niveaux ayant été créer pour l’utilisation d’une boucle FOR. Cela se passe dans le système **PramCompetenceSystem** au niveau de la fonction **startLevel().** Débloque aussi l’accès à l’action FOR et les capteurs.

**F7 :**

Lors de la sélection des niveaux, le système ne sélectionnera que les niveaux ayant été créer pour l’utilisation des conditions IF et IF...ELSE. Cela se passe dans le système **PramCompetenceSystem** au niveau de la fonction **startLevel().** Débloque aussi l’accès aux actions IF et IF...ELSE et les capteurs.

**F8 :**

Rend accessible le bouton " ExecuteButton " qui permet de lancer l’exécution de la séquence rattacher au robot. Si cette fonctionnalité et désactivé, alors le joueur n’aura plus la possibilité de faire exécuté la séquence au robot (utile lors de l’utilisation de fonctionnalité à niveau spécifique comme F12 ou la résolution du niveau par le robot est interdite).

Cela ce passe dans le système **LevelGenerator** au niveau de la fonction **activeDesactiveFunctionality()**.

**F9 :**

Active la possibilité de nommer le robot afin de le rattacher à un séquence. Tant que le robot n’a pas le même nom qu’une des séquences, alors le robot n’exécutera aucune actions présente dans la séquence créer lors de la demande d’exécution de celle-ci.

Dans le componant " AgentEdit " il existe un énumérator " Edit state " qui permet 3 status :

- Locked : Le container de séquence et le robot on le même nom et ne peuvent pas être renommer. On ne peux pas nom plus supprimer le container.

- Editable : Le robot et le container ne comporte pas le même nom et il faut les ronnomer tous les deux avec le même nom afin qu’ils puissent se lier tous les deux. Un fois lier, le changement des noms reste possible mais changer le nom du robot ne fait pas changer celui du container automatiquement et vice-versa. Le container peut être supprimer.

- Synch : Comme Editable sauf que une fois liée si l’on change le nom du robot cela change aussi le nom du container et vice-versa.

Si la fonctionnalité est activé, alors dans le système **levelGenerator** dans la fonction **createEntity** on modifie le " edit state " de l’agent en le passant de Locked (par défaut) à Synch.

F10 :

La librairie, et le drag and drop ne sont plus disponible. Ici l’utilisateur doit sélectionner les actions qui pourrait lui être nécessaire afin de construire une séquence d’action afin de résoudre le niveau. (Il doit en quelque sorte créer lui même les actions disponible dans la librairie).

F11 :

La librairie, le drag and drop et la séquence d’action rattacher au robot ne sont pas disponible. Le robot parcourt le niveau. A la fin de l’exécution on pause des question à l’utilisateur sur ce qu’a effectué le robot.

F12 :

La librairie, et le drag and drop ne sont plus disponible. Ici l’utilisateur doit sélection la séquence d’action parmi une sélection proposé, qu’elle est celle qui permettrai de résoudre le niveau.

F13 :

Fonction de level design avec un élément de décor et une action associé permettant de changer la couleur du robot. Une fois changer le robot peux passer un obstacle. Cette fonction est défini directement au sein de la construction du niveau.

F14 :

Active un onglet supplémentaire permettant de glisser/déposer des éléments représentant des séquences d’action entière.

F15 :

Après la création du niveau, une partie de la séquence du robot est créer mais pas entièrement. Le joueur doit alors compléter la séquence (début ou fin).

F16 :

On propose au joueur une solution non optimal selon un critère donné. Le joueur doit alors modifié la séquence pour améliorer la séquence.

F17 :

Après la création du niveau on propose au joueur une séquence d’actions erroné pour le robot. Le joueur doit alors la modifier pour atteindre l’objectif.

**F18 :**

Lors de la sélection des niveaux, le système ne sélectionnera que les niveaux ayant été créer pour l’utilisation des boucles WHILE. Cela se passe dans le système **PramCompetenceSystem** au niveau de la fonction **startLevel().** Débloque aussi l’accès à l’ action WHILE et les capteurs.

**F19 :**

Lors de la sélection des niveaux, le système ne sélectionnera que les niveaux ayant été créer pour l’utilisation des boucles WHILE et des conditions IF et IF...ELSE. Cela se passe dans le système **PramCompetenceSystem** au niveau de la fonction **startLevel().**

Débloque l’utilisation des opérateurs AND, OR et NOT.

