

Rapport de CPS

Conception par contrat

d'un Dungeon Master

Amel ARKOUB 3301571

Ling-Chun SO 3414546

Descriptif du projet	3
Manuel d'utilisation	4
Executer le projet	4
Comment jouer	4
Spécifications	8
Types basiques	8
Services	8
Service: Map	8
Service: EditMap refines Map	9
Service: Environment includes EditMap	10
Service: Mob	10
Service: Entity includes Mob	14
Service: Cow includes Entity	15
Service: Player includes Entity	16
Service: Engine	17
Rapport de projet	19
Choix d'implantation et difficultés de spécification	19
Test pertinents	21

Descriptif du projet

Le but du projet est d'implanter un jeu similaire à Dungeon Master dont le cahier des charges a été partiellement décrit. Il s'agit d'implanter cette spécification selon la méthode Design-by-Contract et d'écrire une implantation du jeu.

L'archive fournit se compose de:

- Ce rapport
- Des sources du projet

Les sources du projets sont découpés dans des packages qui contiennent donc:

Dungeon.master.components → L'implantation des composants qui spécifie les contrats

Dungeon.master.components.bug → L'implantation des composants qui ne vérifie pas les contrats

Dungeon.master.contracts → L'implantation des contrats

Dungeon.master.decorators → L'implantation des decorateurs

Dungeon.master.enumerations → Les types de bases qui sont des énumérations Java

Dungeon.master.exceptions → Les exceptions des préconditions, postconditions et invariants

Dungeon.master.mbt.test → L'implantation des tests MBT

Dungeon.master.services → Les interfaces services

Dungeon.master.ui → L'implantation de l'interface graphique

Dungeon.master.ui.implementations → Des decorateurs permettant de faire le lien entre l'interface et l'implantation

Manuel d'utilisation

Executer le projet

Afin de faciliter l'utilisation du projet, nous avons utilisé ant dont les différentes commandes sont les suivantes:

- Ant run → Lance le jeu dungeon master (Dans le cas où il y a une erreur de compilation, une alternative serait d'importer le projet dans Eclipse et de lancer `Dungeon.master.ui.Main.java`)
- Ant dist → Génère une distribution
- Ant test → Lance tout les tests junit (Le fichier lancé est `dungeon.master.mbt.test.RunAllTests`)

Comment jouer

Pour se déplacer, utiliser les flèches du pad. (Il est à noter, qu'à chaque changement de direction les flèches du pad sont rebinds afin d'obtenir un mouvement intuitif)

Pour tourner à gauche, appuyer sur S.

Pour tourner à droite, appuyer sur D.

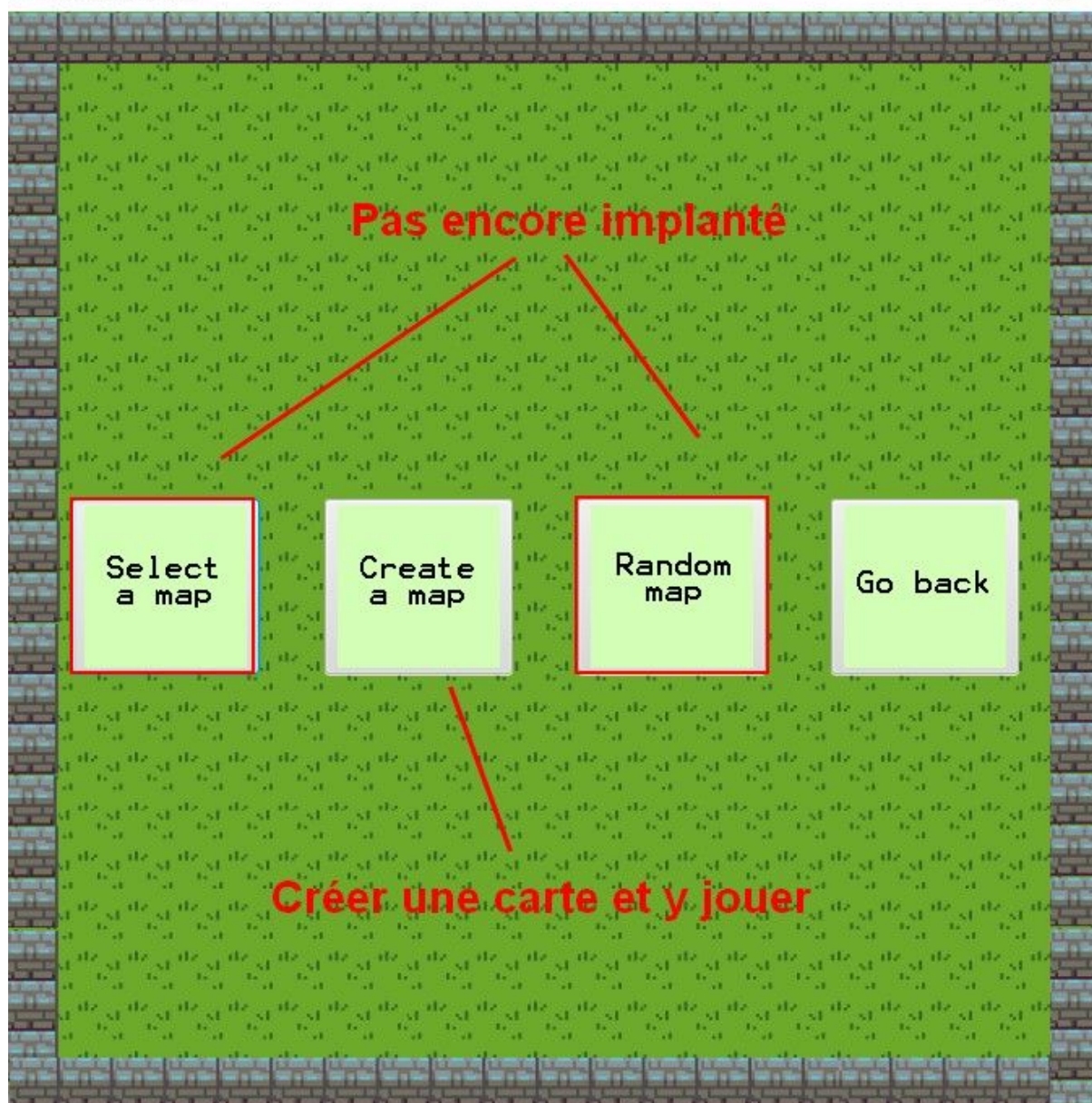
Pour ouvrir une porte, appuyer sur Z.

Pour fermer une porte, appuyer sur E.

