Plateau.py

Attributs :

self.troupes = [ ]
self.terrain = [ ]

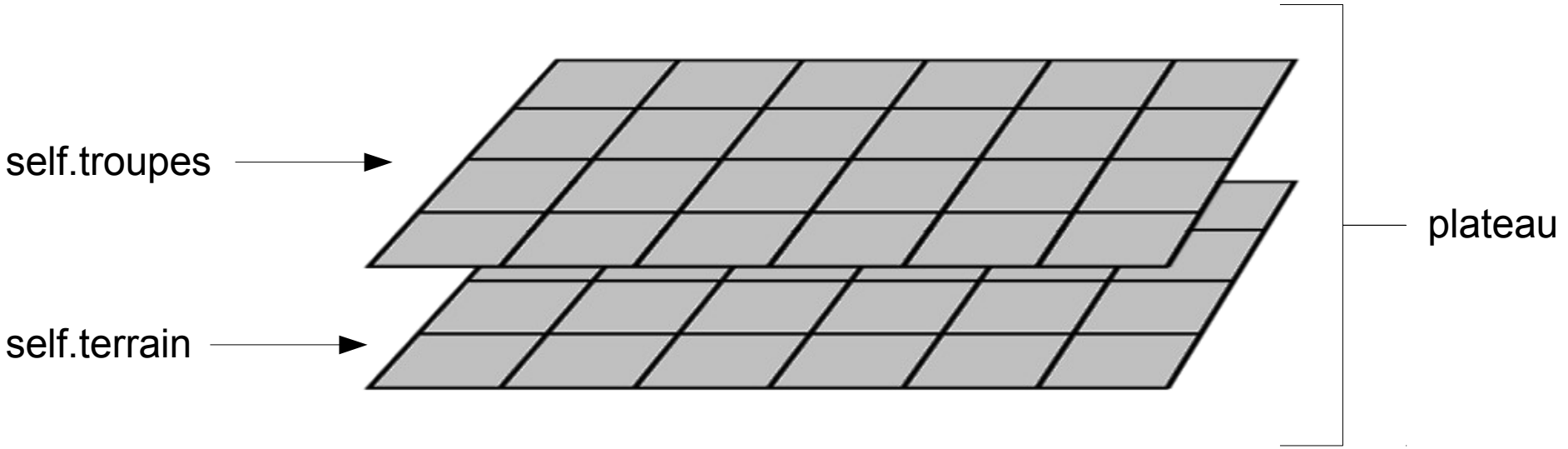
Méthodes :

creerGrille(self,largeur=40,hauteur=40,caseVide=1)
positionnerUnite(self,unite,x,y,caseVide=1)
supprimerUnite(self,x,y,caseVide=1)
genererTerrain(self)
importerTerrain(self,map)
grilleObstacle(self,xd,yd)
deplacement(self,xd,yd,xa,ya)

Plateau.py

Attributs :

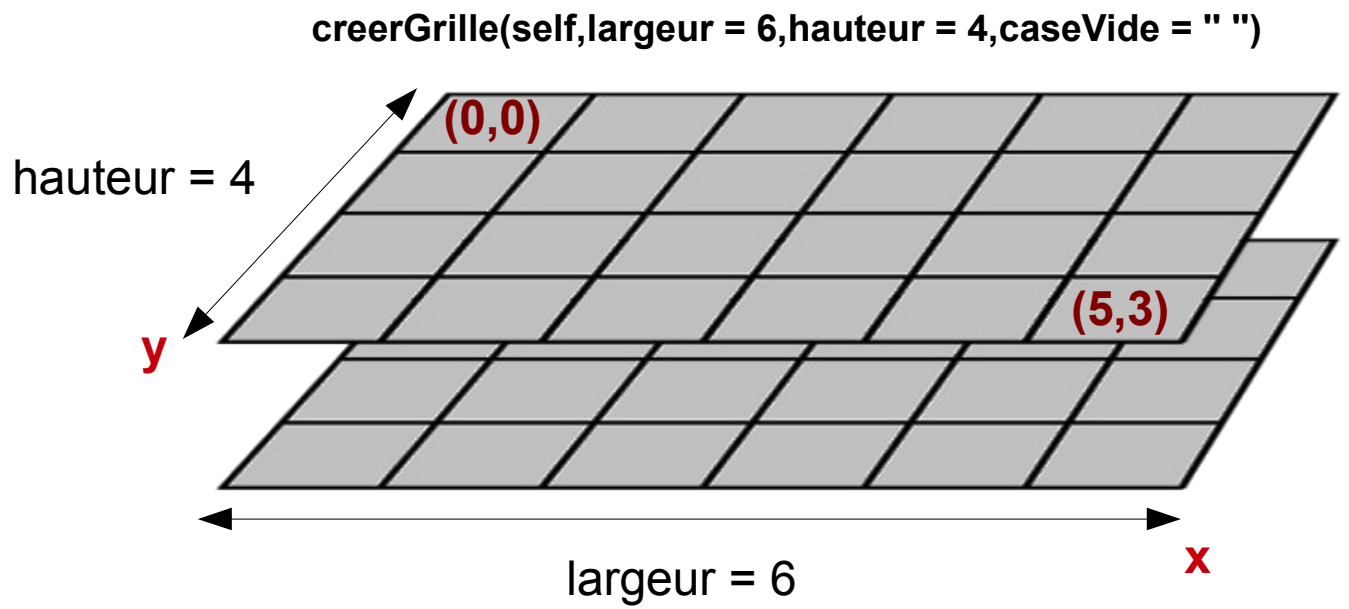
self.troupes = [ ]
self.terrain = [ ]



