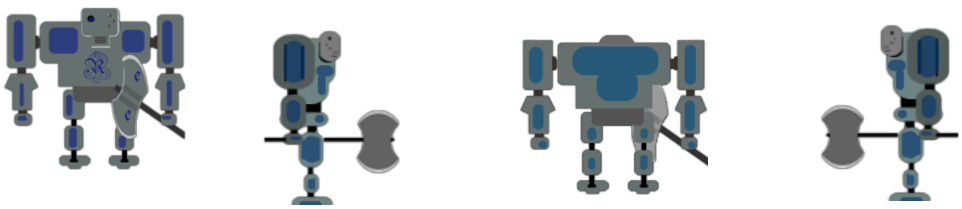
Manuel de l'utilisateur



Juin 2012

I | Installation

a) Comment installer ?

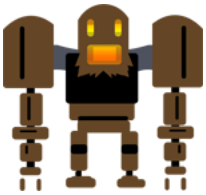
Pour installer Marre des Robots®, il faut télécharger le jeu sous forme d'un fichier jar. Il est alors possible de le lancer par la commande java -jar marreDesRobot.jar

b) Personnaliser le jeu

Dans Marre des Robots®, il est possible de personnaliser le jeu afin d'afficher ses propres images pour représenter ses robots. Pour ajouter un nouvel habillage de robot, il faut suivre les trois étapes suivantes:

1. Il faut déposer dans le dossier *./src/sim/tricycle/ihm/images/robots/* l'image principale du robot. C'est à dire celle qui apparait dans le panneau de sélection des robots.





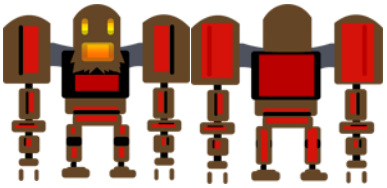
./src/sim/tricycle/ihm/images/robots/ Destructeur.png

2. Il faut ajouter pour chaque couleur d'équipe la lettre 'B' pour l'équipe bleu, 'N' pour l'équipe noire, 'J' pour l'équipe jaune et 'R' pour l'équipe rouge dans le dossier aCharger.

DestructeurR.png

3. Il faut ensuite charger les différentes images selon les positions des robots dans le dossier *./src/sim/tricycle/ihm/images/aCharger/*. Pour que l'affichage soit correct il faut respecter un standard dans les noms des images :
   - L'image de face est considérée comme l'image principale.
   - L'image de dos doit avoir comme suffixe à l'image principale : "Dos".
   - Pour l'image tournée sur la droite: "D".
   - Pour l'image tournée sur la gauche: "G".





./src/sim/tricycle/ihm/images/ACharger/ DestructeurR.png DestructeurDosR.png

II | Utilisation

a) Ecriture des automates

Dans cette partie, nous allons vous décrire les différentes étapes qui vous permettront de créer votre propre automate afin que le logiciel MdR soit à même de le prendre comme tel (c'est-à-dire comme un automate et non comme un texte normal)

Au cours de ce tutoriel, à titre d'exemple nous écrirons petit à petit l'automate qui permettra à un robot de ramasser des pièces.

Etape 1 : Création du fichier

S'il n'existe pas de fichier automate.txt, vous devez donc commencer par le créer. Vérifiez bien que le fichier soit un fichier texte et que vous n'avez pas fait de faute de frappe ou laissé des espaces en fin de mot (si vous êtes sous linux, il ne criera pas au scandale mais sous windows, c'est une autre paire de manche). Cette vérification est primordiale car dans le cas où vous vous êtes trompé, le logiciel MdR ne pourra pas lire votre fichier.

Si le fichier existe déjà vous pouvez passer à l'étape numéro 3.

Etape 2 : Pour bien commencer/finir...

Toujours en supposant que vous venez de créer le fichier, vous devez maintenant ouvrir le dit fichier. La première chose à faire consiste à écrire sur la première ligne, la balise : /*fichier automate*/ .