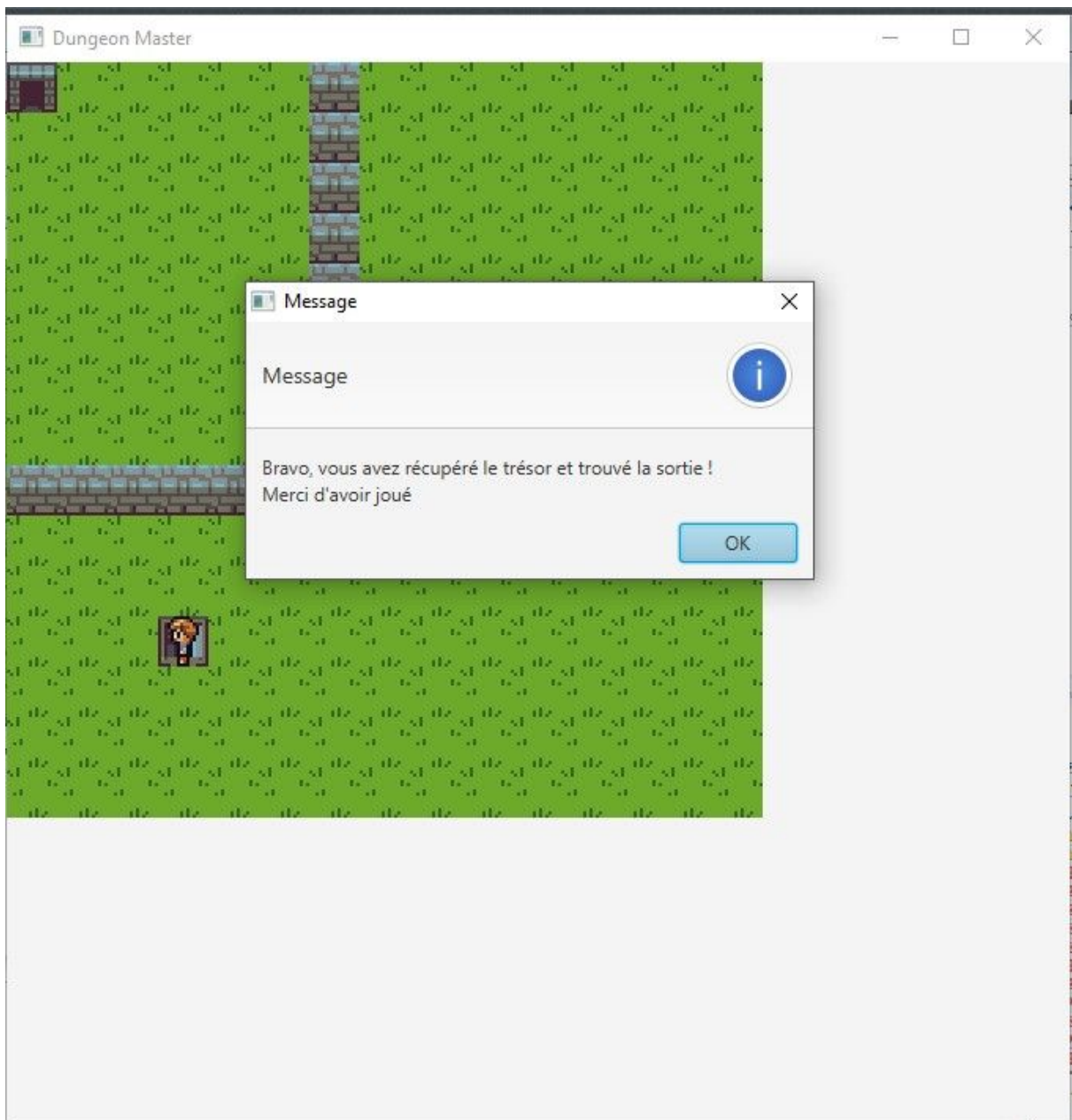
Pour attaquer un ennemi, appuyer sur ESPACE. (attaque en face du joueur)

Pour prendre le trésor, appuyer sur T. (prend le trésor en face du joueur)

Voici des images avec des légendes afin de mieux comprendre les fonctionnalités du jeu:







Spécifications

Types basiques

Type Cell {IN, OUT, EMP, WLL, DNO, DNC, DWO, DWC, TRS}

Type Dir {N, S, W, E}

Type Command {FF, BB, RR, LL, TL, TR, NONE}

Type Option[T] {No, So(T)}

Type Set[T]

Type Array[T1,...,TN]

Services

Service: Map

Types: bool, int, Cell

Observers:

const Height: [Map] \rightarrow int

const Width: [Map] \rightarrow int

CellNature: [Map] x int x int \rightarrow Cell

Pre CellNature(M,x,y) requires $0 \leq x < \text{Width}(M)$ and $0 \leq y < \text{Height}(M)$

Constructors:

init: int x int \rightarrow [Map]

Pre init(w,h) requires $0 < w$ and $0 < h$

Operators:

OpenDoor: [Map] x int x int \rightarrow [Map]

Pre OpenDoor(M,x,y) requires CellNature(M,x,y) \in {DNC, DWC}

CloseDoor: [Map] x int x int \rightarrow [Map]

Pre CloseDoor(M,x,y) requires CellNature(M,x,y) \in {DNO, DWO}

Observation:

[Invariant]: T

[init]:

Height(init(h,w)) = h

Width(init(h,w)) = w

[OpenDoor]:

CellNature(M,x,y) = DWC \Rightarrow CellNature(OpenDoor(M,x,y),x,y) = DWO

CellNature(M,x,y) = DNC \Rightarrow CellNature(OpenDoor(M,x,y),x,y) = DNO

Forall $u \in [0; \text{Width}(M)-1]$ forall $v \in [0; \text{Height}(M)-1]$ ($u \neq x$ or $v \neq y$) \Rightarrow
 $\text{CellNature}(\text{OpenDoor}(M, x, y), u, v) = \text{CellNature}(M, u, x)$
 [CloseDoor]:
 $\text{CellNature}(M, x, y) = \text{DWO} \Rightarrow \text{CellNature}(\text{OpenDoor}(M, x, y), x, y) = \text{DWC}$
 $\text{CellNature}(M, x, y) = \text{DNO} \Rightarrow \text{CellNature}(\text{OpenDoor}(M, x, y), x, y) = \text{DNC}$
 Forall $u \in [0; \text{Width}(M)-1]$ forall $v \in [0; \text{Height}(M)-1]$ ($u \neq x$ or $v \neq y$) \Rightarrow
 $\text{CellNature}(\text{OpenDoor}(M, x, y), u, v) = \text{CellNature}(M, u, x)$

Service: EditMap refines Map

Types: bool, int, Cell

Observers:

isReachable: [EditMap] x int x int x int x int \rightarrow [EditMap]