Plateau.py

Méthodes :

creerGrille(self,largeur=40,hauteur=40,caseVide=1)

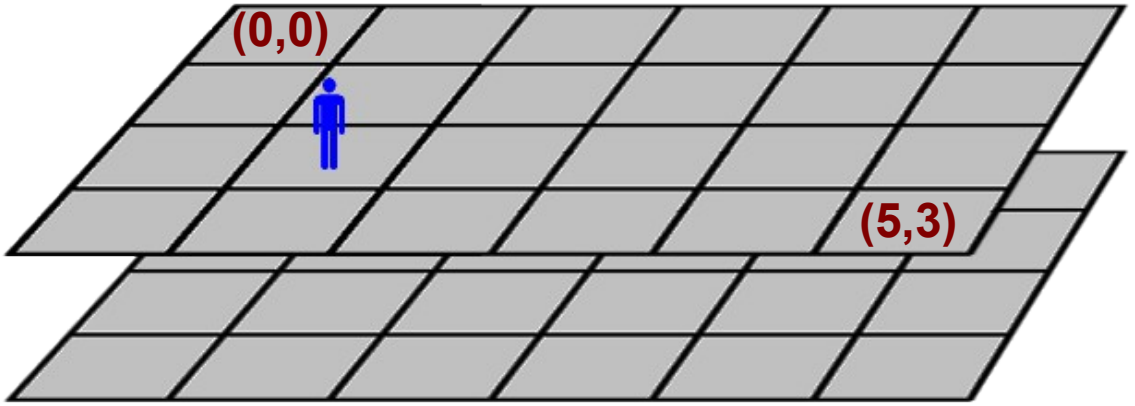


Plateau.py

Méthodes :

positionnerUnite(self,unite,x,y,caseVide=1)

positionnerUnite(self,unite*,1,2,caseVide = " ")



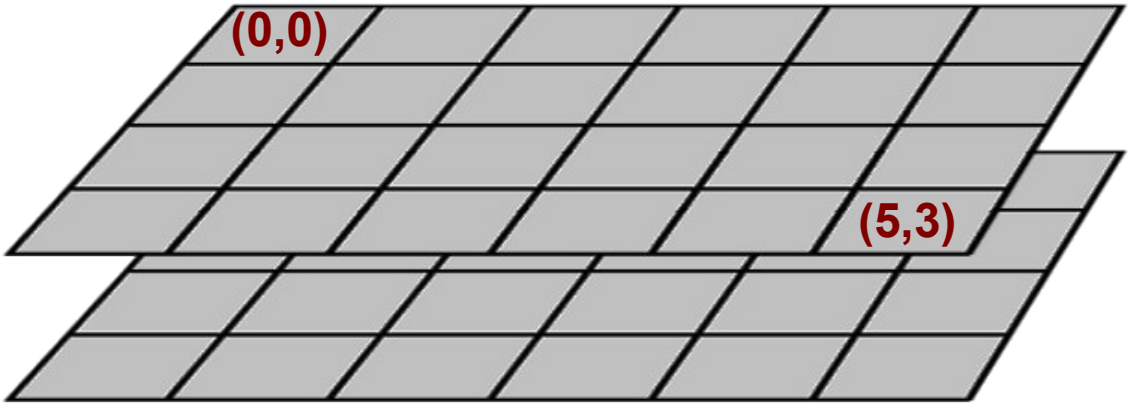
*unite est un objet correspondant à une unité/troupe

