Évitements et chemins

Christophe Papazian

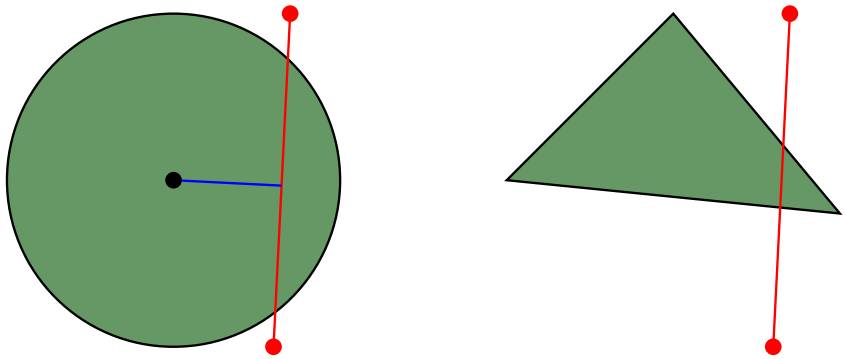
Polytech Nice Sophia

QGL — SI3 — 2022

Évitements

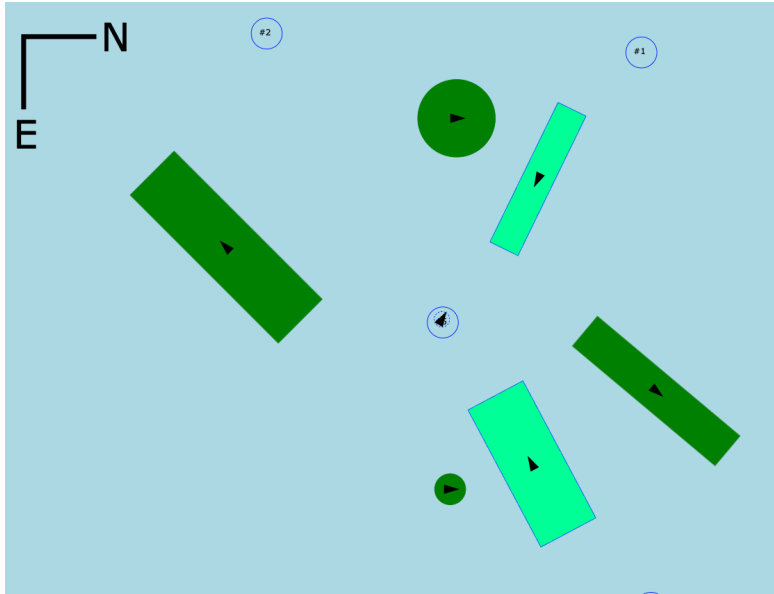
Savoir éviter les obstacles - Collisions

1. Intersections trajectoire - obstacle



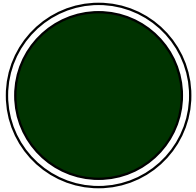
Savoir éviter les obstacles - Collisions

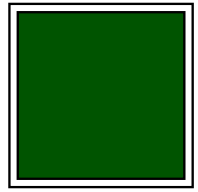
1. Intersections trajectoire - obstacle
2. Tri par boîte limite



Savoir éviter les obstacles - Collisions

1. Intersections trajectoire - obstacle
2. Tri par boîte limite
3. Augmentation des zones de récifs (marge de sureté)





Savoir éviter les obstacles - Stratégie

1. Stratégie locale

Savoir éviter les obstacles - Stratégie

1. Stratégie locale
2. Simple à programmer, efficace dans les cas simples

Savoir éviter les obstacles - Stratégie

1. Stratégie locale
2. Simple à programmer, efficace dans les cas simples
3. Finit toujours par arriver, mais peut être non optimale

Savoir éviter les obstacles - Labyrinthe

1. Pour se sortir d'un labyrinthe simple

Savoir éviter les obstacles - Labyrinthe

1. Pour se sortir d'un labyrinthe simple
2. Choisir toujours la même direction

NOK Si je ne peux pas avancer je tourne à droite pour trouver une trajectoire disponible.

OK Si je peux avancer mais que je ne suis pas dans l'axe, je tourne à gauche tant que la trajectoire est possible (et qu'on n'est pas encore dans l'axe.)