Cette balise permet au logiciel de comprendre que ce que vous allez écrire ensuite sera un automate.

Ensuite sautez une ou plusieurs lignes et écrivez : fin .

Ceci indique que votre automate est fini et que le logiciel peut s'arrêter de lire votre fichier. Aussi, il devient évident que ce mot est un mot interdit. Vous ne devez en aucun cas l'utiliser pour décrire votre automate.

Etape 3 : la première accolade

Vous allez maintenant pouvoir décrire votre automate.

La première chose à faire est de donner le nom de votre automate. Par exemple robot_porteur. Ensuite ouvrez une accolade : { pour dire que tout ce qui suivra appartiendra au robot_porteur.

Pour éviter de vous embrouiller dans les accolades ouvrantes et fermantes (il y en aura beaucoup par la suite), il est préférable de sauter une ligne et de fermer l'accolade dès à présent : }

A partir d'ici votre fichier devrait ressembler à ceci :

```
/*fichier automate*/
robot_porteur {

}

fin
```

Etape 4 : les états

Nous allons maintenant écrire les états que comportera votre automate.
Pour cela écrivez tout d'abord : etat= . *Faites attention à ne pas mettre d'accent sur le « e »*. Ecrivez ensuite le nom de votre état .

Les Tags : *Vous devez savoir que les tags permettent d'établir combien, en terme de coup d'horloge, prendront les actions de bas niveau tel que avancer, tourner etc.*

Après avoir écris le nom de votre état, ouvrez un crochet: [et écrivez le nom de votre tag. Par exemple tag1.
Puis fermez le crochet et ouvrez une accolade :]{. Comme d'habitude, sautez quelques lignes et fermez l'accolade.

Etat spécifique: vous devez savoir que le premier état de tous vos automates devra être un état d'initialisation appelé init.
Il ne comportera pas de tag. C'est un état transitoire qui permet à l'automate d'aller sans transition à l'état le plus stable que vous choisirez.
De ce fait vous ne devrez en aucun cas utiliser le mot init pour appelez un de vos états par la suite.

Après cette étape vous devriez avoir quelque chose comme cela :
(plusieurs états ont été écris et seront utiles par la suite de ce tutoriel)

```
/*fichier automate*/
robot_porteur {
        etat=init{
        }
        etat=normal[tag1]{
        }
        etat= deplacement_normtag1]{
        }
        etat=porteur[tag2]{
        }
        etat=deplacement porteur[tag1]{
        }
```

}

fin

Etape 5 : les transitions

Cette étape est très simple. En effet, à chaque fois que vous voulez rajouter une transition qui vous permettrai d'aller d'un état à un autre, il vous suffira d'écrire : transition{}.

Avant de fermer l'accolade vous devrez mettre l'état dans lequel l'automate va se trouver après avoir parcouru la transition. La syntaxe est la même que pour l'étape précédente, si ce n'est que vous n'avez pas à mettre les tags et que vous devez finir le nom de l'état par un point virgule (car ceci est considérer comme une action comme vous pourrez le comprendre par la suite)

Etape 6 : Les actions

Jusqu'à présent, nous avons suivit l'ordre logique d'écriture de l'automate. Cependant pour que vous compreniez mieux comment vous écrirez les conditions, nous allons tout d'abord parler des actions qui viennent bien après les conditions dans l'ordre d'écriture.

Les actions sont exécutées après vérification de la condition.

Il faut savoir qu'une action comporte un nom, des paramètres et se termine par un point virgule.

Les paramètres : sont mis entre parenthèses. Par conséquent si l'action n'en comporte aucun, vous devrez mettre des parenthèses vides.
Vous pouvez également typer les paramètres. Pour cela, il suffit d'écrire le nom du type, de mettre deux points et d'écrire le nom du paramètre. Le type par défaut est le type string. S'il y a plusieurs paramètres, ils seront séparés par une virgule.