Plateau.py

Méthodes :

supprimerUnite(self,x,y,caseVide=1)

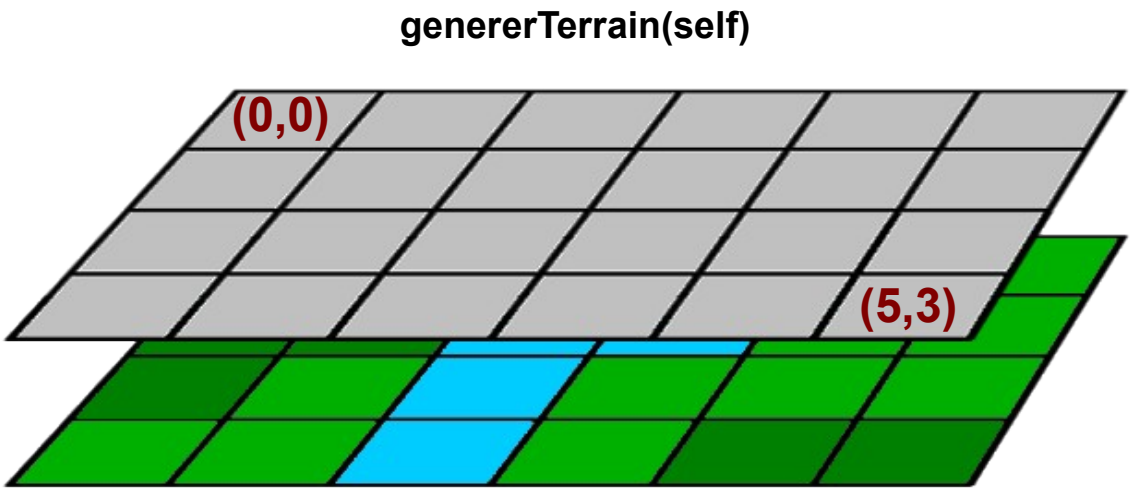
supprimerUnite(self,1,2,caseVide = " ")



Plateau.py

Méthodes :

genererTerrain(self)



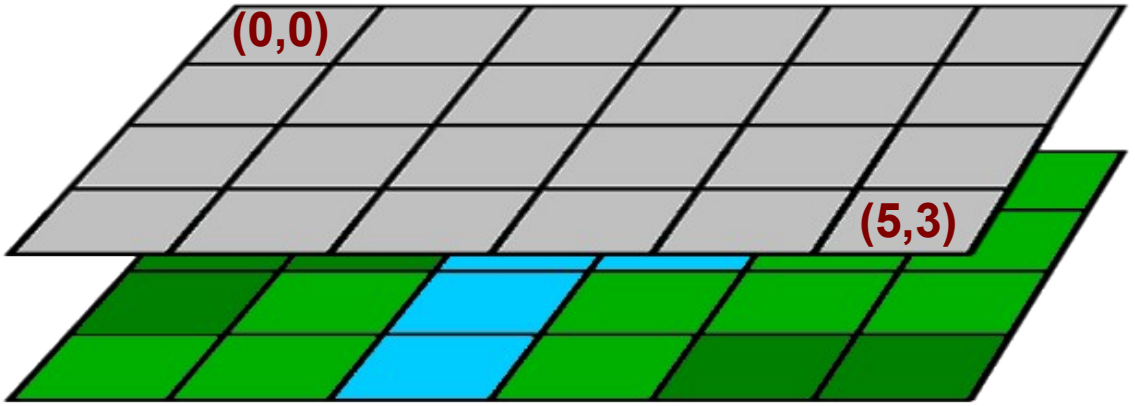
Voir Algorithme Perlin

Plateau.py

Méthodes :

importerTerrain(self,map)

importTerrain(self,"defaut")



