Projet CPS

Par Nathan GERDAY

Présentation rapide

Travail réalisé :

Sujet de base avec

- Affichage en ASCII dans le terminal
- Gestion de niveaux
- Vies et scores
- Gardes qui portent des trésors

Présentation rapide

Travail réalisé :

Extensions

- Direction du joueur
- Armes ramassables : Pistolet, Épée, Bombe Aveuglante
- Une clé ramassable et des portes à ouvrir avec
- Des pièges qui disparaissent lorsqu'on marche dessus
- Des cases dans lesquelles les joueurs ou les gardes ne peuvent pas aller

Points intéressants de spécification

Spécification : Utiliser un item

- Une seule méthode pour gérer l'utilisation en fonction de l'item équipé.
- On définit un prédicat permettant de savoir si on peut utiliser un item

CanUseItem(C) defined by
CurrentlyHeldItem(C) $\neq$ null
and NumberOfUsagesLeftForCurrentItem(C) ≥ 1

Spécification : Utiliser un item (Clé)

- 2 cas à gérer en fonction de la direction dans laquelle on regarde
- On vérifie que la porte s'est bien ouverte et que la case est devenue vide

CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Key **and** FacingRight(C)
and Col(C) < Environment::Width(Envi(C)) - 1
and Environment::CellNature(Envi(C), Col(C)+1, Hgt(C)) = DOR
implies Environment::CellNature(Envi(UseItem(C), Col(C)+1, Hgt(C)) = EMP
CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Key **and not** FacingRight(C)
and Col(C) > 0
and Environment::CellNature(Envi(C), Col(C)-1, Hgt(C)) = DOR
implies Environment::CellNature(Envi(UseItem(C), Col(C)-1, Hgt(C)+1) = EMP

Spécification : Utiliser un item (Bombe Flash)

- Au niveau de l'utilisation, juste un case simple, on vérifie que tous les gardes sont bien paralysés.

```
forall Guard g in Engine::Guards(Engi(UseItem(C))),
  CanUseItem and Item::Nature(CurrentlyHeldItem(C)) = Flash
    implies Guard::TimeLeftParalyzed(g) = 10
```

- Nécessite cependant d'ajouter toute une gestion de la paralysie dans les gardes.

Spécification : Utiliser un item (Épée)

- On vérifie pour chaque case à portée que s'il y avait un garde, il est bien mort. Exemple pour une case :

CanUseItem **and** Item::Nature(CurrentlyHeldItem(C)) = Sword **and** Col(C) - 2 ≥ 0
and exists Guard g **in** Environment::CellContent(Envi(C), Col(C) - 2, Hgt(C))
implies exists g **in** Environment::CellContrent(Envi(UseItem(C)),
Coord::X(Guard::InitCoords(g)), Coord::Y(Guard::InitCoords(g)))

- Nombre de cases à vérifier fixe.

Spécification : Utiliser un item (Pistolet)

- Vérifier aussi que les gardes en face du joueur sont bien morts
- **Problème :** Nombre de cases variable

Spécification : Utiliser un item (Pistolet)

- Vérifier aussi que les gardes en face du joueur sont bien morts
- **Problème :** Nombre de cases variable
- **Solution :** Regarder chaque garde sur toute la ligne et vérifier la présence d'obstacles

On définit donc un prédicat pour vérifier la présence d'obstacles entre 2 cases sur une ligne :

```
ExistsObstacleBetween(x1, x2, y) defined by
    exists Cell c in (union (forall i in [x1;x2], Environment::CellNature(Envi(C), i, y)))
    with (c in {MTL, PLT, DOR, NPL})
```

Spécification : Utiliser un item (Pistolet)

- On peut alors vérifier que tous les gardes présents sur la ligne et sans obstacles sont bien morts en fonction de la direction du joueur (ici, dans le cas où le joueur regarde à droite) :

```
forall i in [0;Environment::Width(Envi(C))[,
  CanUseItem and Item::Nature(CurrentlyHeldItem(C)) = Gun
    and FacingRight(C)
    and i > Col(C)
    and not ExistsObstacleBetween(Col(C), i, Hgt(C))
    and exists Guard gin Environment::CellContent(Envi(C), i, Hgt(C))
    implies exists g in Environment::CellContent(Envi(UseItem(C),
        Coord::X(Guard::InitCoords(g)), Coord::Y(Guard::InitCoords(g)))
```