F20 :

La librairie et le drag and drop ne sont plus disponible. La séquence du robot est déjà prés construite et il existe plusieurs sortie pour le niveau. Le joueur doit alors sélectionner la sortie que prendra le robot si celui-ci exécute la séquence qui lui est attaché.

F21 :

Le niveau doit contenir plusieurs sorties. Doit aussi contenir un critère qui forcera le joueur à identifier la bonne sortie et créer une séquence pour le robot en conséquence.

F22 :

Active l’icône permettant de sauvegarder une séquence créer.

F23 :

Fait apparaître une console permettant de voir (et d’interagir directement dedans) les séquences créées en texte sous forme python.

F24 :

Après la création du niveau on propose au joueur une séquence d’actions erroné (un seul block de faux) pour le robot. Le joueur doit alors sélectionner le mauvais block.

**Structure fichier XML :**

**Information du niveau :**

Ici se trouve les différentes informations du niveau. Dans les balises <comp name="C11" /> se trouve les noms des compétences présentes/activées dans le niveau.

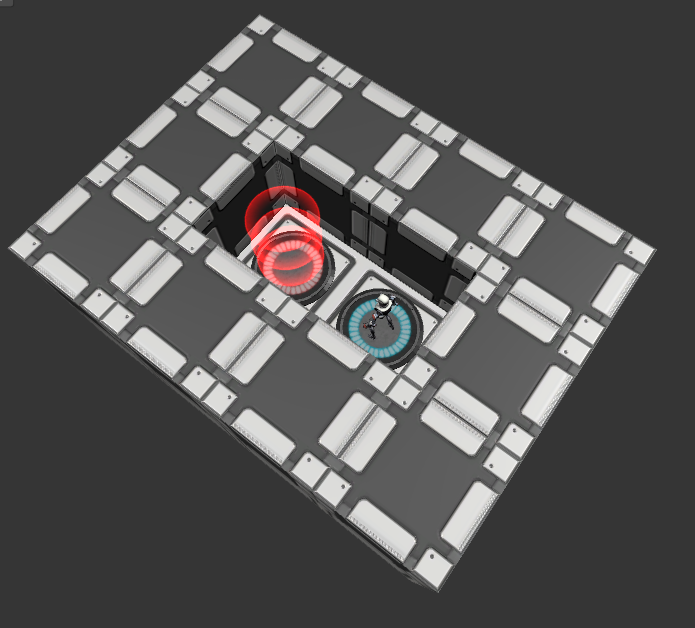
**Création du niveau :**

Le niveau se construise entre les balises <map></map>. Ils doivent se construire ligne par ligne grâce au balise <line></line>. Pour chaque ligne de la map on doit faire apparaître le nombre de bloc nécessaire pour ça création on définissant leurs valeurs pour définir qu'elle sera la type de bloc : -1 pour une case vide ; 0 pour un case avec une sol ; 1 pour une case avec un mur ; 2 pour une case avec le téléporteur de départ ; 3 pour une case avec le téléporteur d'arrivé.

# Exemple de niveau

<map>

<line>

 <row value="1"/>

<row value="1"/>

<row value="1"/>

</line>

<line>

<row value="1"/>

<row value="3"/>

<row value="1"/>

</line>

<line>

<row value="1"/>

<row value="2"/>

<row value="1"/>

</line>

<line>

<row value="1"/>

<row value="1"/>

<row value="1"/>

</line>

<map/>

**Débloquer les bons blocs :**

Les blocs que le joueur pourra drag and drop pour le niveau doivent être renseigné. Tous les blocs et leur limite doivent être renseigné entre les balises <actionBlocLimit><actionBlocLimit/>. Il y a trois limites possible pour les blocs : -1 signifie que le bloc peut être utilisé de façon illimité ; 0 signifie que le bloc ne sera pas présent sur la scène (et donc inutilisable) ; 1 ou plus signifie que le bloc peut être utilisé un nombre de fois égal à la limite.

# Exemple de limite

<actionBlocLimit>

<limit actionType="Forward" limit="-1"/>

<limit actionType="If" limit="1"/>

<limit actionType="TurnBack" limit="3"/>

<limit actionType="Wall" limit="0"/>

<actionBlocLimit/>

**Création robot et sentinelle :**

Les robots et sentinelles doivent être créés en renseignant leurs positions dans le niveau avec PosX et PosZ ainsi que leur direction avec l'attribut direction. Les valeur de direction sont comprise entre "0" et "3" avec : 0 pour l'Est ; 1 pour l'Ouest ; 2 pour le Nord ; 3 pour le Sud.