Pre isReachable(M, x_1, y_1, x_2, y_2) requires $\text{CellNature}(M, x_1, y_1) \neq \text{WLL}$ and
 $\text{CellNature}(M, x_2, y_2) \neq \text{WLL}$

isReady: [EditMap] \rightarrow bool

Constructors: NONE

Operators:

SetNature: [EditMap] x int x int x Cell \rightarrow [EditMap]

Pre SetNature(M, x, y) requires $0 \leq x < \text{Width}(M)$ and $0 \leq y < \text{Height}(M)$

Observation:

[Invariant]:

$\text{IsReachable}(M, x_1, y_1, x_2, y_2) = \text{exists } P \text{ in Array}[\text{int}, \text{int}], P[0] = (x_1, y_1) \text{ and}$
 $P[\text{size}(P)-1] = (x_2, y_2) \text{ and forall } i \text{ in } [1; \text{size}(P)-1], (P[i-1] = (u, v) \text{ and } P[i] = (s, t)) \Rightarrow (u-s)^2 + (v-t)^2 =$
 $1 \text{ and forall } i \text{ in } [1; \text{size}(P)-2], P[i-1] = (u, v) \Rightarrow \text{CellNature}(M, u, v) \neq \text{WLL}$

$\text{isReady}(M) = \text{exists } x_i, y_i = \text{IN} \text{ and } \text{CellNature}(M, x_i, y_i) = \text{OUT}$
 $\text{and } \text{isReachable}(M, x_i, y_i, x_i, y_i)$

and forall x, y in int^2 , $x \neq x_i$ or $y \neq y_i \Rightarrow \text{CellNature}(M, x, y) \neq \text{IN}$

and forall x, y in int^2 , $x \neq x_i$ or $y \neq y_i \Rightarrow \text{CellNature}(M, x, y) \neq \text{OUT}$

forall x, y in int, $\text{CellNature}(M, x, y) \in \{\text{DNO}, \text{DNC}\} \Rightarrow$

$\text{CellNature}(M, x+1, y) = \text{CellNature}(M, x-1, y) = \text{EMP}$ and

$\text{CellNature}(M, x, y-1) = \text{CellNature}(M, x, y+1) = \text{WLL}$

forall x, y in int, $\text{CellNature}(M, x, y) \in \{\text{DWO}, \text{DWC}\} \Rightarrow$

$\text{CellNature}(M, x+1, y) = \text{CellNature}(M, x-1, y) = \text{EMP}$ and

$\text{CellNature}(M, x, y-1) = \text{CellNature}(M, x, y+1) = \text{WLL}$

[SetNature]:

$\text{CellNature}(\text{SetNature}(M, x, y, \text{Na}), x, y) = \text{Na}$

forall u, v in int^2 , $u \neq x$ or $v \neq y \Rightarrow \text{CellNature}(\text{SetNature}(M, x, y), u, v) =$
 $\text{CellNature}(M, u, v)$

Service: Environment includes EditMap

Types: bool, int, Cell, Entity

Observers: CellContent: $\text{int} \times \text{int} \rightarrow \text{Option}[\text{Entity}]$

Operators: CloseDoor: $[\text{Environment}] \times \text{int} \times \text{int} \rightarrow [\text{Environment}]$

Pre CloseDoor(M,x,y) requires CellContent(M,x,y) = No

Service: Mob

Types: bool, int, Cell

Observers:

Env: $[\text{Mob}] \rightarrow \text{Environment}$

Col: $[\text{Mob}] \rightarrow \text{int}$

Row: $[\text{Mob}] \rightarrow \text{int}$

Face: $[\text{Mob}] \rightarrow \text{Dir}$

Constructors:

Init: $\text{Environment} \times \text{int} \times \text{int} \times \text{Dir} \rightarrow [\text{Mob}]$

Pre init(E,x,y,D) requires $0 \leq x < \text{Environment}::\text{Width}(E)$ and $0 \leq y <$

$\text{Environment}::\text{Height}(E)$

Operators:

Forward: $[\text{Mob}] \rightarrow [\text{Mob}]$

Backward: $[\text{Mob}] \rightarrow [\text{Mob}]$

TurnL: $[\text{Mob}] \rightarrow [\text{Mob}]$

TurnR: $[\text{Mob}] \rightarrow [\text{Mob}]$

StrafeL: $[\text{Mob}] \rightarrow [\text{Mob}]$

StrafeR: $[\text{Mob}] \rightarrow [\text{Mob}]$

[Invariant]:

$0 \leq \text{Col}(M) < \text{Environment}::\text{Width}(\text{Envi}(M))$

$0 \leq \text{Row}(M) < \text{Environment}::\text{Height}(\text{Envi}(M))$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)) \in \{\text{WLL}, \text{FNC}, \text{DWC}\}$

[init]:

$\text{Col}(\text{init}(E, x, y, D)) = x$

$\text{Row}(\text{init}(E, x, y, D)) = y$

$\text{Face}(\text{init}(E, x, y, D)) = D$

$\text{Envi}(\text{init}(E, x, y, D)) = E$

Observation:

[Forward]:

Aucun changement dans Face(M)

$\text{Face}(M) = N \Rightarrow$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)+1) \in \{\text{EMP}, \text{DNO}\}$

and $\text{Row}(M)+1 < \text{Environment}::\text{Height}(\text{Envi}(M))$
 and $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)+1) = \text{No}$
 $\Rightarrow \text{Row}(\text{Forward}(M)) = \text{Row}(M)+1$ and $\text{Col}(\text{Forward}(M)) = \text{Col}(M)$

$\text{Face}(M) = \text{N} \Rightarrow$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)+1) \notin \{\text{EMP}, \text{DNO}\}$
 or $\text{Row}(M)+1 \geq \text{Environment}::\text{Height}(\text{Envi}(M))$
 or $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)+1) \neq \text{No}$
 $\Rightarrow \text{Row}(\text{Forward}(M)) = \text{Row}(M)$ and $\text{Col}(\text{Forward}(M)) = \text{Col}(M)$

$\text{Face}(M) = \text{S} \Rightarrow$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)-1) \in \{\text{EMP}, \text{DNO}\}$
 and $\text{Row}(M)-1 \geq 0$
 and $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)-1) = \text{No}$
 $\Rightarrow \text{Row}(\text{Forward}(M)) = \text{Row}(M)-1$ and $\text{Col}(\text{Forward}(M)) = \text{Col}(M)$

$\text{Face}(M) = \text{S} \Rightarrow$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)-1) \notin \{\text{EMP}, \text{DNO}\}$
 or $\text{Row}(M)-1 < 0$
 or $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)+1) \neq \text{No}$
 $\Rightarrow \text{Row}(\text{Forward}(M)) = \text{Row}(M)$ and $\text{Col}(\text{Forward}(M)) = \text{Col}(M)$