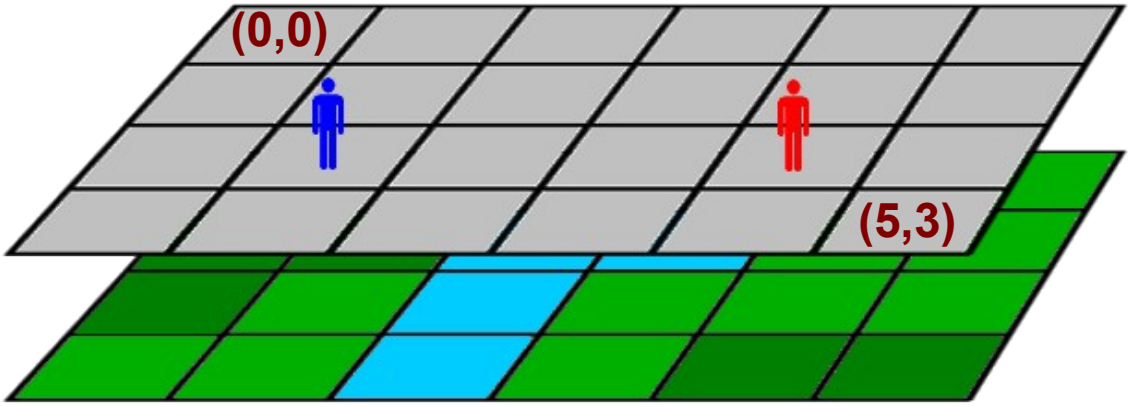
Plusieurs cartes (de terrain) peuvent être enregistrées pour être réutilisées ultérieurement. Celles-ci sont alors stockées sous forme de grille (liste de liste) dans un fichier spécifique. Au démarrage de l'application, l'ensemble de ces cartes sont chargées et stockées dans un dictionnaire, "defaut" fait donc référence à une clé de ce dictionnaire auquel est associé la carte à importer.

Plateau.py

Méthodes :

grilleObstacle(self,xd,yd)

grilleObstacle(self,1,2)



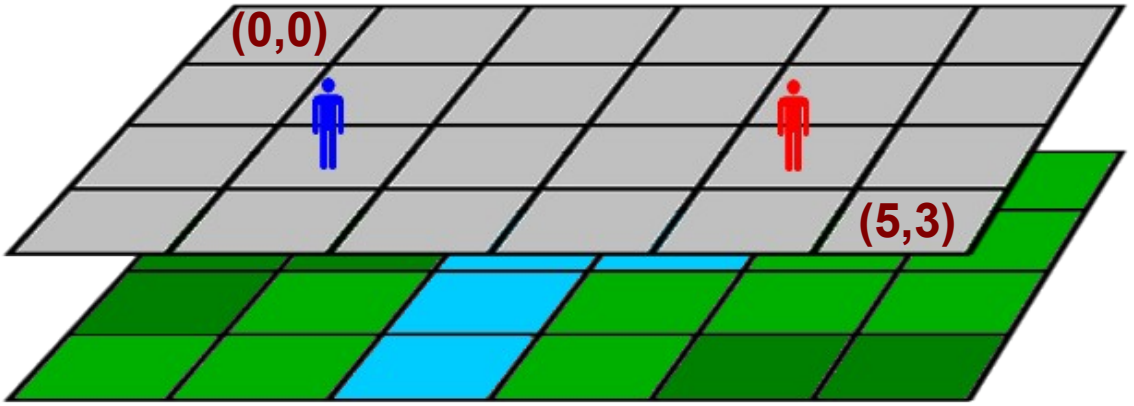
Pour cet exemple, la fonction grilleObstacle va se centrer essentiellement sur l'unité (bleu) aux coordonnées (1,2). Nous admettrons que cette unité peut se déplacer partout sur cette carte de dimensions 6 sur 4. La fonction grilleObstacle va donc générer une grille, de même dimensions, uniquement composée de 0 pour les obstacles (naturel ou ennemi) et de 1 pour les cases traversables.