Exemple :

- une action appelée *ramasser* n'ayant aucun paramètre s'écrira : ramasser() ;
- une action appelée *allerici* ayant un seul paramètre, avec un type *var* s'écrira : allerici(var : piece);
- une action appelée *prendre* ayant deux paramètres, avec un type *ref* et un type *string* s'écrira : prendre(ref : team.base, tour) ; ou encore s'écrira : prendre(ref : team.base, string : tour) ;

Etape 7 : les affectations

Pour écrire une affectation il n'y a rien de plus simple. Vous écrivez le nom qui sera donc un nom de variable, puis vous mettez une flèche dirigée vers la dites variable et vous mettez une action.

Exemple :

nom de la variable : *cible*
action : *casealea* pris sans paramètre
Cela s'écrit donc : cible<-casealea();

Etape 8 : les conditions

les conditions sont ce qui permet de décrire une transition et donc ce qui détermine le passage d'un état à un autre en fonction des événements extérieur.

-Une transition ne peut comporter qu'une seule condition.
-Etant donné que ce sont des conditions de haut niveau, vous ne pouvez pas écrire des conditions multiples. (C'est à dire qui comporterait des opérateurs booléens : par exemple : ou, et). Vous pouvez cependant utiliser l'opérateur booléen *non* (modéliser par le point d'exclamation : !)
Les conditions ont un ordre de priorité qui est celui de l'ordre de lecture, c'est-à-dire de haut en bas et de gauche à droite.

Après ces petits préliminaires, parlons maintenant de l'écriture d'une condition :
Vous devez commencer par écrire le mot *si* sans majuscule. Puis ouvrez une parenthèse.
Ceci étant fait, vous devez maintenant écrire la condition qui a la même syntaxe que celle d'une action moins le point virgule à la fin.
Si vous souhaitez mettre le point d'exclamation, mettez le avant ou après la condition.
Une fois que la condition est écrite, vous n'avez qu'à fermer la parenthèse et à ouvrir une accolade.

Exemple : si(contains(caseCourante,cible)){ }

Condition spéciale :

Il existe la condition spéciale : toujours vrai. La syntaxe est la suivante : sinon{ }

Exemple : sinon{ }

Arrivez à la fin de cette étape vous devriez avoir quelque chose qui ressemble à ceci :

```
/*fichier automate*/
robot_porteur{
```

```
etat=init{
transition{
etat=normal;
}
}
        etat=normal[tag1,tag2]{
transition{
                si(piece_trouvee()){ piece<-pieceplusproche();
etat=deplacement_norm;
}
}
transition{
                sinon{sleep();
                 etat=normal;
                }
}
}

        etat=deplacement_norm[tag1]{
transition{
                si(contains(ref:self.case,var:piece)){ ramasser();
                                etat=porteur;
}
}
transition{
                sinon{allerici(var : piece);
                 etat=normal;
                }
}
        }
        etat=porteur[tag2]{
transition{
                etat=deplacement_port;
}
        }
        etat=deplacement_port[tag2]{
transition{
                si(case==base){deposer();
                                etat=normal;
                }
}
transition{
                sinon{allerici(ref :team.base);
                 etat=deplacement_port;
                }
}
        }
}
fin
```

Etape 9 : les tags

Ceci constituent la dernière étape de l'écriture de l'automate.

Les tags comportent une descriptions des coûts des actions de bas niveaux.

Comme pour les états, vous devez tout d'abord écrire tag= puis le nom de votre tag et ouvrir une accolade.