$\text{Face}(M) = \text{E} \Rightarrow$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M)+1, \text{Row}(M)) \in \{\text{EMP}, \text{DWO}\}$
 and $\text{Col}(M)+1 < \text{Environment}::\text{Width}(\text{Envi}(M))$
 and $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M)+1, \text{Row}(M)) = \text{No}$
 $\Rightarrow \text{Row}(\text{Forward}(M)) = \text{Row}(M)$ and $\text{Col}(\text{Forward}(M)) = \text{Col}(M)+1$

$\text{Face}(M) = \text{E} \Rightarrow$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M)+1, \text{Row}(M)) \notin \{\text{EMP}, \text{DWO}\}$
 or $\text{Col}(M)+1 \geq \text{Environment}::\text{Width}(\text{Envi}(M))$
 or $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M)+1, \text{Row}(M)) \neq \text{No}$
 $\Rightarrow \text{Row}(\text{Forward}(M)) = \text{Row}(M)$ and $\text{Col}(\text{Forward}(M)) = \text{Col}(M)$

$\text{Face}(M) = \text{W} \Rightarrow$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M)-1, \text{Row}(M)) \in \{\text{EMP}, \text{DWO}\}$
 and $\text{Col}(M)-1 \geq 0$
 and $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M)-1, \text{Row}(M)) = \text{No}$
 $\Rightarrow \text{Row}(\text{Forward}(M)) = \text{Row}(M)$ and $\text{Col}(\text{Forward}(M)) = \text{Col}(M)-1$

$\text{Face}(M) = \text{W} \Rightarrow$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M)-1, \text{Row}(M)) \notin \{\text{EMP}, \text{DWO}\}$
 or $\text{Col}(M)-1 < 0$
 or $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M)-1, \text{Row}(M)) \neq \text{No}$
 $\Rightarrow \text{Row}(\text{Forward}(M)) = \text{Row}(M)$ and $\text{Col}(\text{Forward}(M)) = \text{Col}(M)$

[Backward]:

Aucun changement dans $\text{Face}(M)$

$\text{Face}(M) = \text{S} \Rightarrow$

$\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)+1) \in \{\text{EMP}, \text{DNO}\}$
 and $\text{Row}(M)+1 < \text{Environment}::\text{Height}(\text{Envi}(M))$
 and $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)+1) = \text{No}$
 $\Rightarrow \text{Row}(\text{Backward}(M)) = \text{Row}(M)+1$ and $\text{Col}(\text{Backward}(M)) = \text{Col}(M)$

$\text{Face}(M) = \text{S} \Rightarrow$

Environment::CellNature(Envi(M),Col(M),Row(M)+1) \notin {EMP,DNO}
or Row(M)+1 \geq Environment::Height(Envi(M))
or Environment::CellContent(Envi(M),Col(M),Row(M)+1) \neq No
 \Rightarrow Row(Backward(M)) = Row(M) and Col(Backward(M)) = Col(M)

Face(M) = N \Rightarrow

Environment::CellNature(Envi(M),Col(M),Row(M)-1) \in {EMP,DNO}
and Row(M)-1 \geq 0
and Environment::CellContent(Envi(M),Col(M),Row(M)-1) = No
 \Rightarrow Row(Backward(M)) = Row(M)-1 and Col(Backward(M)) = Col(M)

Face(M) = N \Rightarrow

Environment::CellNature(Envi(M),Col(M),Row(M)-1) \notin {EMP,DNO}
or Row(M)-1 < 0
or Environment::CellContent(Envi(M),Col(M),Row(M)+1) \neq No
 \Rightarrow Row(Backward(M)) = Row(M) and Col(Backward(M)) = Col(M)

Face(M) = W \Rightarrow

Environment::CellNature(Envi(M),Col(M)+1,Row(M)) \in {EMP,DWO}
and Col(M)+1 < Environment::Width(Envi(M))
and Environment::CellContent(Envi(M),Col(M)+1,Row(M)) = No
 \Rightarrow Row(Backward(M)) = Row(M) and Col(Backward(M)) = Col(M)+1

Face(M) = W \Rightarrow

Environment::CellNature(Envi(M),Col(M)+1,Row(M)) \notin {EMP,DWO}
or Col(M)+1 \geq Environment::Width(Envi(M))
or Environment::CellContent(Envi(M),Col(M)+1,Row(M)) \neq No
 \Rightarrow Row(Backward(M)) = Row(M) and Col(Backward(M)) = Col(M)

Face(M) = E \Rightarrow

Environment::CellNature(Envi(M),Col(M)-1,Row(M)) \in {EMP,DWO}
and Col(M)-1 \geq 0
and Environment::CellContent(Envi(M),Col(M)-1,Row(M)) = No
 \Rightarrow Row(Backward(M)) = Row(M) and Col(Backward(M)) = Col(M)-1

Face(M) = E \Rightarrow

Environment::CellNature(Envi(M),Col(M)-1,Row(M)) \notin {EMP,DWO}
or Col(M)-1 < 0
or Environment::CellContent(Envi(M),Col(M)-1,Row(M)) \neq No
 \Rightarrow Row(Backward(M)) = Row(M) and Col(Backward(M)) = Col(M)

[StrafeL]:

Aucun changement dans Face(M)

Face(M) = E \Rightarrow

Environment::CellNature(Envi(M),Col(M),Row(M)+1) \in {EMP,DNO}
and Row(M)+1 < Environment::Height(Envi(M))
and Environment::CellContent(Envi(M),Col(M),Row(M)+1) = No
 \Rightarrow Row(StrafeL(M)) = Row(M)+1 and Col(StrafeL(M)) = Col(M)

Face(M) = E \Rightarrow

Environment::CellNature(Envi(M),Col(M),Row(M)+1) \notin {EMP,DNO}
or Row(M)+1 \geq Environment::Height(Envi(M))
or Environment::CellContent(Envi(M),Col(M),Row(M)+1) \neq No
 \Rightarrow Row(StrafeL(M)) = Row(M) and Col(StrafeL(M)) = Col(M)

Face(M) = W \Rightarrow

Environment::CellNature(Envi(M),Col(M),Row(M)-1) \in {EMP,DNO}
and Row(M)-1 \geq 0
and Environment::CellContent(Envi(M),Col(M),Row(M)-1) = No
 \Rightarrow Row(StrafeL(M)) = Row(M)-1 and Col(StrafeL(M)) = Col(M)

