Dungeon by Contract

SERGEANT William
HA-VINH Vincent

1 juin 2018
Master STL

Sommaire

- Introduction

1. Points importants
   - Spécifications
   - Contrats : step
   - Test : MapService

2. Présentation de Hit
   - Spécifications
   - Contrats
   - Tests MBT (paires de transitions)

3. Génération de cartes

- Démonstration

1.Spécifications

EditMap manipule la nature,

Environment manipule le contenu.

Service: Environment

Constructor:
init: Map → [Environment]

Operators:
SetContent: [Environment] × int × int × Mob → [Environment]
pre SetContent(M,x,y,C) **requires** CellNature(M,x,y) ∉ {**WLL**, **DNC**, **DWC**}
and C ≠ **No implies** CellContent(M,x,y) = **No**

Observations:
[SetContent]

- CellContent(SetContent(M,x,y,C),x,y) = C
- **forall** u,v **in** int², u ≠ x **or** v ≠ y
  implies CellContent(SetContent(M,x,y, C),u,v) = CellContent(M,u,v)

1.Contrats : step

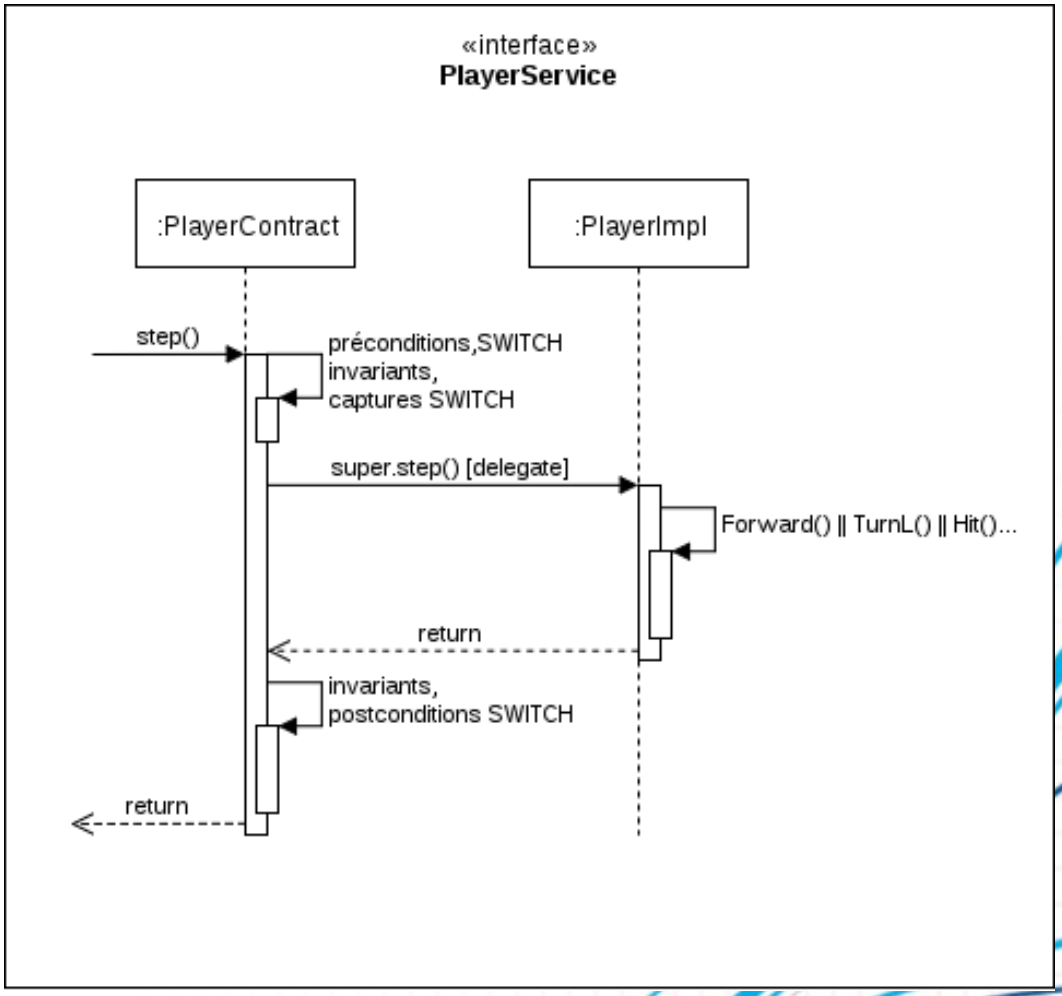
Dans PlayerService :

[step]:

