M1 Informatique STL
Université Pierre et Marie Curie
2011-2012

Boulder Dash (CPS)

Objectifs de Tests

Maxime Ancelin et Mickaël Menu

17 mai 2012

Table des matières

1 Introduction

1.1 Taux de couverture

Le taux de couverture est de (#objectifs atteints / #objectifs) * 100 %

avec #objectifs = #objectifs atteints = 68.

Donc notre taux de couverture est de 100%.

1.2 Objets de test utilisés

Les objets de test suivant – implémentés dans `tests.ObjectsFactory` – peuvent être référencés dans les cas de test :

- `POS1` $\stackrel{def}{=}$ `Position::init(5, 3, 1, 1)`
- `TER1` $\stackrel{def}{=}$ `Terrain::init(5, 3)`

  Avec pour blocs :

  ```
  .####
  .OX.Y
  ....?
  ```

  - `X` héro
  - `#` mur
  - `.` vide
  - `O` rocher
  - `Y` diamant
  - `?` sortie fermée

2 Service Bloc

Couverture des préconditions

Aucune précondition dans Bloc.

Couverture des invariants

Objectif `Bloc_invariant1` (minimisation de isVide)

Cas de test `Bloc_invariant1` :
Préambule : `B = init(VIDE, POS1)`
Contenu : ∅
Oracle : `isVide(B)`

Objectif `Bloc_invariant2` (minimisation de isSolide)

Cas de test `Bloc_invariant2` :
Préambule : `B = init(ROCHER, POS1)`
Contenu : ∅
Oracle : `isSolide(B)`

Objectif `Bloc_invariant3` (minimisation de isDeplacable)

Cas de test `Bloc_invariant3` :
Préambule : `B = init(ROCHER, POS1)`
Contenu : ∅
Oracle : `isDeplacable(B)`

Objectif `Bloc_invariant4` (minimisation de isTombable)

Cas de test `Bloc_invariant4` :
Préambule : `B = init(ROCHER, POS1)`
Contenu : ∅
Oracle : `isTombable(B)`

Objectif `Bloc_invariant5` (minimisation de isSortie)

Cas de test `Bloc_invariant5` :
Préambule : `B = init(SORTIE_OUVERTE, POS1)`
Contenu : ∅
Oracle : `isSortie(B)`

Objectif `Bloc_invariant6` (minimisation de isSortieFermee)

Cas de test `Bloc_invariant6` :
Préambule : `B = init(SORTIE_FERMEE, POS1)`
Contenu : ∅
Oracle : `isSortieFermee(B)`

Objectif `Bloc_invariant7` (minimisation de isHero)

Cas de test `Bloc_invariant7` :
Préambule : `B = init(HERO, POS1)`
Contenu : ∅
Oracle : `isHero(B)`

Objectif `Bloc_invariant8` (minimisation de isTerre)

Cas de test `Bloc_invariant8` :
Préambule : `B = init(TERRE, POS1)`
Contenu : ∅
Oracle : `isTerre(B)`

Couverture des postconditions

Objectif `Bloc_init_post1` (postcondition de init sur getType)

Cas de test `Bloc_init_post1` :
Préambule : `aucun`
Contenu : `B = init(HERO, POS1)`
Oracle : `getType(B) = HERO`

Objectif `Bloc_init_post2` (postcondition de init sur getPosition)

Cas de test `Bloc_init_post2` :
Préambule : `aucun`
Contenu : `B = init(HERO, POS1)`
Oracle : `getPosition(B) = POS1`

Objectif `Bloc_setType_post1` (postcondition de setType sur getType)

Cas de test `Bloc_setType_post1` :

Préambule : `B1 = init(HERO, POS1)`

Contenu : `B2 = setType(B1, ROCHER)`

Oracle : `getType(B2) = ROCHER`

Objectif `Bloc_setType_post2` (postcondition de setType sur getPosition)

Cas de test `Bloc_setType_post2` :

Préambule : `B1 = init(HERO, POS1)`

Contenu : `B2 = setType(B1, ROCHER)`

Oracle : `getPosition(B2) = POS1`

Couverture des transitions

Objectif `Bloc_setType_trans` (transition de setType)

Cas de test `Bloc_setType_trans` :

Préambule : `B1 = init(HERO, POS1)`

Contenu : `B2 = setType(B1, ROCHER)`

Oracle : `getType(B2) = ROCHER` $\wedge$ `getPosition(B2) = POS1`

3 Service Position

Couverture des préconditions

Objectif `Position_init_pre` (précondition de init)

Cas de test `Position_init_pre_true` :
Préambule : ∅
Contenu : ∅
Oracle : ∃ `MJ, MJ = init(5, 5, 2, 3)`

Cas de test `Position_init_pre_false1` :
Préambule : ∅
Contenu : ∅
Oracle : ∄ `MJ, MJ = init(0, 5, 2, 3)`

Cas de test `Position_init_pre_false2` :
Préambule : ∅
Contenu : ∅
Oracle : ∄ `MJ, MJ = init(5, 0, 2, 3)`

Cas de test `Position_init_pre_false3` :
Préambule : ∅
Contenu : ∅
Oracle : ∄ `MJ, MJ = init(5, 5, -1, 3)`

Cas de test `Position_init_pre_false4` :
Préambule : ∅
Contenu : ∅
Oracle : ∄ `MJ, MJ = init(5, 5, 2, -1)`

Couverture des invariants

Aucun invariant dans Position.