Face(M) = W \Rightarrow

Environment::CellNature(Envi(M),Col(M),Row(M)-1) \notin {EMP,DNO}
or Row(M)-1 < 0
or Environment::CellContent(Envi(M),Col(M),Row(M)+1) \neq No
 \Rightarrow Row(StrafeL(M)) = Row(M) and Col(StrafeL(M)) = Col(M)

Face(M) = S \Rightarrow

Environment::CellNature(Envi(M),Col(M)+1,Row(M)) \in {EMP,DWO}
and Col(M)+1 < Environment::Width(Envi(M))
and Environment::CellContent(Envi(M),Col(M)+1,Row(M)) = No
 \Rightarrow Row(StrafeL(M)) = Row(M) and Col(StrafeL(M)) = Col(M)+1

Face(M) = S \Rightarrow

Environment::CellNature(Envi(M),Col(M)+1,Row(M)) \notin {EMP,DWO}
or Col(M)+1 \geq Environment::Width(Envi(M))
or Environment::CellContent(Envi(M),Col(M)+1,Row(M)) \neq No
 \Rightarrow Row(StrafeL(M)) = Row(M) and Col(StrafeL(M)) = Col(M)

Face(M) = N \Rightarrow

Environment::CellNature(Envi(M),Col(M)-1,Row(M)) \in {EMP,DWO}
and Col(M)-1 \geq 0
and Environment::CellContent(Envi(M),Col(M)-1,Row(M)) = No
 \Rightarrow Row(StrafeL(M)) = Row(M) and Col(StrafeL(M)) = Col(M)-1

Face(M) = N \Rightarrow

Environment::CellNature(Envi(M),Col(M)-1,Row(M)) \notin {EMP,DWO}
or Col(M)-1 < 0
or Environment::CellContent(Envi(M),Col(M)-1,Row(M)) \neq No
 \Rightarrow Row(StrafeL(M)) = Row(M) and Col(StrafeL(M)) = Col(M)

[StrafeR]:

Aucun changement dans Face(M)

Face(M) = W \Rightarrow

Environment::CellNature(Envi(M),Col(M),Row(M)+1) \in {EMP,DNO}
and Row(M)+1 < Environment::Height(Envi(M))
and Environment::CellContent(Envi(M),Col(M),Row(M)+1) = No
 \Rightarrow Row(StrafeR(M)) = Row(M)+1 and Col(StrafeR(M)) = Col(M)

Face(M) = W \Rightarrow

Environment::CellNature(Envi(M),Col(M),Row(M)+1) \notin {EMP,DNO}
or Row(M)+1 \geq Environment::Height(Envi(M))
or Environment::CellContent(Envi(M),Col(M),Row(M)+1) \neq No
 \Rightarrow Row(StrafeR(M)) = Row(M) and Col(StrafeR(M)) = Col(M)

Face(M) = E \Rightarrow

Environment::CellNature(Envi(M),Col(M),Row(M)-1) \in {EMP,DNO}
and Row(M)-1 \geq 0
and Environment::CellContent(Envi(M),Col(M),Row(M)-1) = No

$\Rightarrow \text{Row}(\text{StrafeR}(M)) = \text{Row}(M)-1$ and $\text{Col}(\text{StrafeR}(M)) = \text{Col}(M)$
 $\text{Face}(M) = E \Rightarrow$
 $\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)-1) \notin \{\text{EMP}, \text{DNO}\}$
or $\text{Row}(M)-1 < 0$
or $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M), \text{Row}(M)+1) \neq \text{No}$
 $\Rightarrow \text{Row}(\text{StrafeR}(M)) = \text{Row}(M)$ and $\text{Col}(\text{StrafeR}(M)) = \text{Col}(M)$
 $\text{Face}(M) = N \Rightarrow$
 $\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M)+1, \text{Row}(M)) \in \{\text{EMP}, \text{DWO}\}$
and $\text{Col}(M)+1 < \text{Environment}::\text{Width}(\text{Envi}(M))$
and $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M)+1, \text{Row}(M)) = \text{No}$
 $\Rightarrow \text{Row}(\text{StrafeR}(M)) = \text{Row}(M)$ and $\text{Col}(\text{StrafeR}(M)) = \text{Col}(M)+1$
 $\text{Face}(M) = N \Rightarrow$
 $\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M)+1, \text{Row}(M)) \notin \{\text{EMP}, \text{DWO}\}$
or $\text{Col}(M)+1 \geq \text{Environment}::\text{Width}(\text{Envi}(M))$
or $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M)+1, \text{Row}(M)) \neq \text{No}$
 $\Rightarrow \text{Row}(\text{StrafeR}(M)) = \text{Row}(M)$ and $\text{Col}(\text{StrafeR}(M)) = \text{Col}(M)$
 $\text{Face}(M) = S \Rightarrow$
 $\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M)-1, \text{Row}(M)) \in \{\text{EMP}, \text{DWO}\}$
and $\text{Col}(M)-1 \geq 0$
and $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M)-1, \text{Row}(M)) = \text{No}$
 $\Rightarrow \text{Row}(\text{StrafeR}(M)) = \text{Row}(M)$ and $\text{Col}(\text{StrafeR}(M)) = \text{Col}(M)-1$
 $\text{Face}(M) = S \Rightarrow$
 $\text{Environment}::\text{CellNature}(\text{Envi}(M), \text{Col}(M)-1, \text{Row}(M)) \notin \{\text{EMP}, \text{DWO}\}$
or $\text{Col}(M)-1 < 0$
or $\text{Environment}::\text{CellContent}(\text{Envi}(M), \text{Col}(M)-1, \text{Row}(M)) \neq \text{No}$
 $\Rightarrow \text{Row}(\text{StrafeR}(M)) = \text{Row}(M)$ and $\text{Col}(\text{StrafeR}(M)) = \text{Col}(M)$

[TurnL]:

$\text{Face}(M)=N \Rightarrow \text{Face}(\text{TurnL}(M)) = W$
 $\text{Face}(M)=W \Rightarrow \text{Face}(\text{TurnL}(M)) = S$
 $\text{Face}(M)=S \Rightarrow \text{Face}(\text{TurnL}(M)) = E$
 $\text{Face}(M)=E \Rightarrow \text{Face}(\text{TurnL}(M)) = N$

[TurnR]:

$\text{Face}(M)=N \Rightarrow \text{Face}(\text{TurnR}(M)) = E$
 $\text{Face}(M)=E \Rightarrow \text{Face}(\text{TurnR}(M)) = S$
 $\text{Face}(M)=S \Rightarrow \text{Face}(\text{TurnR}(M)) = W$
 $\text{Face}(M)=W \Rightarrow \text{Face}(\text{TurnR}(M)) = N$

Service: Entity includes Mob

Observer:

Hp: [Entity] \rightarrow int

Damage: [Entity] \rightarrow int

Constructors:

Init: Environment x int x int x Dir x int x int \rightarrow [Entity]

Pre init(E,x,y,D,h) requires $h > 0$ and $h \geq 0$ and $0 \leq x <$

Environment::Width(E) and $0 \leq y < \text{Environment::Height}(E)$

Operators:

Step: [Entity] \rightarrow [Entity]

Attack: [Entity] \rightarrow [Entity]

SetHp: [Entity] x int \rightarrow [Entity]

Pre SetHp(E,h) requires $h \geq 0$

Observations:

[init]:

Hp(init(E,x,y,D,h,d)) = h

Damage(init(E,x,y,D,h,d)) = d

[attack]:

Face(E) = N and $\text{Row}(E)+1 < \text{Height}(\text{Env}(E))$ and

$\text{CellContent}(\text{Env}(E), \text{Row}(E)+1, \text{Col}(E)) \neq \text{No} \Rightarrow$

$\text{Hp}(\text{CellContent}(\text{Env}(\text{Attack}(E)), \text{Col}(\text{Attack}(E)), \text{Row}(\text{Attack}(E))+1)) =$

$\text{Hp}(\text{CellContent}(\text{Env}(E), \text{Col}(E), \text{Row}(E)+1)) - \text{Damage}(E)$

Face(E) = S and $\text{Row}(E)-1 \geq 0$ and $\text{CellContent}(\text{Env}(E), \text{Col}(E), \text{Row}(E)-1) \neq$

$\text{No} \Rightarrow \text{Hp}(\text{CellContent}(\text{Env}(\text{Attack}(E)), \text{Col}(\text{Attack}(E)), \text{Row}(\text{Attack}(E))-1)) =$

$\text{Hp}(\text{CellContent}(\text{Env}(E), \text{Col}(E), \text{Row}(E)-1)) - \text{Damage}(E)$

Face(E) = W and $\text{Col}(E)-1 \geq 0$ and $\text{CellContent}(\text{Env}(E), \text{Col}(E)-1, \text{Row}(E)) \neq$

$\text{No} \Rightarrow \text{Hp}(\text{CellContent}(\text{Env}(\text{Attack}(E)), \text{Col}(\text{Attack}(E))-1, \text{Row}(\text{Attack}(E)))) =$

$\text{Hp}(\text{CellContent}(\text{Env}(E), \text{Col}(E)-1, \text{Row}(E))) - \text{Damage}(E)$

Face(E) = E and $\text{Col}(E)+1 < \text{Width}(\text{Env}(E))$ and

$\text{CellContent}(\text{Env}(E), \text{Col}(E)+1, \text{Row}(E)) \neq \text{No} \Rightarrow$

$\text{Hp}(\text{CellContent}(\text{Env}(\text{Attack}(E)), \text{Col}(\text{Attack}(E))+1, \text{Row}(\text{Attack}(E)))) =$

$\text{Hp}(\text{CellContent}(\text{Env}(E), \text{Col}(E)+1, \text{Row}(E))) - \text{Damage}(E)$

Service: Cow includes Entity

Constructor: init: Environment x int x int x Dir x int x int \rightarrow [Entity]

Pre init(E,x,y,D,h) requires $4 \geq h \geq 3$

Observations:

[Step]:

$\text{Col}(M) - 1 \leq \text{Col}(\text{step}(M)) \leq \text{Col}(M)+1$

$\text{Row}(M) - 1 \leq \text{Row}(\text{step}(M)) \leq \text{Row}(M)+1$

[Attack]:

Face(E) = N and $\text{Row}(E)+1 < \text{Height}(\text{Env}(E))$ and

$\text{CellContent}(\text{Env}(E), \text{Row}(E)+1, \text{Col}(E)) \neq \text{No} \Rightarrow$

$\text{Hp}(\text{CellContent}(\text{Env}(\text{Attack}(E)), \text{Col}(\text{Attack}(E)), \text{Row}(\text{Attack}(E))+1)) =$
 $\text{Hp}(\text{CellContent}(\text{Env}(E), \text{Col}(E), \text{Row}(E)+1))$
 $\text{Face}(E) = \text{S} \text{ and } \text{Row}(E)-1 \geq 0 \text{ and } \text{CellContent}(\text{Env}(E), \text{Col}(E), \text{Row}(E)-1) \neq$
 $\text{No} \Rightarrow \text{Hp}(\text{CellContent}(\text{Env}(\text{Attack}(E)), \text{Col}(\text{Attack}(E)), \text{Row}(\text{Attack}(E))-1)) =$
 $\text{Hp}(\text{CellContent}(\text{Env}(E), \text{Col}(E), \text{Row}(E)-1))$
 $\text{Face}(E) = \text{W} \text{ and } \text{Col}(E)-1 \geq 0 \text{ and } \text{CellContent}(\text{Env}(E), \text{Col}(E)-1, \text{Row}(E)) \neq$
 $\text{No} \Rightarrow \text{Hp}(\text{CellContent}(\text{Env}(\text{Attack}(E)), \text{Col}(\text{Attack}(E))-1, \text{Row}(\text{Attack}(E)))) =$
 $\text{Hp}(\text{CellContent}(\text{Env}(E), \text{Col}(E)-1, \text{Row}(E)))$
 $\text{Face}(E) = \text{E} \text{ and } \text{Col}(E)+1 < \text{Width}(\text{Env}(E)) \text{ and}$
 $\text{CellContent}(\text{Env}(E), \text{Col}(E)+1, \text{Row}(E)) \neq \text{No} \Rightarrow$
 $\text{Hp}(\text{CellContent}(\text{Env}(\text{Attack}(E)), \text{Col}(\text{Attack}(E))+1, \text{Row}(\text{Attack}(E)))) =$
 $\text{Hp}(\text{CellContent}(\text{Env}(E), \text{Col}(E)+1, \text{Row}(E)))$

Service: Player includes Entity

Observer:

$\text{LastCom}: [\text{Player}] \rightarrow \text{Option}[\text{Command}]$
 $\text{Content}: [\text{Player}] \times \text{int} \times \text{int} \rightarrow \text{Option}[\text{Entity}]$
 $\text{Pre Content}(P, x, y) \text{ requires } x \in \{-1, 0, 1\} \text{ and } y \in \{-1, +3\}$
 $\text{Nature}: [\text{Player}] \times \text{int} \times \text{int} \rightarrow \text{Cell}$
 $\text{Pre Nature}(P, x, y) \text{ requires } x \in \{-1, 0, 1\} \text{ and } y \in \{-1, +3\}$
 $\text{Viewable}: [\text{Player}] \times \text{int} \times \text{int} \rightarrow \text{Cell}$
 $\text{Pre Viewable}(P, x, y) \text{ requires } x \in \{-1, 0, 1\} \text{ and } y \in \{-1, +3\}$
 $\text{foundTreasure}: \text{Player} \rightarrow \text{boolean}$

