IA02
Projet Helltaker

Dereux Robin
Laval Alexandre
Adorni Pierre
(TP3A-Tr1)

14 juin 2022

Réalisation d'une "IA" résolvant le jeu *Helltaker*, par SatPlan et Recherche dans un espace d'états

Table des matières

Introduction

Afin de garantir une certaine qualité de code dans nos programmes, nous avons utilisé un outil de *lint* (black) couplé avec le système d'**intégration continue** de GitHub. Ainsi, les développeurs étaient obligés de repasser sur leur code avant de pouvoir pousser sur une branche protégée.

De plus, nous avons suivi une logique de **développement piloté par les tests** à l'aide du package `unittest` pour développer petit à petit les fonctionnalités de notre solver tout en nous assurant que nous n'avons rien cassé ailleurs.

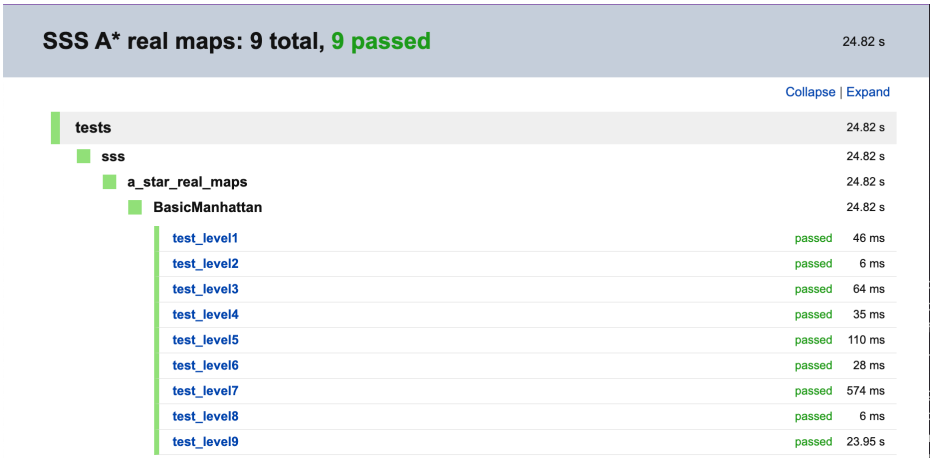

FIGURE 1 – Sortie des tests pour la REE A*

1 Présentation du jeu

1.1 Présentation des règles du jeu

Helltaker est un jeu vidéo freeware du style Sokoban possédant plusieurs niveaux, dans lequel le joueur doit trouver l'issue d'un labyrinthe en un nombre d'actions limité. Il s'agit d'un jeu séquentiel à un joueur. Les objets pouvant se situer dans un niveau sont les suivants :

- le héros
- une caisse poussable par le héros
- un ennemi poussable par le héros, et tuable
- un démon (la sortie du labyrinthe)
- une porte fermée
- une clef (permet d'ouvrir la porte)
- une pointe
- un piège (pointe ouverte un état sur deux)

Le personnage est piloté par seulement quatre touches : haut, bas, gauche, droite ; effectuant des actions différentes en fonction de la position.

FIGURE 2 – Niveau 7 de *Helltaker*

Voici la liste des règles que nous avons considérée pour écrire nos programmes :

- Le personnage peut se déplacer sur une case adjacente si elle n'est pas occupée par une caisse, un ennemi, une porte ou un démon.
- Le personnage peut pousser une caisse adjente si la case derrière elle n'est pas occupée par une caisse, un ennemi, une porte ou un démon.
- Le personnage peut pousser un ennemi adjacent si la case derrière lui n'est pas occupée par un démon.
- Le personnage peut donner un coup de pied dans une caisse adjacente si la case derrière elle est occupée par une caisse, un ennemi, une porte ou un démon, ce qui n'a aucun effet sur la position de la caisse ou du personnage.
- Un ennemi se situant sur une caisse, une porte, une pointe ou un piège actif est tué instantanément.
- Chaque action effectuée par le personnage réduit son compte d'actions de 1.
- Si le personnage se situe sur une pointe ou un piège actif après avoir effectué une action, son compte d'actions est réduit de 2 au lieu de 1.
- Chaque action effectuée par le personnage a pour effet d'échanger les pièges actifs et les pièges inactifs, et ce, avant le calcul de réduction du nombre d'actions.

1.1.1 Représentation STRIPS

Prédicats

```
# un objet se situe en position P
caisse\1; ennemi\1; héros\1; clé\1; porte\1
# prédicats constants
démon\1
# définition des positions en prenant en compte les murs
gauche\2; haut\2
nombreCoups\1
```

Buts

```
héros(P1), démon(P2), gauche(P1, P2)
héros(P1), démon(P2), gauche(P2, P1)
héros(P1), démon(P2), haut(P1, P2)
héros(P1), démon(P2), haut(P2, P1)
```

Actions

```
Action movement_droite(P)
    Précondition:
        héros(P)
        ∧ gauche(P,P1)
        ∧ ¬caisse(P1)
        ∧ ¬ennemi(P1)
        ∧ ¬porte(P1)
        ∧ ¬clé(P1)
    Postcondition:
        ¬héros(P)
        ∧ gauche(P,P1)
        ∧ héros(P1)
...

Action pousser_caisse_haut(P)
    Précondition:
        héros(P)
        ∧ haut()
```

2 Recherche dans un espace d'états

3 Planification par SAT

Résultats - Conclusion

⋆ ⋆ ⋆