Couverture des postconditions

Objectif `Position_init_post1` (postcondition de init sur getLargeur)

Cas de test `Position_init_post1` :
Préambule : ∅
Contenu : `P = init(5, 4, 2, 3)`
Oracle : `getLargeur(P) = 5`

Objectif `Position_init_post2` (postcondition de init sur getHauteur)

Cas de test `Position_init_post2` :

Préambule : ∅

Contenu : `P = init(5, 4, 2, 3)`

Oracle : `getHauteur(P) = 4`

Objectif `Position_init_post3` (postcondition de init sur getX)

Cas de test `Position_init_post3` :

Préambule : ∅

Contenu : `P = init(5, 4, 2, 3)`

Oracle : `getX(P) = 2 % 5`

Objectif `Position_init_post4` (postcondition de init sur getY)

Cas de test `Position_init_post4` :

Préambule : ∅

Contenu : `P = init(5, 4, 2, 3)`

Oracle : `getY(P) = 3 % 4`

Objectif `Position_deplacerVersDirection_post1` (postcondition de deplacerVersDirection sur getX)

Cas de test `Position_deplacerVersDirection_post1_1` :

Préambule : `P1 = init(5, 4, 2, 3)`

Contenu : `P2 = deplacerVersDirection(P1, GAUCHE)`

Oracle : `getX(P2) = (2 - 1) % 5`

Cas de test `Position_deplacerVersDirection_post1_2` :

Préambule : `P1 = init(5, 4, 2, 3)`

Contenu : `P2 = deplacerVersDirection(P1, DROITE)`

Oracle : `getX(P2) = (2 + 1) % 5`

Cas de test `Position_deplacerVersDirection_post1_3` :

Préambule : `P1 = init(5, 4, 2, 3)`

Contenu : `P2 = deplacerVersDirection(P1, HAUT)`

Oracle : `getX(P2) = 2`

Objectif `Position_deplacerVersDirection_post2` (postcondition de deplacerVersDirection sur getY)

Cas de test `Position_deplacerVersDirection_post2_1` :
Préambule : `P1 = init(5, 4, 2, 3)`
Contenu : `P2 = deplacerVersDirection(P1, HAUT)`
Oracle : `getY(P2) = (3 - 1) % 4`

Cas de test `Position_deplacerVersDirection_post2_2` :
Préambule : `P1 = init(5, 4, 2, 3)`
Contenu : `P2 = deplacerVersDirection(P1, BAS)`
Oracle : `getY(P2) = (3 + 1) % 4`

Cas de test `Position_deplacerVersDirection_post2_3` :
Préambule : `P1 = init(5, 4, 2, 3)`
Contenu : `P2 = deplacerVersDirection(P1, GAUCHE)`
Oracle : `getY(P2) = 3`

Couverture des transitions

Objectif `Position_deplacerVersDirection_trans` (transition de deplacerVersDirection)

Cas de test `Position_deplacerVersDirection_trans1` :
Préambule : `P1 = init(5, 4, 2, 3)`
Contenu : `P2 = deplacerVersDirection(P1, HAUT)`
Oracle : `getY(P2) = (3 - 1) % 4 ∧ getX(P2) = 2`

Cas de test `Position_deplacerVersDirection_trans2` :
Préambule : `P1 = init(5, 4, 2, 3)`
Contenu : `P2 = deplacerVersDirection(P1, BAS)`
Oracle : `getY(P2) = (3 + 1) % 4 ∧ getX(P2) = 2`

Cas de test `Position_deplacerVersDirection_trans3` :
Préambule : `P1 = init(5, 4, 2, 3)`
Contenu : `P2 = deplacerVersDirection(P1, GAUCHE)`
Oracle : `getY(P2) = 3 ∧ getX(P2) = (2 - 1) % 5`

Cas de test `Position_deplacerVersDirection_trans4` :
Préambule : `P1 = init(5, 4, 2, 3)`
Contenu : `P2 = deplacerVersDirection(P1, DROITE)`
Oracle : `getY(P2) = 3 ∧ getX(P2) = (2 + 1) % 5`