Observations:

[Invariant]:

Il faut vérifier que $\text{Col}(P)+u$, $\text{Col}(P)+v$, $\text{Row}(P)+u$ et $\text{Row}(P)+v$ ne dépasse pas la définition des dimensions

$\text{Face}(P) = \text{N} \Rightarrow \text{Content}(P, u, v) =$
 $\text{Environment}::\text{CellContent}(\text{Envi}(P), \text{Col}(P)+u, \text{Row}(P)+v)$
 $\text{Face}(P) = \text{N} \Rightarrow \text{Nature}(P, u, v) =$
 $\text{Environment}::\text{CellNature}(\text{Envi}(P), \text{Col}(P)+u, \text{Row}(P)+v)$
 $\text{Face}(P) = \text{S} \Rightarrow \text{Content}(P, u, v) =$
 $\text{Environment}::\text{CellContent}(\text{Envi}(P), \text{Col}(P)-u, \text{Row}(P)-v)$
 $\text{Face}(P) = \text{S} \Rightarrow \text{Nature}(P, u, v) =$
 $\text{Environment}::\text{CellNature}(\text{Envi}(P), \text{Col}(P)-u, \text{Row}(P)-v)$
 $\text{Face}(P) = \text{E} \Rightarrow \text{Content}(P, u, v) =$
 $\text{Environment}::\text{CellContent}(\text{Envi}(P), \text{Col}(P)+v, \text{Row}(P)-u)$
 $\text{Face}(P) = \text{E} \Rightarrow \text{Nature}(P, u, v) =$
 $\text{Environment}::\text{CellNature}(\text{Envi}(P), \text{Col}(P)+v, \text{Row}(P)-u)$
 $\text{Face}(P) = \text{W} \Rightarrow \text{Content}(P, u, v) =$
 $\text{Environment}::\text{CellContent}(\text{Envi}(P), \text{Col}(P)-v, \text{Row}(P)+u)$

Face(P) = W \Rightarrow Nature(P,u,v) =
Environment::CellNature(Envi(P),Col(P)-v,Row(P)+u)

[step]:

LastCom(P)=FF \Rightarrow step(P)=Forward(P)
LastCom(P)=BB \Rightarrow step(P)=Backward(P)
LastCom(P)=RR \Rightarrow step(P)=StrifeR(P)
LastCom(P)=LL \Rightarrow step(P)=StrifeL(P)
LastCom(P)=TR \Rightarrow step(P)=TurnR(P)
LastCom(P)=TL \Rightarrow step(P)=TurnL(P)

Service: Engine

Observer:

Envi: [Engine] \rightarrow Environment
Entities: [Engine] \rightarrow Array[Entity]
getEntity: [Engine] x int \rightarrow Entity
getPlayer: [Engine] \rightarrow Player
isOut: [Engine] \rightarrow boolean
isFinished: [Engine] \rightarrow boolean
isLost: [Engine] \rightarrow boolean

Constructor:

Init: Environment x Player \rightarrow [Engine]

Operator:

removeEntity: [Engine] x int \rightarrow [Engine]
Pre removeEntity(E,i) requires $0 \leq i < \text{size}(\text{Entities}(E))$
addEntity: [Engine] x Entity \rightarrow [Engine]
step: [Engine] \rightarrow [Engine]
Pre step() requires forall i in $[0;\text{size}(\text{Entities}(E))-1]$, Entity::Hp(getEntity(E,i))>0
clean: [Engine] \rightarrow [Engine]

Observations:

[Invariant]:

forall i in $[0;\text{size}(\text{Entities}(E))-1]$, Entity::Envi(getEntity(E,i))=Envi(E)
forall i in $[0;\text{size}(\text{Entities}(E))-1]$, Entity::Col(getEntity(E,i))=x and
Entity::Row(getEntity(E,i))=y
 \Rightarrow Environment::CellContent(Envi(E),x,y)=getEntity(E,i)
(Def) isOut = Environment::CellNature(Envi(E),i,j) == OUT, i ==
Player::Col(getPlayer(E)) and j == Player::Row(getPlayer(E))
(Def) isFinished = foundTreasure(getPlayer(E)) and isOut(E) and isLost(E)
(Def) isLost = Player::getHp(getPlayer(E)) \leq 0

```

[removeEntity]:
    size(Entities(removeEntity(E,i))) = size(Entities(E))-1
    forall k in [0,i-1], getEntity(removeEntity(E,i),k) = getEntity(E,k)
    forall k in [i,size(Entities(E))-2], getEntity(removeEntity(E,i),k) =
getEntity(E,k+1)
[addEntity]:
    size(Entities(addEntity(E,e))) = size(Entities(E))+1
    forall k in [0,size(Entities(E))-1], getEntity(addEntity(E,e),k) = getEntity(E,k)
    getEntity(addEntity(E,e),size(Entities(E))) = e
[clean]:
    forall i in [0;size(Entities(E))-1], Entity::Hp(getEntity(E,i))>0

```

Rapport de projet

Choix d'implantation et difficultés de spécification

Tout d'abord, nous avons écrit les différents contrats en suivant la spécification donnée, cependant nous nous sommes rendus compte que cette spécification était parfois incohérente ou incomplète ; étant donnée qu'elle n'est pas issue de notre travail, nous avons pris un temps non négligeable à corriger et à compléter. En effet, nous avons dû corriger, entre autres, `forward` qui vérifie pour le Nord la `row+1` et `Width` alors qu'il faudrait que cela soit `Height` d'après la spécification plus haut. Nous avons aussi dû vérifier dans les invariants de `Player` que `Col()+-u` et `Col()+-v` ne dépasse pas la définition du donjon. L'environnement, tel qu'il est décrit dans la spécification, ne permet pas la modification de cellule. Or elle nous a été indispensable tant pour effectuer des tests que pour poser les Cellules etc...

Bien que nous ayons pris un certain temps pour compléter la spécification, nous y sommes parvenus. Cependant, nous avons appris assez tard que le `viewable` de `Player` était spécifié uniquement lorsqu'il regardait le Nord, ainsi en raison du temps qu'il nous restait nous avons donc décidé de le laisser ainsi. Cependant, nous aurions pu simplement écrire les autres cas si nous avions eu assez de temps.

D'un autre côté, certaines spécifications n'étaient pas simple à retranscrire en code, par exemple, `isReachable`, qui vérifie que deux points sont accessibles. Nous avons réussi à écrire une fonction récursive qui effectue un parcours en profondeur jusqu'à arriver au point désiré. C'est un algorithme récursif pour les quatre directions possibles qui garde en mémoire le chemin qui sera vérifié par la suite dans le contrat.

