Implémentation d'un Bot Starcraft 2 "Rush Roaches"

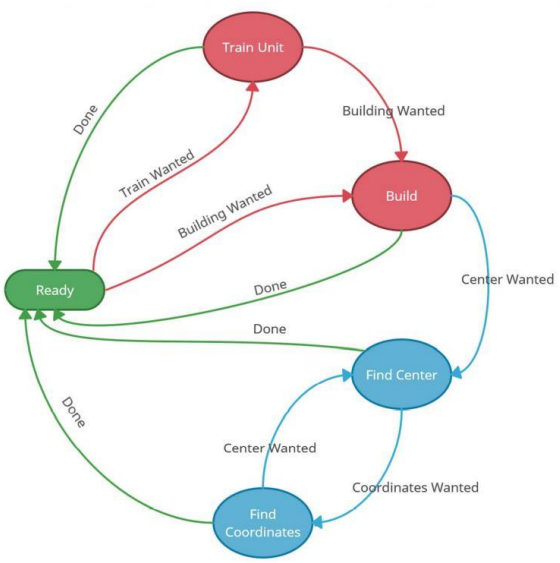
Par Emmanuel Mompi et Théo Montaigu

1/ Notre base d'implémentation :

La mise en place du bot s'est engagée à travers la réalisation d'un premier TP sur PySc2. L'objectif de ce dernier était la construction de n'importe quel bâtiment et unité de notre race (à savoir "Zerg"). De ce fait, nous souhaitions mettre en place une base d'implémentation qui nous serais potentiellement utile pour le projet. La logique de cette base étant de ne pas coder à la main les positions des bâtiments ainsi que d'avoir une gestion des dépendances des unités et bâtiments.

Pour trouver des coordonnées valides de construction de bâtiments nous procédons en plusieurs étapes (celles-ci peuvent se visualiser grâce au diagramme ci-dessous). Nous allons centrer la caméra sur chacune des Hatchery, tour à tour, afin de trouver un espace valide de construction. L'analyse du terrain à travers la combinaison de 3 matrices. La matrice "obs.observation.feature_screen.buildable" nous indique si le terrain est plat, la matrice "obs.observation.feature_screen.creep" nous indique si le terrain est recouvert de creep et la matrice "obs.observation.feature_screen.unit_type nous indique quelle(s) unité(s) sont présente sur le terrain. Cette espace doit donc être plat, avoir du creep et aucune unité présente dessus. Autour des Hatcheries nous cherchons donc un carré ayant ces propriétés et de largeur égale à "largeur du bâtiment + 2" (ceci afin d'avoir une bordure et de ne jamais bloquer nos unités). Si après avoir cherché autour de chacune de nos Hatchery nous ne trouvons pas d'emplacement valide, nous construisons une nouvelle Hatchery.

Pour entrainer une unité ou construire un bâtiment nous utilisons un arbre de dépendance. Celui-ci est un dictionnaire de dictionnaire contenu dans le fichier "updatedtree.py". Il nous indique les dépendances directes et indirectes des unités ainsi que d'autres informations utiles comme la taille du batiment, l'unité nécessaire au morph, les ressources nécessaires ect. Si nous donnons l'ordre d'entrainer ou de construire mais qu'une dépendance directe manque, alors l'ordre de construction de cette dernière est automatiquement lancé. Si par exemple l'ennemi détruit notre Roach Warren et notre Spawning Pool, l'ordre d'entrainer un Roach aura pour conséquence de construire ces bâtiments manquants.



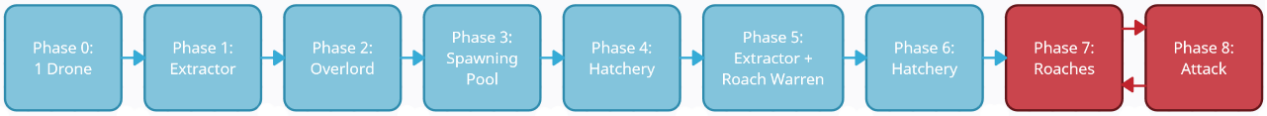
2/ Le BO “Rush Roaches”

Notre BO est un “Rush Roaches” propre à la map “Simple 64” et organisé par nous-même. Il repose sur 3 Hatcheries. Nous ne récoltons que sur une seule, les deux autres servants à la production de larves nécessaires à l’entrainement des Roaches. Avec cette configuration nous avons approximativement un équilibre entre récolte des ressources et production d’unités. Notre bot utilise deux automates à états finis. Le premier, décrit ci-dessus, organisant les étapes nécessaires à la création d’une unité ou d’un bâtiment. Le second, décrit ci-dessous, permettant de progresser dans notre BO. Le premier est donc utilisé au sein du second. L’utilisation de plusieurs FSM nous a permis de différencier de façon claire des enchainements d’états aux buts spécifiques.