4 Service Terrain

Couverture des préconditions

Objectif `Terrain_getBlocHero_pre` (précondition de getBlocHero)

Cas de test `Terrain_getBlocHero_pre_true` :
Préambule : `TER1`
Contenu : $\emptyset$
Oracle : `isHeroVivant(TER1)` $\wedge$ $\exists$ `B, B = getBlocHero(TER1)`

Cas de test `Terrain_getBlocHero_pre_false` :
Préambule : `T = init(5, 5)`
Contenu : $\emptyset$
Oracle : $\neg$`isHeroVivant(TER1)` $\wedge$ $\nexists$ `B, B = getBlocHero(T)`

Objectif `Terrain_init_pre` (précondition de init)

Cas de test `Terrain_init_pre_true` :
Préambule : $\emptyset$
Contenu : $\emptyset$
Oracle : $\exists$ `T, T = init(5, 5)`

Cas de test `Terrain_init_pre_false1` :
Préambule : $\emptyset$
Contenu : $\emptyset$
Oracle : $\nexists$ `T, T = init(0, 5)`

Cas de test `Terrain_init_pre_false2` :
Préambule : $\emptyset$
Contenu : $\emptyset$
Oracle : $\nexists$ `T, T = init(5, 0)`

Objectif `Terrain_deplacerBlocVersDirection_pre` (précondition de deplacerBlocVersDirection)

Cas de test `Terrain_deplacerBlocVersDirection_pre_true` :
Préambule : `TER1`
Contenu : $\emptyset$
Oracle :

```
isDeplacementBlocPossible(TER1, getBlocHero(TER1), DROITE)
∧ ∃ T2, T2 = deplacerBlocVersDirection(TER1, getBlocHero(TER1), DROITE)
```

Cas de test `Terrain_deplacerBlocVersDirection_pre_false` :

Préambule : `TER1`

Contenu : $\emptyset$

Oracle :

$\neg$`isDeplacementBlocPossible(TER1, getBlocHero(TER1), HAUT)`
$\wedge \nexists$ `T2, T2 = deplacerBlocVersDirection(TER1, getBlocHero(TER1), HAUT)`

Couverture des invariants

Objectif `Terrain_invariant1` (minimisation de getBlocHero)

Cas de test `Terrain_invariant1` :

Préambule : `TER1`

Contenu : $\emptyset$

Oracle : `getBlocHero(TER1) = getBloc(2, 1)`

Objectif `Terrain_invariant2` (minimisation de getBlocVersDirection)

Cas de test `Terrain_invariant2` :

Préambule : `TER1`

Contenu : $\emptyset$

Oracle : `getBlocVersDirection(getBlocHero(TER1), DROITE) = getBloc(3, 1)`

Objectif `Terrain_invariant3` (minimisation de isHeroVivant)

Cas de test `Terrain_invariant3_true` :

Préambule : `TER1`

Contenu : $\emptyset$

Oracle : `isHeroVivant(TER1)`

Cas de test `Terrain_invariant3_false` :

Préambule : `T = init(5, 5)`

Contenu : $\emptyset$

Oracle : $\neg$`isHeroVivant(T)`

Objectif `Terrain_invariant4` (minimisation de isDiamantsRestants)

Cas de test `Terrain_invariant4_true` :
Préambule : `TER1`
Contenu : ∅
Oracle : `isDiamantsRestants(TER1)`

Cas de test `Terrain_invariant4_false` :
Préambule : `T = init(5, 5)`
Contenu : ∅
Oracle : ¬`isDiamantsRestants(T)`

Objectif `Terrain_invariant5` (minimisation de isDeplacementBlocPossible)

Cas de test `Terrain_invariant5_true` :
Préambule : `TER1`
Contenu : ∅
Oracle : `isDeplacementBlocPossible(TER1, Terrain::getBlocHero(TER1), DROITE)`

Cas de test `Terrain_invariant5_false` :
Préambule : `TER1`
Contenu : ∅
Oracle : ¬`isDeplacementBlocPossible(TER1, Terrain::getBlocHero(TER1), HAUT)`

Objectif `Terrain_invariant6` (minimisation de getBlocDepuisPosition)

Cas de test `Terrain_invariant6` :
Préambule : `TER1, POS1`
Contenu : ∅
Oracle : `getBlocDepuisPosition(TER1, POS1) = getBloc(1, 1)`

Objectif `Terrain_invariant7` (minimisation de getBlocs)