Spécification : Retour aux coordonnées initiales

- Lors de la mort, retour à la position initiale
- Au niveau implémentation et spécification, semble être un simple changement de coordonnées
- **Problème :** Que faire s'il y a un autre garde là où on veut aller ?
- **Solution :** Déplacer cet autre garde à ses propres coordonnées initiales, répéter jusqu'à état cohérent.

```
exists Guard g in Environment::CellContent(Envi(C),
  Coord::X(InitCoords(C)), Coord::Y(InitCoords(C))) implies exists g in
  Environment::CellContent(Envi(MoveToInitCoords(C)),
  Coord::X(InitCoords(g)), Coord::Y(InitCoords(g)))
```

Spécification : Gardes portant des trésors

Beaucoup de cas différents en fonction de si :

- Le garde porte déjà un trésor
- Il y a un trésor dans la case du garde
- Il y a un trou sous le garde

Spécification : Gardes portant des trésors

```
exists Treasure t in Environment::CellContent(Envi(C), Col(C), Hgt(C))
  and not CarryingTreasure(C)
  implies CarryingTreasure(Step(C))
    and not exists t in Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))
exists Treasure t in Environment::CellContent(Envi(C), Col(C), Hgt(C))
  and CarryingTreasure(C)
  implies CarryingTreasure(Step(C))
    and exists t in Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))
not exists Treasure t in Environment::CellContent(Envi(C), Col(C), Hgt(C))
  and not CarryingTreasure(C)
  implies not CarryingTreasure(Step(C))
    and not exists t in Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))
```

Spécification : Gardes portant des trésors

not exists Treasure t **in** Environment::CellContent(Envi(C), Col(C), Hgt(C))
  and CarryingTreasure(C)
  and Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) = HOL
  and not Environment::CellNature(Envi(C), Col(C), Hgt(C)) **in** {**LAD**, **HDR**, **HOL**}
  and not exists Guard g **in** Environment::CellContent(Envi(C), Col(C), Hgt(C)-1)
  implies not CarryingTreasure(Step(C))
    and exists Treasure t2 **in** Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))
not exists Treasure t **in** Environment::CellContent(Envi(C), Col(C), Hgt(C))
  and CarryingTreasure(C)
  (**and** Environment::CellNature(Envi(C), Col(C), Hgt(C)-1) ≠ HOL
    or Environment::CellNature(Envi(C), Col(C), Hgt(C)) **in** {**LAD**, **HDR**, **HOL**}
    or exists Guard g **in** Environment::CellContent(Envi(C), Col(C), Hgt(C)-1))
  implies CarryingTreasure(Step(C))
    and not exists Treasure t2 **in** Environment::CellContent(Envi(Step(C)), Col(C), Hgt(C))

Spécification : Grimper avec joueur au dessus

- Vérification de contact entre Garde et Joueur faite une fois pas step
- **Problème :** Le fait de grimper passe par 2 cases et on ne veut pas qu'un Garde soit dans 2 cases à la fois
- **Solution :** Séparer en 2 cas qui permettent de choisir la meilleur case sur laquelle grimper

Spécification : Grimper avec joueur au dessus

Col(C) $\neq$ 0 **and**
Screen::CellNature(Envi(C),Col(C)-1,Hgt(C)+1) **not in** {**MTL**, **PLT**, **DOR**, **NGU** }
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C)+1)
and exists Player p **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Col(ClimbLeft(C)) = Col(C) **and** Hgt(ClimbLeft(C)) = Hgt(C)+1

Col(C) $\neq$ 0 **and**
Screen::CellNature(Envi(C),Col(C)-1,Hgt(C)+1) **not in** {**MTL**, **PLT**, **DOR**, **NGU** }
and not exists Guard g **in** Environment::CellContent(Envi(C),Col(C)-1,Hgt(C)+1)
and not exists Player p **in** Environment::CellContent(Envi(C),Col(C),Hgt(C)+1)
implies Col(ClimbLeft(C)) = Col(C)-1 **and** Hgt(ClimbLeft(C)) = Hgt(C)+1