Plateau.py

Méthodes :

grilleObstacle(self,xd,yd)

grilleObstacle(self,1,2)



Return

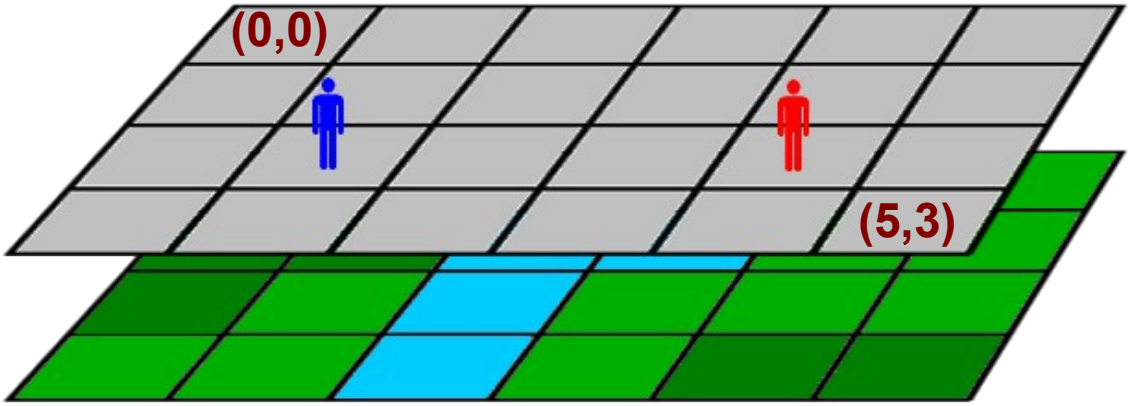
1	1	1	0	1	1
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1

Plateau.py

Méthodes :

grilleObstacle(self,xd,yd)

grilleObstacle(self,1,2)



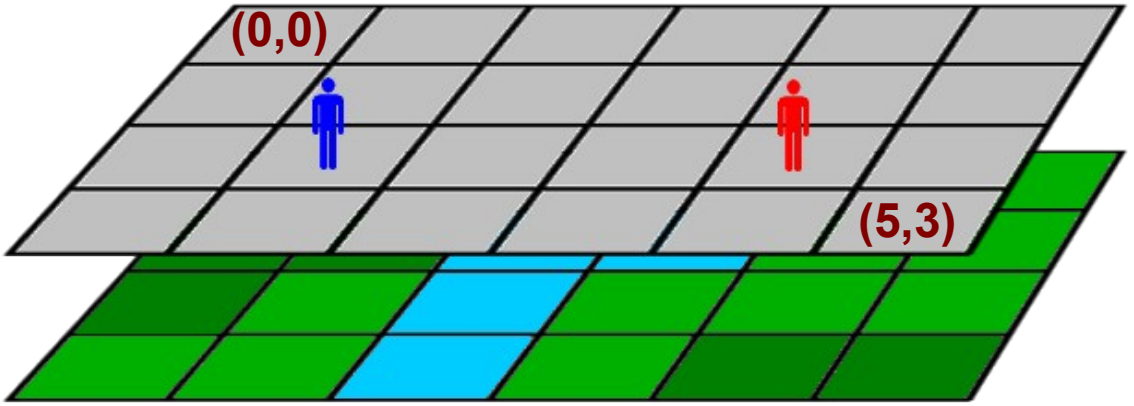
Second cas, si notre unité (bleu) ne peut se déplacer que de 1 case, la fonction grilleObstacle se centralisera sur celle-ci, en revanche, elle ne fournira qu'une grille minimisée suffisante, c'est à dire, ici, une grille de dimensions 3 sur 3.
Ne pouvant se déplacer que de 1 case, cette unité pourra au maximum se diriger d'une case à "droite", à "gauche", en "haut" ou en "bas", la grille minimisée répond alors à toutes les possibilités de déplacement de notre unité.

Plateau.py

Méthodes :

grilleObstacle(self,xd,yd)

grilleObstacle(self,1,2)