Cas de test `Terrain_invariant7` :
Préambule : `TER1`
Contenu : ∅
Oracle :

```
getBlocs(TER1) = {
    Bloc::init(VIDE, Position::init(0, 0))
    Bloc::init(MUR, Position::init(1, 0))
    Bloc::init(MUR, Position::init(2, 0))
    Bloc::init(MUR, Position::init(3, 0))
    Bloc::init(MUR, Position::init(4, 0))
    Bloc::init(VIDE, Position::init(0, 1))
    Bloc::init(ROCHER, Position::init(1, 1))
    Bloc::init(HERO, Position::init(2, 1))
    Bloc::init(VIDE, Position::init(3, 1))
    Bloc::init(DIAMANT, Position::init(4, 1))
    Bloc::init(VIDE, Position::init(0, 2))
    Bloc::init(VIDE, Position::init(1, 2))
    Bloc::init(VIDE, Position::init(2, 2))
    Bloc::init(VIDE, Position::init(3, 2))
    Bloc::init(SORTIE_FERMEE, Position::init(4, 2))
}
```

Couverture des postconditions

Objectif `Terrain_init_post1` (postcondition de init sur getLargeur)

Cas de test `Terrain_init_post1` :

Préambule : ∅

Contenu : `T = init(10, 15)`

Oracle : `getLargeur(T) = 10`

Objectif `Terrain_init_post2` (postcondition de init sur getHauteur)

Cas de test `Terrain_init_post2` :

Préambule : ∅

Contenu : `T = init(10, 15)`

Oracle : `getHauteur(T) = 15`

Objectif `Terrain_init_post3` (postcondition de init sur getPosSortie)

Cas de test `Terrain_init_post3` :

Préambule : ∅

Contenu : `T = init(10, 15)`

Oracle : `getPosSortie(T) = ∅`

Objectif `Terrain_init_post4` (postcondition de init sur getPosHero)

Cas de test `Terrain_init_post4` :
Préambule : ∅
Contenu : `T = init(10, 15)`
Oracle : `getPosHero(T) = ∅`

Objectif `Terrain_init_post5` (postcondition de init sur getBloc)

Cas de test `Terrain_init_post5` :
Préambule : ∅
Contenu : `T = init(10, 15)`
Oracle : `∀x ∈ [0..9], ∀y ∈ [0..14], getBloc(T, x, y) = ∅`

Objectif `Terrain_setBloc_post1` (postcondition de setBloc sur getPosSortie)

Cas de test `Terrain_setBloc_post1_conseq` :
Préambule : `TER1`
Contenu : `T2 = setBloc(TER1, SORTIE_FERMEE, 1, 2)`
Oracle : `getPosSortie(T2) = Position::init(5, 3, 1, 2)`

Cas de test `Terrain_setBloc_post1_alt` :
Préambule : `TER1`
Contenu : `T2 = setBloc(TER1, ROCHER, 1, 2)`
Oracle : `getPosSortie(T2) = Position::init(5, 3, 4, 2)`

Objectif `Terrain_setBloc_post2` (postcondition de setBloc sur getPosHero)

Cas de test `Terrain_setBloc_post2_conseq` :
Préambule : `TER1`
Contenu : `T2 = setBloc(TER1, HERO, 1, 2)`
Oracle : `getPosHero(T2) = Position::init(5, 3, 1, 2)`

Cas de test `Terrain_setBloc_post2_alt` :
Préambule : `TER1`
Contenu : `T2 = setBloc(TER1, VIDE, 1, 2)`
Oracle : `getPosHero(T2) = Position::init(5, 3, 2, 1)`

Objectif `Terrain_setBloc_post3` (postcondition de setBloc sur getBloc)

Cas de test `Terrain_setBloc_post3` :

Préambule : `TER1`

Contenu : `T2 = setBloc(TER1, ROCHER, 3, 1)`

Oracle :

```
∀x ∈ [0..4], ∀y ∈ [0..2],
let bloc = getBloc(T2, x, y)
in
    if x = 3 ∧ y = 1 then
        Bloc::getType(bloc) = ROCHER
    else
        bloc = getBloc(TER1, x, y)
```

Objectif `Terrain_deplacerBlocVersDirection_post1` (postcondition de deplacerBlocVersDirection sur getPosSortie)

Cas de test `Terrain_deplacerBlocVersDirection_post1` :

Préambule : `TER1`

Contenu : `T2 = deplacerBlocVersDirection(TER1, getBlocHero(TER1), DROITE)`

Oracle : `getPosSortie(T2) = Position::init(5, 3, 4, 2)`

Objectif `Terrain_deplacerBlocVersDirection_post2` (postcondition de deplacerBlocVersDirection sur getPosHero)

Cas de test `Terrain_deplacerBlocVersDirection_post2_conseq` :

Préambule : `TER1`

Contenu : `T2 = deplacerBlocVersDirection(TER1, getBlocHero(TER1), DROITE)`

Oracle : `getPosHero(T2) = Position::init(5, 3, 3, 1)`

Cas de test `Terrain_deplacerBlocVersDirection_post2_alt` :