Pour décrire vos actions de bas niveaux, il suffit :

- d'écrire le nom de votre action
- de mettre un égale
- de donner le coût de votre action en chiffre
- de terminer par un point virgule

Ce qui vous donnera par exemple dans notre exemple de robot_porteur ceci :

```
tag=tag1{
avancer=1;
reculer=1;
tourner_droite=1;
tourner_gauche=1;
}
tag=tag2{
avancer=2;
reculer=2;
tourner_droite=1;
tourner_gauche=1;
}
```

Vous obtenez finalement le fichier suivant :

```
/*fichier automate*/
robot_porteur{
etat=init{
transition{
etat=normal;
}
}
        etat=normal[tag1,tag2]{
transition{
                si(piece_trouvee()){ piece<-pieceplusproche();
etat=deplacement_norm;
}
}
transition{
                sinon{sleep();
                 etat=normal;
                }
}
}

        etat=deplacement_norm[tag1]{
transition{
                si(contains(ref:self.case,var:piece)){ ramasser();
```

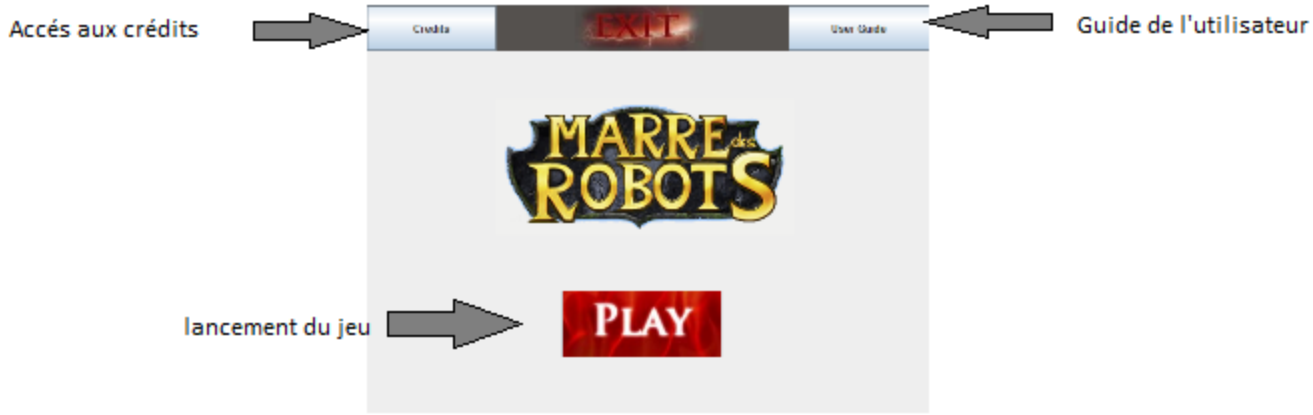
```
                                        etat=porteur;
}
}
transition{
                    sinon{allerici(var : piece);
                     etat=normal;
                    }
}
          }
          etat=porteur[tag2]{
transition{
                    etat=deplacement_port;
}
          }
          etat=deplacement_port[tag2]{
transition{
                    si(case==base){deposer();
                                        etat=normal;
                    }
}
transition{
                    sinon{allerici(ref :team.base);
                     etat=deplacement_port;
                    }
}
          }

tag=tag1{
avancer=1;
reculer=1;
tourner_droite=1;
tourner_gauche=1;
}
tag=tag2{
avancer=2;
reculer=2;
tourner_droite=1;
tourner_gauche=1;
}
}
fin
```