Return

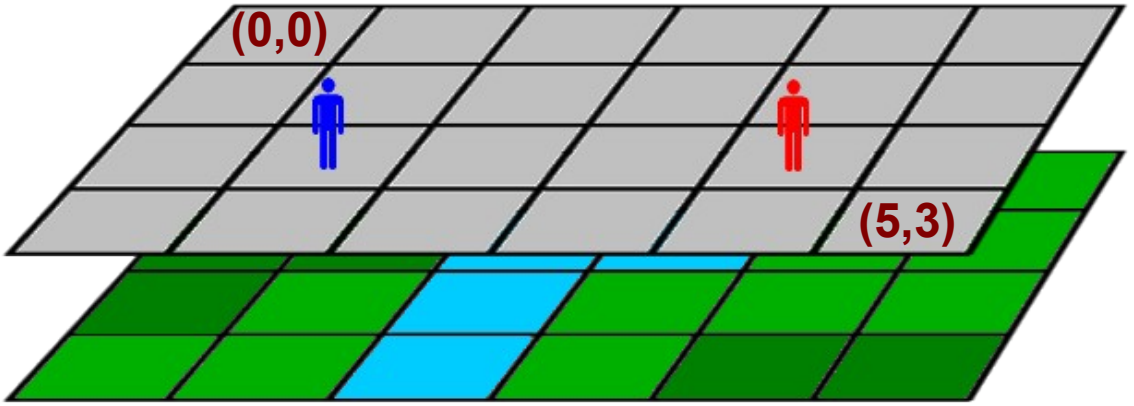
1	**1**	**0**
1	**1**	**0**
1	**1**	**0**

Plateau.py

Méthodes :

grilleObstacle(self,xd,yd)

grilleObstacle(self,1,2)



Cette "réduction" de grille permettra par la suite de limiter les comparaisons à réaliser, notamment dans l'Algorithme A*. Sur une grille de dimensions 40 sur 40, pour une unité ayant un déplacement maximum de 7 cases, on aura $(7x2+1)^2 = 225$ comparaisons à réaliser, contre $40^2 = 1600$ comparaisons pour la grille entière.

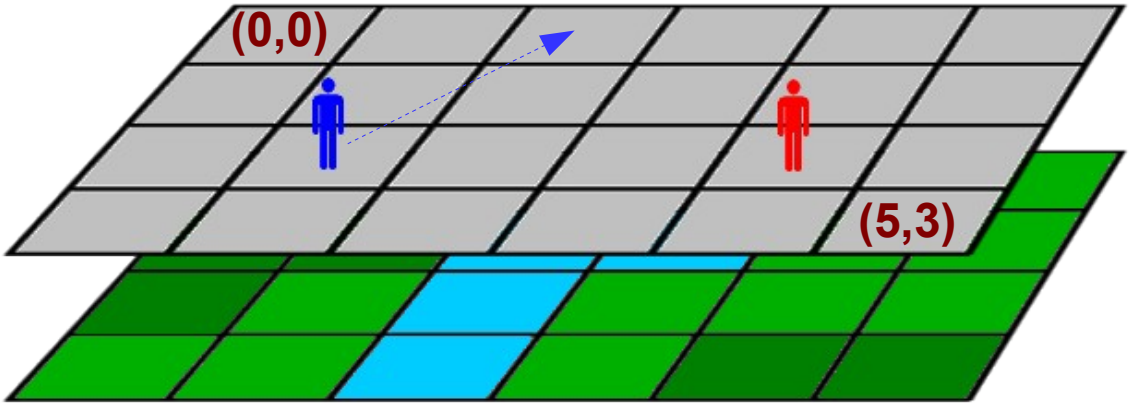
Plateau.py

1	1	1	0	1	1
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1

Méthodes :

deplacement(self,xd,yd,xa,ya)

deplacement(self,1,2,2,0)



Lors du lancement de la méthode de déplacement, la méthode grilleObstacle, vu précédemment, sera lancé afin d'analyser, plus tard, si un chemin est possible ou non. Ensuite, nous observons dans un premier temps si un chemin direct (un déplacement sur les x et un autre sur les y) est possible (Voir analyse "Gain de cycles avec Astar"), si non, on lance l'Algorithme Astar. Si un chemin est renvoyé, on observe si celui-ci n'excède pas la capacité de déplacement de l'unité.

Plateau.py

Méthodes :