Préambule : `TER1`

Contenu : `T2 = deplacerBlocVersDirection(TER1, getBloc(1, 1), BAS)`

Oracle : `getPosHero(T2) = Position::init(5, 3, 2, 1)`

Objectif `Terrain_deplacerBlocVersDirection_post3` (postcondition de deplacerBlocVersDirection sur getBloc)

Cas de test `Terrain_deplacerBlocVersDirection_post3` :

Préambule : `TER1`

Contenu : `T2 = deplacerBlocVersDirection(TER1, getBlocHero(TER1), DROITE)`

Oracle :

```
∀x ∈ [0..4], ∀y ∈ [0..2],
let bloc = getBloc(T2, x, y)
in
    if x = 2 ∧ y = 1 then
        Bloc::isVide(bloc)
    else if x = 3 ∧ y = 1 then
        Bloc::isHero(bloc)
    else
        bloc = getBloc(TER1, x, y)
```

Objectif `Terrain_fairePasDeMiseAJour_post1` (postcondition de fairePasDeMiseAJour sur getPosSortie)

Cas de test `Terrain_fairePasDeMiseAJour_post1` :

Préambule : `TER1`

Contenu : `T2 = fairePasDeMiseAJour(TER1)`

Oracle : `getPosSortie(T2) = Position::init(5, 3, 4, 2)`

Objectif `Terrain_fairePasDeMiseAJour_post2` (postcondition de fairePasDeMiseAJour sur getPosHero)

Cas de test `Terrain_fairePasDeMiseAJour_post2` :

Préambule : `TER1`

Contenu : `T2 = fairePasDeMiseAJour(TER1)`

Oracle : `getPosHero(T2) = Position::init(5, 3, 2, 1)`

Objectif `Terrain_fairePasDeMiseAJour_post3` (postcondition de fairePasDeMiseAJour sur getBloc)

Cas de test `Terrain_fairePasDeMiseAJour_post3` :

Préambule : `TER1`

Contenu : `T2 = fairePasDeMiseAJour(TER1)`

Oracle :

```
∀x ∈ [0..4], ∀y ∈ [0..2],
let bloc = getBloc(T2, x, y)
in
    if x = 1 ∧ y = 1 then
        Bloc::isVide(bloc)
    else if x = 1 ∧ y = 2 then
        Bloc::getType(bloc) = ROCHER
    else
        bloc = getBloc(TER1, x, y)
```

Couverture des transitions

Objectif `Terrain_setBloc_trans` (transition de setBloc)

Cas de test `Terrain_setBloc_trans1` :

Préambule : `TER1`

Contenu : `T2 = setBloc(TER1, SORTIE_FERMEE, 1, 2)`

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 1, 2)
∧ getPosHero(T2) = Position::init(5, 3, 2, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 1 ∧ y = 2 then
            Bloc::isSortieFermee(bloc)
        else
            bloc = getBloc(TER1, x, y)
```

Cas de test `Terrain_setBloc_trans2` :

Préambule : `TER1`

Contenu : `T2 = setBloc(TER1, HERO, 1, 2)`

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 1, 2)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 1 ∧ y = 2 then
            Bloc::isHero(bloc)
        else
            bloc = getBloc(TER1, x, y)
```

Objectif `Terrain_deplacerBlocVersDirection_trans` (transition de deplacerBlocVersDirection)

Cas de test `Terrain_deplacerBlocVersDirection_trans1` :

Préambule : `TER1`

Contenu : `T2 = deplacerBlocVersDirection(TER1, getBlocHero(TER1), DROITE)`

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 3, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 2 ∧ y = 1 then
```

```
            Bloc::isVide(bloc)
        else if x = 3 ∧ y = 1 then
            Bloc::isHero(bloc)
        else
            bloc = getBloc(TER1, x, y)
```

Cas de test `Terrain_deplacerBlocVersDirection_trans2` :

Préambule : `TER1`

Contenu : `T2 = deplacerBlocVersDirection(TER1, getBloc(1, 1), BAS)`

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 2, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 1 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 1 ∧ y = 2 then
            Bloc::getType(bloc) = ROCHER
        else
            bloc = getBloc(TER1, x, y)
```

Objectif `Terrain_fairePasDeMiseAJour_trans` (transition de fairePasDeMiseAJour)

Cas de test `Terrain_fairePasDeMiseAJour_trans` :

Préambule : `TER1`

Contenu : `T2 = fairePasDeMiseAJour(TER1)`

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 2, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 1 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 1 ∧ y = 2 then
            Bloc::getType(bloc) = ROCHER
        else
            bloc = getBloc(TER1, x, y)
```

Couverture des paires de transitions

Objectif `Terrain_setBloc_deplacerBlocVersDirection_trans` (transitions de setBloc puis deplacerBlocVersDirection)

Cas de test `Terrain_setBloc_deplacerBlocVersDirection_trans` :

Préambule : TER1

Contenu :