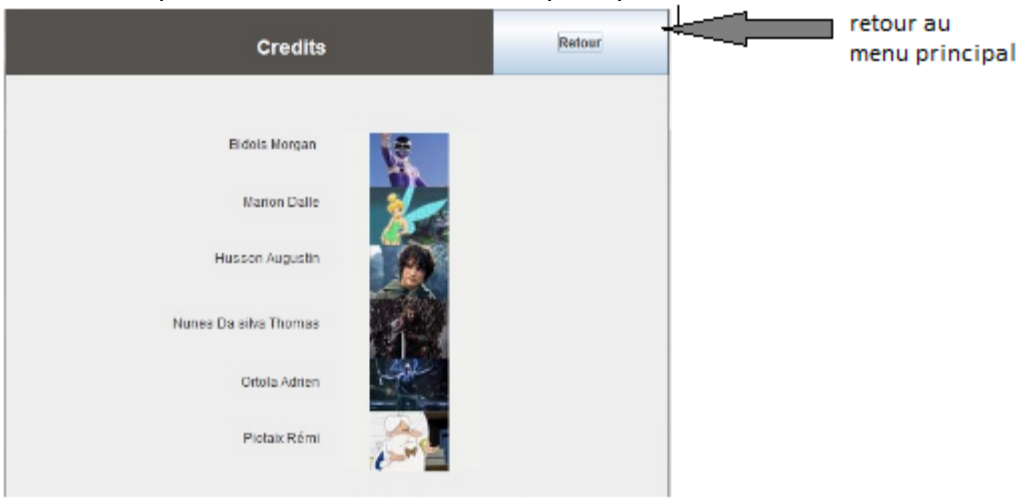
b) Création d’une partie.

Menu principal

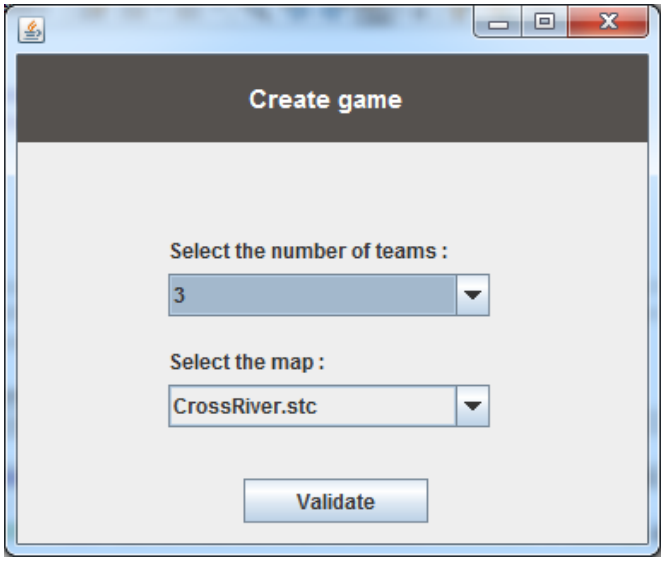
La création d’une partie de de Marre des Robots® se fait en plusieurs étapes. Lors du lancement d’une partie on accède au menu principal. A partir de ce menu, il est possible d’accèdé à la page des crédits, afficher sous format pdf ce manuel ou de lancer une partie.



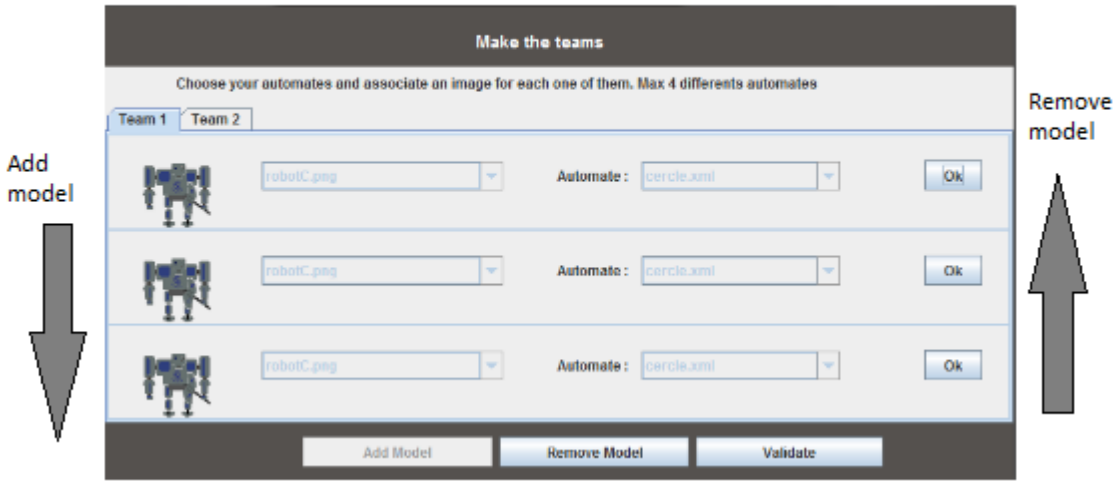
A partir des crédit on peut ensuite revenir au menu principal.