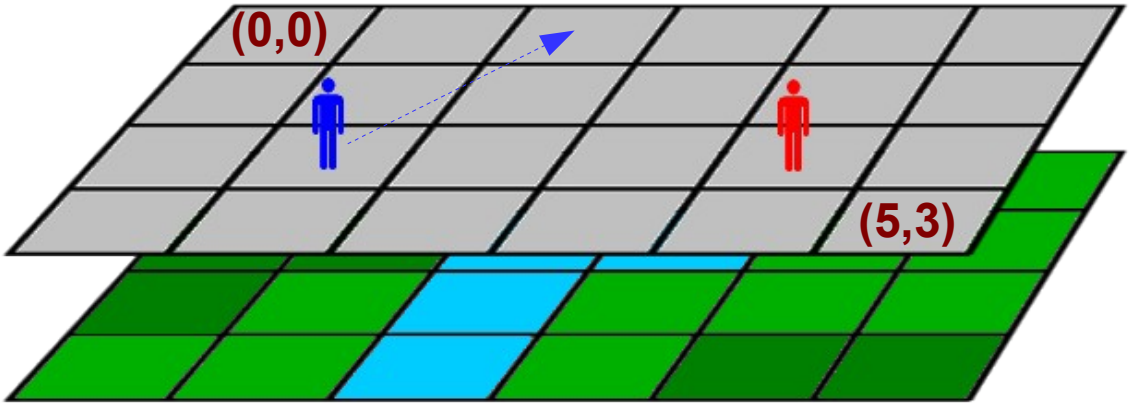
deplacement(self,xd,yd,xa,ya)

1	1	1	0	1	1
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1

Chemin$_1$ direct possible

Chemin$_2$ direct impossible

deplacement(self,1,2,2,0)



Pour déplacer notre unité (bleu), en admettant qu'elle peut se déplacer d'autant de cases qu'elle le souhaite, aux coordonnées (2,0), un chemin direct est possible (2 cases vers le haut, 1 vers la droite). Dans le cas où un obstacle lui ferait barrage, par exemple aux coordonnées (1,1), l'Algorithme Astar serait déclenché est renverrait le chemin suivant : 1 case à gauche, 2 cases en haut puis 2 cases à droite.

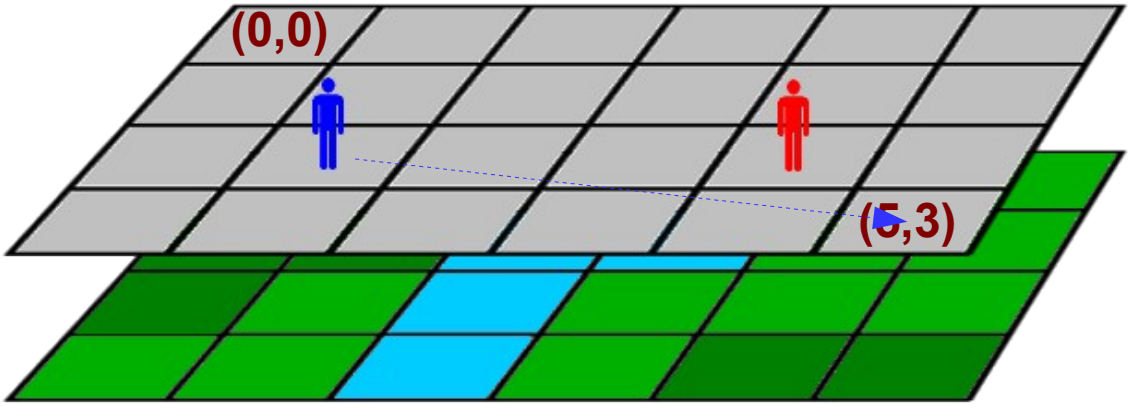
Plateau.py

1	1	1	0	1	1
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1

Méthodes :

deplacement(self,xd,yd,xa,ya)

deplacement(self,1,2,5,3)



Dernier cas, si notre unité (bleu) veut se déplacer aux coordonnées (5,3), il n'y aura pas de chemin direct utilisable, la rivière empêchant un passage direct. Astar va donc se déclencher, mais la rivière bloquant totalement le passage, l'Algorithme renverra la valeur False, autrement dit, il n'y a pas moyen d'accéder aux coordonnées voulues. Dans ce cas, ou dans celui où le chemin serait trop long, aucun déplacement ne sera effectué.

Plateau.py

En lien avec l'objet Plateau() :

Module de Sauvegarde, contient entre autres :

importerSauvegarde(self**,game) *"game" est chargé au lancement de l'application, marche sur le même principe que "defaut" de la méthode d'importation de terrain.

Récupérera le fichier de nom "game", ainsi donc les grilles troupes et terrains, les compositions d'armées, le joueur devant jouer etc...

sauvegarderPartie(self**,nomSauvegarde)

Stockera dans un fichier, entre autres, les grilles troupes et terrains, les compositions d'armées, le tour de tel ou tel joueur ...