```
T2 = deplacerBlocVersDirection(setBloc(TER1, TERRE, 0, 0),
        getBlocHero(TER1), DROITE)
```

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 3, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 0 ∧ y = 0 then
            Bloc::isTerre(bloc)
        else if x = 2 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 3 ∧ y = 1 then
            Bloc::isHero(bloc)
        else
            bloc = getBloc(TER1, x, y)
```

Objectif `Terrain_setBloc_fairePasDeMiseAJour_trans` (transitions de setBloc puis fairePasDeMiseAJour)

Cas de test Terrain_setBloc_fairePasDeMiseAJour_trans :

Préambule : TER1

Contenu : `T2 = fairePasDeMiseAJour(setBloc(TER1, TERRE, 0, 0))`

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 2, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 0 ∧ y = 0 then
            Bloc::isTerre(bloc)
        else if x = 1 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 1 ∧ y = 2 then
            Bloc::getType(bloc) = ROCHER
        else
            bloc = getBloc(TER1, x, y)
```

Objectif `Terrain_deplacerBlocVersDirection_setBloc_trans` (transitions de deplacerBlocVersDirection puis setBloc)

Cas de test Terrain_deplacerBlocVersDirection_setBloc_trans :

Préambule : TER1

Contenu :

```
T2 = setBloc(deplacerBlocVersDirection(TER1, getBlocHero(TER1), DROITE),
       TERRE, 0, 0)
```

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 3, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 0 ∧ y = 0 then
            Bloc::isTerre(bloc)
        else if x = 2 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 3 ∧ y = 1 then
            Bloc::isHero(bloc)
        else
            bloc = getBloc(TER1, x, y)
```

Objectif Terrain_deplacerBlocVersDirection_fairePasDeMiseAJour_trans (transitions de deplacerBlocVersDirection puis fairePasDeMiseAJour)

Cas de test Terrain_deplacerBlocVersDirection_fairePasDeMiseAJour_trans :

Préambule : TER1

Contenu :

```
T2 = fairePasDeMiseAJour(
       deplacerBlocVersDirection(TER1, getBlocHero(TER1), DROITE))
```

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 3, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 2 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 3 ∧ y = 1 then
            Bloc::isHero(bloc)
        else if x = 1 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 1 ∧ y = 2 then
            Bloc::getType(bloc) = ROCHER
        else
            bloc = getBloc(TER1, x, y)
```

Objectif `Terrain_fairePasDeMiseAJour_deplacerBlocVersDirection_trans` (transitions de fairePasDeMiseAJour puis deplacerBlocVersDirection)

Cas de test `Terrain_fairePasDeMiseAJour_deplacerBlocVersDirection_trans` :

Préambule : `TER1`

Contenu :

```
T2 = deplacerBlocVersDirection(fairePasDeMiseAJour(TER1),
        getBlocHero(TER1), DROITE)
```

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 3, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 2 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 3 ∧ y = 1 then
            Bloc::isHero(bloc)
        else if x = 1 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 1 ∧ y = 2 then
            Bloc::getType(bloc) = ROCHER
        else
            bloc = getBloc(TER1, x, y)
```

Objectif `Terrain_fairePasDeMiseAJour_setBloc_trans` (transitions de fairePasDeMiseAJour puis setBloc)

Cas de test `Terrain_fairePasDeMiseAJour_setBloc_trans` :

Préambule : `TER1`

Contenu : `T2 = setBloc(fairePasDeMiseAJour(TER1), TERRE, 0, 0)`

Oracle :

```
getPosSortie(T2) = Position::init(5, 3, 4, 2)
∧ getPosHero(T2) = Position::init(5, 3, 2, 1)
∧ ∀x ∈ [0..4], ∀y ∈ [0..2],
    let bloc = getBloc(T2, x, y)
    in
        if x = 0 ∧ y = 0 then
            Bloc::isTerre(bloc)
        else if x = 1 ∧ y = 1 then
            Bloc::isVide(bloc)
        else if x = 1 ∧ y = 2 then
            Bloc::getType(bloc) = ROCHER
        else
            bloc = getBloc(TER1, x, y)
```

5 Service MoteurJeu

Couverture des préconditions

Objectif `MoteurJeu_init_pre` (précondition de init)

Cas de test `MoteurJeu_init_pre_true` :

Préambule : ∅

Contenu : ∅

Oracle : ∃ `MJ, MJ = init(TER1, 30)`

Cas de test `MoteurJeu_init_pre_false` :

Préambule : ∅

Contenu : ∅

Oracle : ∄ `MJ, MJ = init(TER1, 0)`

Objectif `MoteurJeu_deplacerHero_pre` (précondition de deplacerHero)