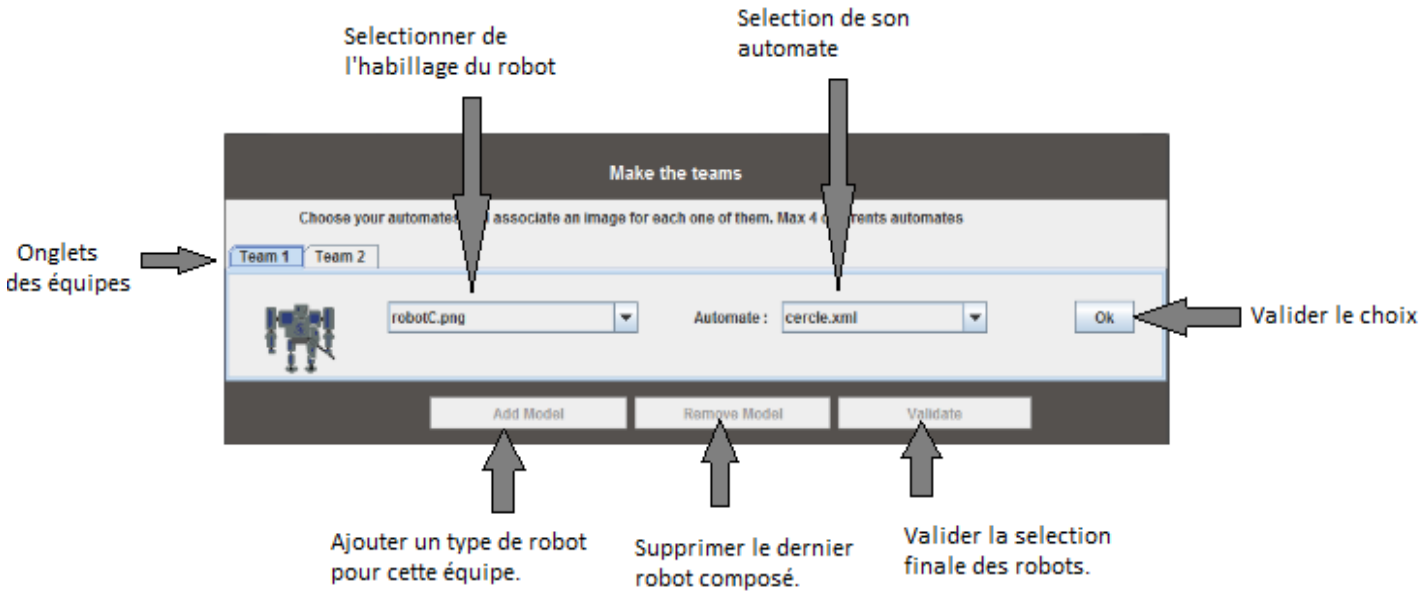
En lançant une partie en appuyant sur PLAY, Une nouvelle fenêtre s’affiche permettant de sélectionner le nombre d’équipes (2 à 4) ainsi que la carte choisie.