- LastCom(P)=**FF implies** step(P) = Forward(P)
- LastCom(P)=**BB implies** step(P) = Backward(P)
- LastCom(P)=**LL implies** step(P) = StrafeLeft(P)
- LastCom(P)=**RR implies** step(P) = StrafeRight(P)
- LastCom(P)=**TL implies** step(P) = TurnLeft(P)
- LastCom(P)=**TR implies** step(P) = TurnRight(P)
- LastCom(P)=**HIT implies** step(P) = Hit(P)

Le step de joueur appelle des opérateurs de son propre service

1.Contrats : step



1.Contrats : step

```
@Override
    public void step() {
        //pre

        //inv pre
        checkInvariants();

        //capture
        Command lastCom_atpre = getLastCom();

        /*captures des methodes Forward Strafe Turn ....*/
        /*captures de hit*/

       //run
       super.step();

       //post
              switch(lastCom_atpre) {
              case FF:
                     //post forward
             ...
}
```

1.Test : MapService

MapService :

- pas d'invariants → comportement indéterminé → besoin de configuration → EditMap

```
public void beforeTest() {
        EditMapService editmap = new EditMapImpl();
        setEditMap(editmap);
        setMap(new MapContract(editmap));
}
public void preOpenDoorPositif() {
        //init
        editmap.init(14, 35);
        editmap.setNature(13, 34,Cell.DWC);
        //operation
        try {   map.openDoor(13, 34); }
        //oracle
        catch(PreconditionError e) {
                fail(e.toString());
        }
```

2. Hit - Spécifications

Pour EntityService:

Operator:
hit: [Entity] → [Entity]

Pour PlayerService:

Observations:
[Hit]

- Face(M)=**N** **implies**
  - Environment::CellNature(Envi(M),Col(M),Row(M)+1) = **DWO** **and** Environment::CellContent(Envi(M),Col(M),Row(M)+1) = **No** **implies** CloseDoor(Envi(M),Col(M),Row(M)+1)
  - Environment::CellNature(Envi(M),Col(M),Row(M)+1) = **DWC** **implies** OpenDoor(Envi(M),Col(M),Row(M)+1)
  - Environment::CellContent(Envi(M),Col(M),Row(M)+1) ≠ **No** **implies** takeHit(CellContent(Envi(M),Col(M),Row(M)+1))
  - Environment::CellNature(Envi(M),Col(M),Row(M)+1) ∉ {**DWO**, **DWC**} **or** Environment::CellContent(Envi(M),Col(M),Row(M)+1) = **No** **implies** Envi(Hit(P)) = Envi(P)

De même pour les autres directions (en changeant les coordonnées)
(W et E fonctionnent avec DNO et DNC)

2. Hit - Contrats

```
public void Hit(){
        //pre
        //inv
        checkinvariant();
        //capture
        La case devant le joueur selon sa direction
        Le nombre de HP du monstre dans la case s'il y en a un
        //run
        super.hit();
        //inv
        checkinvariant();
        //post
        Switch(direction du joueur)
                case N:
                        si DNO et pas de mob => postconditions de CloseDoor
                        si DNC => postconditions de OpenDoor
                        si monstre => postcondition : le monstre a perdu 1 HP
                        sinon => Nature&Content(getEnv(Hit(J),Case devant Joueur)
                                        inchangée
        ...
```

2. Hit - Tests MBT

Paires de transition:

C.I: editmap.init(14,35);
editmap.setNature(7, 23, Cell.EMP);
editmap.setNature(7, 24, Cell.DWO);
env.init(editmap);
player.init(env, 7, 23, Dir.N, 10);
cow.init(env, 7, 24, Dir.E, 3);
env.setContent(7,24, cow);
env.setContent(7,23, player);

operations:
player.hit(); //hp de la vache -1
player.hit(); //hp de la vache -1
player.hit(); //mort de la vache
player.hit(); //ferme porte

2. Hit - Tests MBT

oracle:

```
assertInv();                    //assertTrue(true);
assertTrue(map.getCellNature(7, 24) == Cell.DWC);
assertTrue(map.getCellContent(7, 24) == null);

for(int x=0; x<14; x++)
for(int y=0; y<35; y++)
    if( (x!=10 && y!=10) && (x!=13 && y!=34)) {
        assertTrue(map.getCellNature(x, y) == cellsNature_atpre[y][x]);
        asserTrue(map.getCellContent(x, y) == cellsContent_atpre[y][x]);
    }
```

3. Génération de cartes

Algorithme du marcheur ivre

Quatre paramètres :

- Largeur de la carte → width
- Hauteur de la carte → heigth
- Nombre maximal de couloirs → max_cor
- Longueur maximale des couloirs → max_len

Exécution:

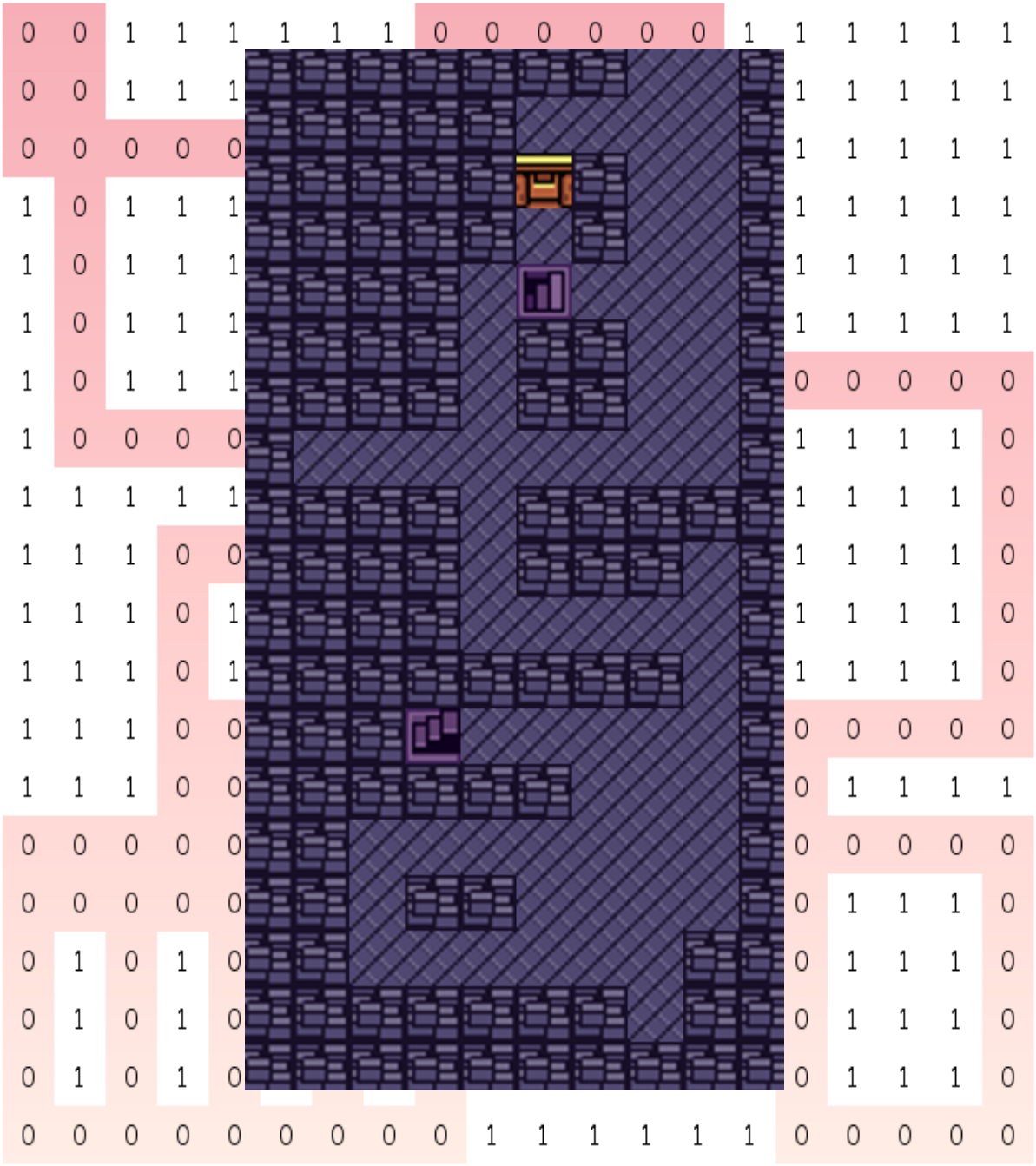
1. Remplir la carte avec des cellules WLL
2. Choisir aléatoirement un point de départ → Case IN
3. Choisir aléatoirement une direction → Dir
4. Choisir aléatoirement une valeur entre 1 et max_len → N
5. Avancer de N cases vers Dir
6. Décrémenter max_cor
7. Répéter depuis l'étape 3 tant que max_cor > 0
8. Si max_cor == 0 → Case OUT = position actuelle

Puis placement des portes et des monstres.

DIMENSIONS 20 MAXTUNNELS 40 MAXLENGTH 8

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

DIMENSIONS 20 MAXTUNNELS 40 MAXLENGTH 8




SORBONNE
UNIVERSITÉ
CRÉATEURS DE FUTURS
DEPUIS 1257
Questions ?

Démonstration