Cas de test `MoteurJeu_deplacerHero_pre_true` :

Préambule : `MJ1 = init(TER1, 30)`

Contenu : ∅

Oracle :

```
(¬isPartieTerminee(MJ1) ∧ isDeplacementHeroPossible(MJ1, DROITE))
∧ ∃ MJ2, MJ2 = deplacerHero(MJ1, DROITE)
```

Cas de test `MoteurJeu_deplacerHero_pre_false1` :

Préambule : `MJ1 = init(TER1, 30)`

Contenu : ∅

Oracle :

```
¬(¬isPartieTerminee(MJ1) ∧ isDeplacementHeroPossible(MJ1, HAUT))
∧ ∄ MJ2, MJ2 = deplacerHero(MJ1, HAUT)
```

Cas de test `MoteurJeu_deplacerHero_pre_false2` :

Préambule : `MJ1 = deplacerHero(init(TER1, 1), DROITE)`

Contenu : ∅

Oracle :

```
¬(¬isPartieTerminee(MJ1) ∧ isDeplacementHeroPossible(MJ1, HAUT))
∧ ∄ MJ2, MJ2 = deplacerHero(MJ1, GAUCHE)
```

Couverture des invariants

Objectif `MoteurJeu_invariant1` (minimisation de isPartieTerminee)

Cas de test `MoteurJeu_invariant1_true` :
Préambule : `MJ = deplacerHero(init(TER1, 1), DROITE)`
Contenu : ∅
Oracle : `isPartieTerminee(MJ)`

Cas de test `MoteurJeu_invariant1_false` :
Préambule : `MJ = init(TER1, 30)`
Contenu : ∅
Oracle : ¬`isPartieTerminee(MJ)`

Objectif `MoteurJeu_invariant2` (minimisation de isPartieGagnee)

Cas de test `MoteurJeu_invariant2_true` :
Préambule :

```
MJ =
    fairePasDeMiseAJour(deplacerHero(
        fairePasDeMiseAJour(deplacerHero(
            fairePasDeMiseAJour(deplacerHero(init(TER1, 30), DROITE)),
        DROITE)),
    BAS))
```

Contenu : ∅
Oracle : `isPartieGagnee(MJ)`

Cas de test `MoteurJeu_invariant2_false` :
Préambule : `MJ = init(TER1, 30)`
Contenu : ∅
Oracle : ¬`isPartieGagnee(MJ)`

Objectif `MoteurJeu_invariant3` (minimisation de isDeplacementHeroPossible)

Cas de test `MoteurJeu_invariant3_conseq` :
Préambule : `MJ = init(TER1, 30)`
Contenu : ∅
Oracle : `isDeplacementHeroPossible(MJ, DROITE)`

Cas de test `MoteurJeu_invariant3_alt` :
Préambule : `MJ = init(TER1, 30)`
Contenu : ∅
Oracle : `isDeplacementHeroPossible(MJ, BAS)`

Couverture des postconditions

Objectif `MoteurJeu_init_post1` (postcondition de init sur getPasRestants)

Cas de test `MoteurJeu_init_post1` :

Préambule : ∅

Contenu : `MJ = init(TER1, 30)`

Oracle : `getPasRestants(MJ) = 30`

Objectif `MoteurJeu_init_post2` (postcondition de init sur getTerrain)

Cas de test `MoteurJeu_init_post2` :

Préambule : ∅

Contenu : `MJ = init(TER1, 30)`

Oracle : `getTerrain(MJ) = TER1`

Objectif `MoteurJeu_deplacerHero_post1` (postcondition de deplacerHero sur getPasRestants)

Cas de test `MoteurJeu_deplacerHero_post1` :

Préambule : `MJ1 = init(TER1, 30)`

Contenu : `MJ2 = deplacerHero(MJ, DROITE)`

Oracle : `getPasRestants(MJ2) = 30 - 1`

Objectif `MoteurJeu_deplacerHero_post2` (postcondition de deplacerHero sur getTerrain)

Cas de test `MoteurJeu_deplacerHero_post2_1` :

Préambule : `MJ1 = init(TER1, 30)`

Contenu : `MJ2 = deplacerHero(MJ, DROITE)`

Oracle :

```
getTerrain(MJ2) =
    Terrain::deplacerBlocVersDirection(TER1, Terrain::getBlocHero(TER1), DROITE)
```

Cas de test `MoteurJeu_deplacerHero_post2_2` :

Préambule : `MJ = init(TER1, 30)`

Contenu : `MJ2 = deplacerHero(MJ, GAUCHE)`

Oracle :

```
let* blocHero = Terrain::getBlocHero(TER1)
and blocDest = Terrain::getBlocVersDirection(TER1, blocHero, GAUCHE)
in
    getTerrain(MJ2) = Terrain::deplacerBlocVersDirection(
        Terrain::deplacerBlocVersDirection(TER1, blocDest, GAUCHE),
            blocHero, GAUCHE)
```