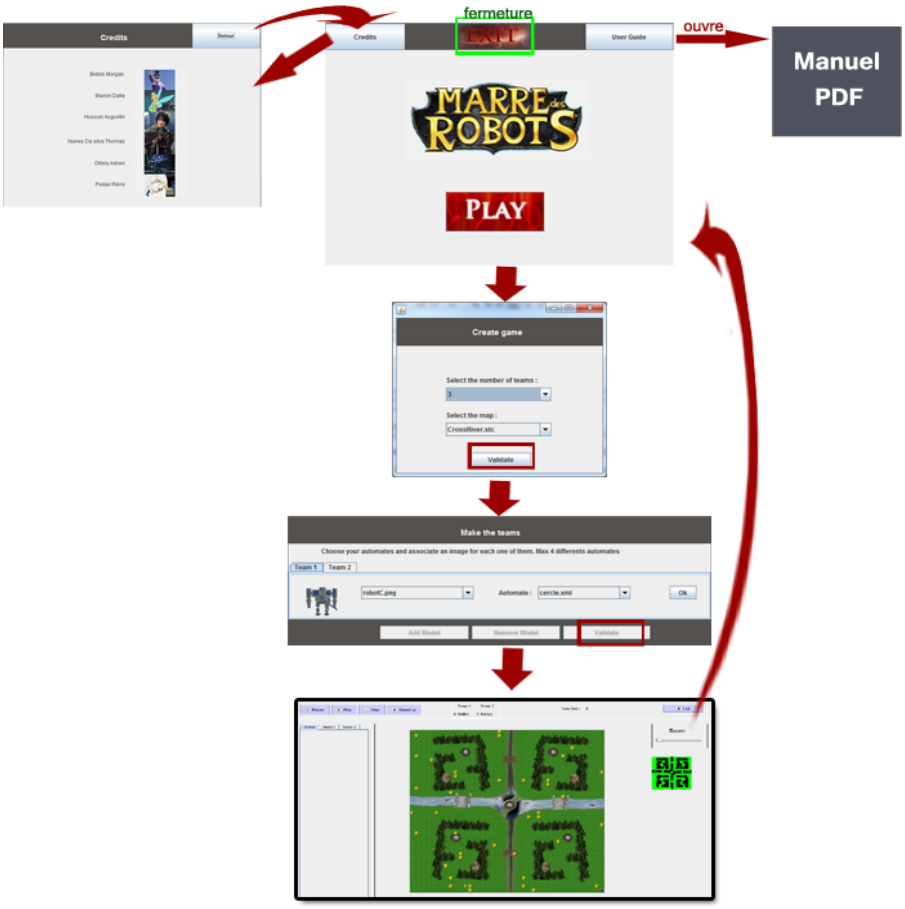
En validant ce choix, on accède à la composition des équipes. Le nombre d'onglet correspond au nombre d'équipes choisies dans la fenêtre précédente. Pour chaque équipe il faut sélectionner les différents robots de chaque équipe. **Add Model** permet d'ajouter un type de robot pour l'équipe en cours de sélection. **Remove Model** supprime le dernier type de robot sélectionné.



Pour chaque robot il faut lui choisir un habillage et un automate puis valider en cliquant sur **OK**. Une fois que toutes les équipes possèdent au moins un type de robot validé, il est possible de lancer le jeu en cliquant sur **Validate**.



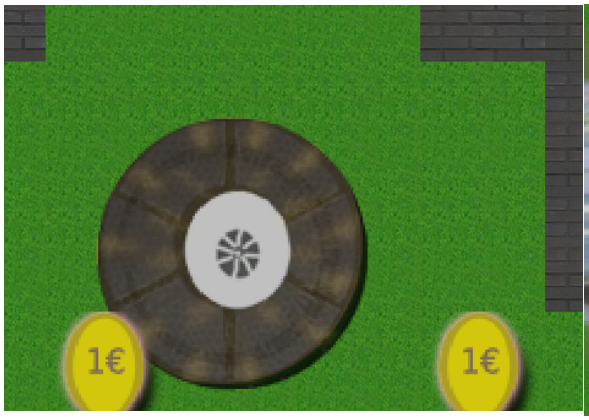
Une fois validé, le jeu se lance. Un bouton EXIT permet de revenir au menu principal et permet de relancer un jeu. Le diagramme suivant résume les différents liens existant entre les fenêtres:



c| Régles du jeu

Chaque équipe possède une base d'où apparaissent leurs robots. Les robots coûtent des pièces d'or récupérablent si elles sont ramenés à la base. L'or permet d'acheter de nouveaux robots.