Une fois la spécification implantée, nous devons l'enrichir de certaines fonctionnalités. Nous avons ajouté le trésor en tant que type `Cell.TRS` et un observateur sur `Player` pour indiquer si le joueur a récupéré le trésor via une commande (`pick` il ramasse le trésor devant lui s'il existe).

En ce qui concerne les conditions de victoire et défaite, nous avons décidé qu'il y a défaite lorsque le joueur n'a plus de vie, c'est-à-dire `getHp() ≤ 0`. Nous avons donc ajouté un observateur indiquant s'il est mort dans `Player`, d'un autre côté nous avons ajouté dans `engine` un observateur `isOut` indiquant si le joueur est dans la Cellule de sortie et `isFinished` qui vérifie que le joueur est vivant, qu'il a récupéré le trésor et qu'il est dans la sortie. Il s'agit là juste d'un "and" des observateurs. De plus, le joueur étant une entité particulière nous avons décidé d'ajouter dans `EngineService` un observateur permettant de récupérer le `Player`.

En ce qui concerne les combats, nous avons choisi d'ajouter dans le service `Entity` la méthode "`attack`". En effet, nous avons d'abord pensé à l'ajouter dans `MobService`, cependant d'un point de vu sémantique, il s'agit que d'un objet mouvant. De plus, `attack` nécessite d'avoir des points de vie, et nous sommes arrivés à ajouter cette méthode dans `EntityService`. Ceci a induit un autre changement, tous les `Option<MobService>` ont été convertis en `Option<EntityService>` car sans cela il nous aurait été impossible d'utiliser la méthode `attack` et de voir les points de vies des entités à moins de les caster, ce que nous voulions à tout prix éviter. Ainsi, `attack` pour un `Player` `attack` la cible devant lui s'il existe, par contre pour un monstre il `attack` de tous les côtés sauf en diagonales.

Ces ajouts et modifications nous ont pris un certain temps à ajouter puisqu'il faut modifier en cascade les fonctionnalités ajoutées, à partir des interfaces, nous devons remonter aux décorateurs puis modifier les contrats pour enfin compléter l'implantation. Sans compter que devons refaire tourner les tests afin de vérifier la non régression et la validité des contrats ainsi modifiés.

Concernant l'interface graphique, nous avons décidé de prendre une vue du dessus, à la troisième personne. Nous avons utilisé des sprites pour le graphisme. Finalement, par manque de temps, nous ne proposons aucune map pré-conçue : pour jouer, l'utilisateur doit créer sa map. Une fois la map créée et validée, l'utilisateur peut jouer. Le jeu fonctionne correctement. La barre de vie du joueur n'est pas affichée. Il part avec 20 points de vie, les ennemis lui font un mal de 1 point. Les ennemis ont 4 points de vie et le joueur fait un mal de 2 points. La map est visible toute entière, car `isVisible` n'est écrit que pour la direction Nord.

Test pertinents

Nous avons écrit des tests suivant le Model-based testing (MBT). Cependant, pour des raisons de temps, d'efficacité et du nombre de tests à écrire, nous avons pris la décision d'indiquer uniquement les tests de préconditions et les tests de transitions, qui nous semblent ici les plus importants.

Nous avons écrit des tests unitaires qui vérifient que les préconditions et les transitions sont correctes, c'est-à-dire qu'une exception de précondition est levée lorsque le contrat n'est pas respecté ou que l'implantation ne respecte pas les valeurs attendues.

```

@Test
public void setCellContentTransitionTest_1() {
    //cas positif
    try {
        env.init(10, 20);
        EntityService m = new Cow();
        m.init(env, 5, 7, Dir.N, 2, 10);
        @SuppressWarnings("unchecked")
        Option<EntityService> map[][] = new Option[10][20];
        for(int i = 0; i < map.length; i++){
            for(int j = 0; j < map[0].length; j++){
                map[i][j] = env.getCellContent(i, j);
            }
        }
        env.setCellContent(5, 5, m);

        assertTrue(env.getCellContent(5, 5).getValue() == m);
        for(int i = 0; i < 10; i++) {
            for(int j = 0; j < 20; j++) {
                if(i != 5 || j != 5) {
                    assertTrue(map[i][j].getValue() == env.getCellContent(i, j).getValue());
                }
            }
        }
    } catch (InvariantError ie) {
        fail();
    }
}

```

Nous avons aussi des tests qui permettent de vérifier des points plus élaboré tel que `isReachable`.

```

@Test
public void isReachableTransitionTest_1() {
    //cas positif
    try {
        ems.init(10, 20);
        ems.setNature(0, 0, Cell.IN);
        ems.setNature(0, 19, Cell.OUT);
        assertTrue(ems.isReachable(0, 0, 0, 19));
    } catch (PreconditionError ex) {
        fail();
    } catch (InvariantError ie) {
        fail();
    }
}

@Test
public void isReachableTransitionTest_2() {
    //cas negatif
    try {
        ems.init(10, 20);
        ems.setNature(0, 0, Cell.IN);
        ems.setNature(1, 0, Cell.WLL);
        ems.setNature(1, 1, Cell.WLL);
        ems.setNature(0, 1, Cell.WLL);
        ems.setNature(9, 19, Cell.OUT);
        assertTrue(!ems.isReachable(0, 0, 9, 19));
    } catch (PreconditionError ex) {
        fail();
    } catch (InvariantError ie) {
        fail();
    }
}
}

```

Nous vérifions dans le cas positif que la position est bien atteignable, dans l'autre cas, nous enfermons le point dans des murs et vérifions qu'il n'est pas possible de passer.

Il y a aussi un cas de scénario de jeu existant qui cherche le trésor et va à la sortie, nous vérifions que les conditions de victoire sont correctes.


```

@Test
public void GameTransTest_1() {
    //cas positif
    try {
        PlayerService player = new Player();
        player.init(env, 7, 2, Dir.N, 100, 10);
        es.init(env, player);
        player.strafeL();
        player.strafeL();
        player.forward();
        player.forward();
        player.forward();
        player.forward();
        player.pickItem();
        player.forward();
        player.forward();
        player.forward();
        player.forward();
        player.forward();
        player.forward();
        player.forward();
        player.strafeR();
        player.strafeR();
        player.strafeR();
        player.strafeR();
        assertTrue(es.isFinished());
    } catch (PreconditionError pe) {
        fail();
    } catch (InvariantError e) {
        fail();
    }
}

```

Finalement le fichier RunningGameTest, nous a permis de faire quelques tests informels, puisqu'il s'agit d'une instantiation du jeu en ASCII avec les contrats et qui attend une commande à effectuer à chaque tour.