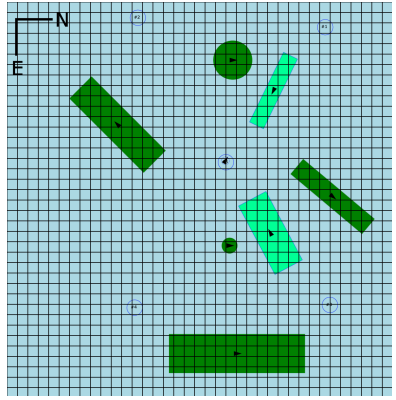
Savoir éviter les obstacles - Labyrinthe

1. Pour se sortir d'un labyrinthe simple
2. Choisir toujours la même direction
3. Limité à 2D et sortie sur le contour

Recherche du chemin le plus court

Chemin le plus court - Dijkstra

1. Discrétisation



obstacle dans la case : case inaccessible

Chemin le plus court - Dijkstra

1. Discrétisation
2. Algorithme par file de priorité (distance en tours)

   DS Pour chaque case : distance $(\infty, 0)$, parcourue (`false`), antécédent + file de priorité par distance (case initiale)

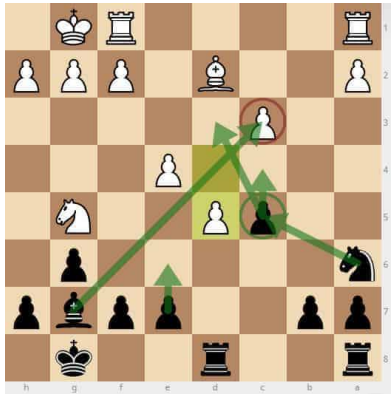
   Algo On défile. si la case n'est pas encore traitée, on la note parcourue, et mise à jour des distances des cases voisines. Si mise à jour, change antécédent et on enfile.

Chemin le plus court - Dijkstra

1. Discrétisation
2. Algorithme par file de priorité (distance en tours)
3. Prise en compte simple des aléas (grâce au voisinage)

Chemin le plus court - Stratégie

1. Parcours en largeur



Chemin le plus court - Stratégie

1. Parcours en largeur
2. Permet de prévoir les coups exactement

Chemin le plus court - Stratégie

1. Parcours en largeur
2. Permet de prévoir les coups exactement
3. Nécessite une bonne simulation

Chemin le plus court - Stratégie

1. Parcours en largeur
2. Permet de prévoir les coups exactement
3. Nécessite une bonne simulation
4. Structure et critères permettant d'éviter les doublons pour plus de profondeur.

Chemin le plus court - A*

1. Mix des précédentes stratégies

Chemin le plus court - A*

1. Mix des précédentes stratégies
2. Algorithme par file de priorité sur distance trouvé + estimation de la distance restante.

Chemin le plus court - A*

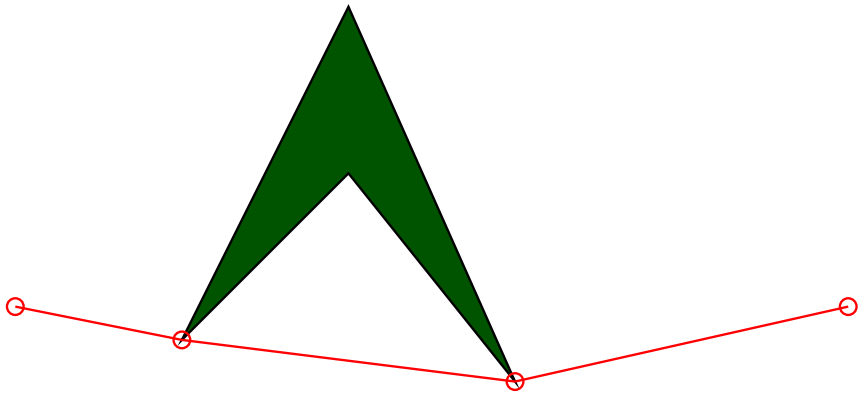
1. Mix des précédentes stratégies
2. Algorithme par file de priorité sur distance trouvé + estimation de la distance restante.
3. Prise en compte simple des aléas

Chemin le plus court - A*

1. Mix des précédentes stratégies
2. Algorithme par file de priorité sur distance trouvé + estimation de la distance restante.
3. Prise en compte simple des aléas
4. Possible amélioration de Dijsktra ?

Chemin le plus court - Carte de Visibilité

1. Purs calculs mathématiques avec vectorisation



Chemin le plus court - Carte de Visibilité

1. Purs calculs mathématiques avec vectorisation
2. Algorithme rapide et efficace

Chemin le plus court - Carte de Visibilité

1. Purs calculs mathématiques avec vectorisation
2. Algorithme rapide et efficace
3. Difficile à mettre en oeuvre avec les aléas
   (calcul différentiel pour les courants)

Intégration
Adaptation/Implémentation
Optimisation
à vous de jouer !