Couverture des transitions

Objectif `MoteurJeu_deplacerHero_trans` (transition de deplacerHero)

Cas de test `MoteurJeu_deplacerHero_trans1` :

Préambule : `MJ1 = init(TER1, 30)`

Contenu : `MJ2 = deplacerHero(MJ, DROITE)`

Oracle :

```
getPasRestants(MJ2) = 30 - 1
∧ getTerrain(MJ2) = Terrain::deplacerBlocVersDirection(TER1,
    Terrain::getBlocHero(TER1), DROITE)
```

Cas de test `MoteurJeu_deplacerHero_trans2` :

Préambule : `MJ = init(TER1, 30)`

Contenu : `MJ2 = deplacerHero(MJ, GAUCHE)`

Oracle :

```
let* blocHero = Terrain::getBlocHero(TER1)
and blocDest = Terrain::getBlocVersDirection(TER1, blocHero, GAUCHE)
in
    getPasRestants(MJ2) = 30 - 1
    ∧ getTerrain(MJ2) = Terrain::deplacerBlocVersDirection(
        Terrain::deplacerBlocVersDirection(TER1, blocDest, GAUCHE),
            blocHero, GAUCHE)
```

6 Scénarios utilisateurs

Objectif `Scenario1_PartieGagnee` (récupérer le diamant et rejoindre la sortie)

1. Récupérer le diamant
2. Mettre à jour le terrain
3. Rejoindre la sortie
4. Mettre à jour le terrain
5. Constater la victoire

Cas de test `Scenario1_PartieGagnee` :

Préambule : `MJ = init(TER1, 30)`

Contenu :

1. `MJ2 = deplacerHero(deplacerHero(MJ, DROITE), DROITE))`
2. `T2 = fairePasDeMiseAJour(getTerrain(MJ2))`
3. `MJ3 = deplacerHero(MJ2, BAS)`
4. `T3 = fairePasDeMiseAJour(getTerrain(MJ3))`

Oracle :

1. Préconditions avant MJ2
2. Postconditions + invariant après MJ2
3. Postconditions + invariant après T2
4. Préconditions avant MJ3
5. Postconditions + invariant après MJ3
6. Postconditions + invariant après T3
7. `isPartieGagnee(MJ)`

Objectif `Scenario2_PartiePerdue_ManqueDePas` (se déplacer jusqu'à épuiser le nombre de pas et ainsi perdre la partie)

1. Se déplacer jusqu'à épuiser le nombre de pas
2. Mettre à jour le terrain
3. Constater la défaite

Cas de test `Scenario2_PartiePerdue_ManqueDePas` :

Préambule : `MJ = init(TER1, 2)`

Contenu :

1. `MJ2 = deplacerHero(deplacerHero(MJ, DROITE), DROITE)`
2. `T2 = fairePasDeMiseAJour(getTerrain(MJ2))`

Oracle :

1. Préconditions avant MJ2
2. Postconditions + invariant après MJ2
3. Postconditions + invariant après T2
4. `isPartieTerminee(MJ)` $\wedge$ $\neg$`isPartieGagnee(MJ)`

Objectif `Scenario3_PartiePerdue_MortDuHero` (se déplacer sous le rocher, puis mourir)

1. Se déplacer sous le rocher
2. Mettre à jour le terrain
3. Constater la mort du héro

Cas de test `Scenario3_PartiePerdue_MortDuHero` :

Préambule : `MJ = init(TER1, 30)`

Contenu :

1. `MJ2 = deplacerHero(deplacerHero(MJ, BAS), GAUCHE)`
2. `T2 = fairePasDeMiseAJour(getTerrain(MJ2))`

Oracle :

1. Préconditions avant MJ2
2. Postconditions + invariant après MJ2
3. Postconditions + invariant après T2
4. ¬`isHeroVivant(T2)`

Objectif `Scenario4_DeplacerUnRocher` (pousser le rocher vers la gauche)

1. Pousser le rocher vers la gauche
2. Constater le déplacement du rocher
3. Mettre à jour le terrain

Cas de test `Scenario4_DeplacerUnRocher` :

Préambule : `MJ = init(TER1, 30)`

Contenu :

1. `MJ2 = deplacerHero(MJ, GAUCHE)`
2. `T2 = fairePasDeMiseAJour(getTerrain(MJ2))`

Oracle :

1. Préconditions avant MJ2
2. Postconditions + invariant après MJ2
3. `getType(getBloc(getTerrain(MJ2), 0, 1)) = ROCHER`
4. Postconditions + invariant après T2