Pour gagner une partie il faut collecter un maximum de boule, celle n'apparaissent que si un point de contrôle est capturé. En effet ces points de controles commencent la partie dans un état neutre, si des robots sont présent sur celles ci, ils sont capturés et commencent à générer des boules de manière régulière.



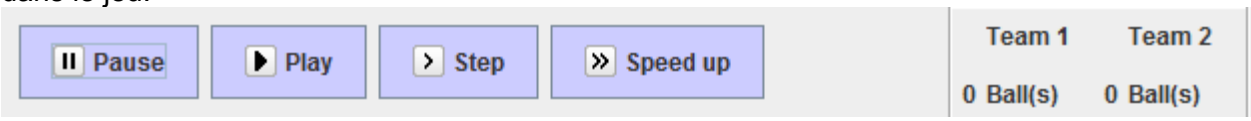
Point de contrôle neutre *Point contrôlé par l'équipe bleu*

Lorsqu'ils sont capturés ces points de controle change de couleur ils passent de gris à la couleur de l'équipe qui vient de capturer le point. En cas de combat sur ce point, celui ci ne peut être capturé.

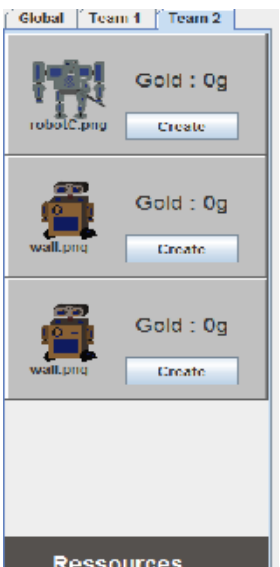
d| Interface du jeu

Gestion du temps

L'interface permet de gérer le temps, il est possible de mettre sur pause et de relancer le jeu, de l'accéléré **SPEED UP** ou encore de lui faire éxécuter des action pas à pas grace à **STEP.** L'affichage du nombre de balle par équipe s'adapte en fonction du nombre d'équipe présente dans le jeu.



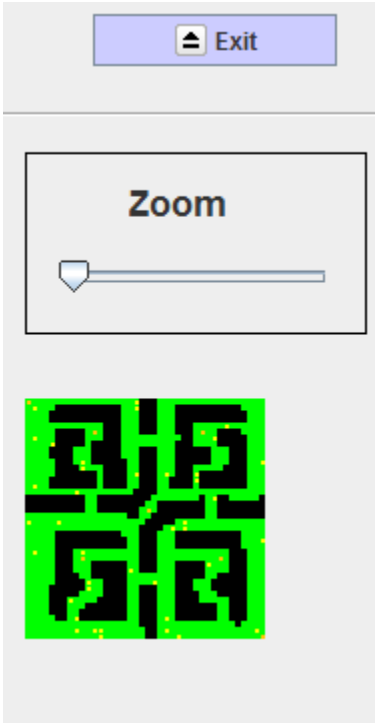
Système d'onglets



Des onglets permettent de naviguer entre les cartes des équipes et la carte objective. A partir de ses onglets on peut connaitre les différents types de robots possibles ainsi que leurs prix.

Les onglets indiquent également les ressources dont disposent chaque équipe.

Zoom et mini carte



Le Jeu dispose d'un zoom sur la carte qui peut être controlé par un bouton ou avec la molette de la souris. En cliquant sur la carte, l'utilisateur peut alors la déplacer. Pour accéder à des zones éloignées, la mini carte permet non seulement d'avoir une vue globale du jeu mais aussi de recentrer la carte sur un point précis.

La mini carte se désactive si la carte est entierement visible dans la fenêtre du jeu.