Ci-dessous nous expliquons brievement en quoi consiste chaque phase du BO. Certaines informations de sont pas explicitées. Par exemple le fait que si on manque de récolteurs de gaz on en ajoute, ainsi que le fait que si on manque d’Overlord alors on en entraine. Certaines actions déclenchent un changement de phase et seront donc suivis de la mention “Transistion”.

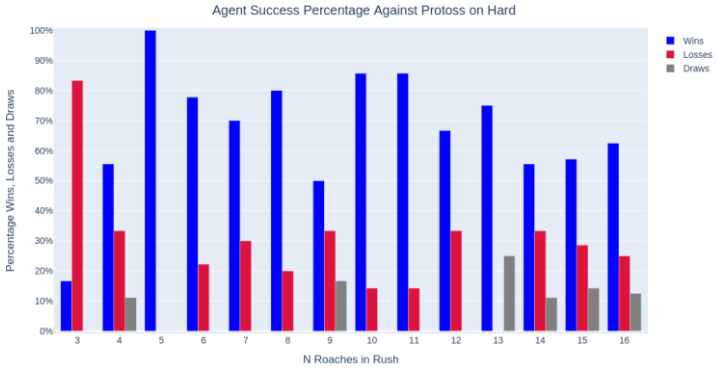
- Phase 0 : Actualisation de certaines informations en fonction de la position de notre bot sur la map + Création d’un drone (Transition).
- Phase 1 : Entrainement de drones jusqu’à en avoir 14 suivis de la construction d’un Extractor (Transition).
- Phase 2 : Quand l’Extractor a fini sa construction on donne l’ordre à 3 drones de récolter sur celui-ci. Tant qu’on a moins de 14 drones on en entraine. Si on possède 14 drones, on a atteint la limite de supply et donc on entraine un Overlord (Transition).
- Phase 3 : Si on a 200 de minerais on construit une Spawning Pool (Transition), sinon on entraine des drones.
- Phase 4 : Si on a 300 de minerais on construit une Hatchery (Transition), sinon on entraine des drones.
- Phase 5 : Si on a 150 de minerais on construit une Roach Warren (Transition), sinon on entraine des drones.
- Phase 6 : Si on a 300 de minerais on construit une Hatchery (Transition), sinon on entraine des drones.
- Phase 7 : Entrainement d’un Roache (Transition).

- Phase 8 : Si on possède N Roaches alors on leur donne l'ordre d'attaquer l'ennemi (Transition). La position d'attaque varie cycliquement en fonction des steps afin d'être sûr d'attaquer les différentes bases de l'adversaire.



3/ Résultats et conclusion

Notre programme contient aussi la récolte de statistiques. Comme expliqué précédemment notre bot ordonne aux Roaches d'attaquer s'il en possède au moins N. Dans l'état actuel du bot nous faisons varier ce N à chaque partie et récoltons les statistiques de Wins/Losses/Draws en fonction du N et en fonction de la race de l'adversaire. Nous souhaitions trouver un N optimal. A priori nous pensions que certaines attaques rapides ou au contraire certaines attaques massives aurait des impacts significatifs sur les résultats de victoire et que peut-être cela aurait été fonction de la race et/ou du niveau de difficulté de l'adversaire. Cette analyse n'a pour l'instant pas été fructueuse mais elle nous donne des résultats comme exposés dans le graphe ci-dessous.



Par ailleurs, nous obtenons des résultats de victoire plutôt convaincants. Ci-dessous un graphe nous montre dans quelles proportions notre bot gagne et ce en fonction du niveau de difficulté de l'adversaire. Les niveaux de victoires sont à peu près proportionnels à la difficulté de l'adversaire.