Pour la sentinelle il faut aussi ajouter sont range, sont selfRange et sont typeRange. Le range range correspond à la portée (nombre de case devant lui) de sa zone de détéction.

LE RESTE VOIR AVEC MATHIEU

Pour chacun des entités un script peux lui est être attaché (voir après pour la construction d'un script). Pour cela il suffit d'écrire le script au sein des balise de l'entité.

# Exemple Robot

<player posX="1" posZ="3" direction="1" />

# Exemple Sentinelle

<enemy posX="1" posZ="3" direction="1" range="2" selfRange="False" typeRange="0">

<script>

…

<script />

<enemy />

**Ecrire un script :**

Un script pour un agent s'écrit toujours entre les balise <script><script/>. Les éléments pouvant être utilisés pour le script sont les suivants.

If :

# Début de structure du If

< control type="If">

<condition>

# Ajouter ici tous les éléments ou opérateur pour condition

# Exemple

<action type="Wall" />

</condition>

<container type="ThenContainer">

# Ajout ici n'importe qu'elle action ou structure d'action

# Exemple

<action type="Forward" />

</container>

</control>

IfElse :

# Début de structure du IfElse

<control type="IfElse">

<condition>

# Ajouter ici tous les éléments ou opérateur pour condition

# Exemple

<captor type="Wall" />

</condition>

<container type="ThenContainer">

# Ajout ici n'importe qu'elle action ou structure d'action

# Exemple

<action type="Forward" />

</container>

<container type="ThenContainer">

# Ajout ici n'importe qu'elle action ou structure d'action

# Exemple

<action type="TurnBack" />

</container>

</control>

While :

# Début de structure du While

< control type="While">

< condition>

# Ajouter ici tous les éléments ou opérateur pour condition

# Exemple

<action type="Wall" />

</condition>

<container type="ThenContainer">

# Ajout ici n'importe qu'elle action ou structure d'action

# Exemple

<action type="Forward" />

</container>

</control>

For :

# Début de structure du For

<control type="For" nbFor="X"> # Xnombre de fois ou la boucle doit être exécutée

# Ajout ici n'importe qu'elle action ou structure d'action

# Exemple

<action type="TurnBack" />

</control>

Forever :

# Début de structure du For

<control type="Forever">

# Ajout ici n'importe qu'elle action ou structure d'action

# Exemple

<action type="TurnBack" />

</control>

NotOperator :

# Début de structure du NotOperator

<operator type="NotOperator">

# Ajouter ici tous les éléments ou opérateur pour condition

# Exemple

<captor type="Wall" />

</operator>

AndOperator :

# Début de structure du AndOperator

< operator type="AndOperator">

<condition type="Left">

# Ajouter ici tous les éléments ou opérateur pour condition

# Exemple

< captor type="Wall" />

</condition>

< condition type="Right">

# Ajouter ici tous les éléments ou opérateur pour condition

# Exemple

< captor type="RedArea" />

</condition>

</operator>

OrOperator :

# Début de structure du OrOperator

<operator type="OrOperator">

<condition type="Left">

# Ajouter ici tous les éléments ou opérateur pour condition

# Exemple

<captor type="Wall" />

</condition>

< condition type="Right">

# Ajouter ici tous les éléments ou opérateur pour condition

# Exemple

< captor type="RedArea" />

</condition>

</operator>

Elément de condition:

<captor type="Wall" /> # Pour une mur

<captor type="FielGate" /> # Pour un champ de force

<captor type="Terminal" /> # Pour un terminal

<captor type="Enemie" /> # Pour une sentinelle

<captor type="RedArea" /> # Pour une zone de détection

<captor type="Exit" /> # Pour une zone de détection

Autre action simple (sans structure):

<action type="Forward" /> # Le robot avance d'une case

<action type="TurnLeft" /> # Le robot pivote sur la gauche

<action type="TurnRight" /> # Le robot pivote sur la droite

<action type="TurnBack" /> # Le robot effectue une rotation à 180°

<action type="Active" /> # Le robot active un élément activable sur sa case

<action type="Wait" /> # Le robot ne fait rien pendant un pas de temp