SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

CPS 2019: Liens et Spécification partielle.
(version du 05/04/19)

Liens

Page *Wikipedia* de *Lode Runner*	`https://en.wikipedia.org/wiki/Lode_Runner`
Vidéo *Youtube* d'une partie de *Lode Runner*	`https://www.youtube.com/watch?v=PWwyhymcDxI`
Interview du créateur du jeu	`https://www.retrogamer.net/retro_games80/the-making-of-lode-runner/`
Jeu dans un navigateur	`http://loderunnerwebgame.com/game/`
Vidéo *Youtube* d'un écran de *Lode Runner 2* (en 3D)	`https://www.youtube.com/watch?v=iMlhmiRCPO8`

Spécification Partielle

Cette spécification est une correction partielle du sujet d'examen permettant de disposer d'une base commune pour le projet. Il n'est absolument pas obligatoire de l'utiliser dans le projet, les spécifications personnelles sont encouragées.

En découvrant des erreurs dans cette spécification, il convient de contacter l'équipe pédagogique (ou d'en parler en TME) pour qu'elle soit mise à jour.

Ecran

Service: Screen

Observators:
const `Height`: [Screen] $\to$ int
const `Width`: [Screen] $\to$ int
`CellNature`: [Screen] $\times$ int $\times$ int $\to$ Cell
  pre `CellNature(S,x,y)` **requires** $0 \leq$ `y` $<$ `Height(S)` **and** $0 \leq$ `x` $<$ `Width(S)`

Constructors:
`init`: int $\times$ int $\to$ [Screen]
  pre `init(h,w)` **requires** $0 <$ `h` **and** $0 <$ `w`

Operators:
`Dig`: [Screen] $\times$ int $\times$ int $\to$ [Screen]
  pre `Dig(S,x,y)` **requires** `CellNature(S,x,y)` $=$ **PLT**
`Fill`: [Screen] $\times$ int $\times$ int $\to$ [Screen]
  pre `Dig(S,x,y)` **requires** `CellNature(S,x,y)` $=$ **HOL**

Observations:
[init]: `Height(init(h,w))` $=$ `h`
`Width(init(h,w))` $=$ `w`
forall (x, y) **in** `[0;Width(S)[`$\times$`[0;Height(S)[`, `CellNature(init(h,w),x,y)` $=$ **EMP**
[Dig]: `CellNature(Dig(S,x,y)),x,y` $=$ **HOL**
forall (x, y) **in** `[0;Width(S)[` $\times$ `[0;Height(S)[`,
  (`x` $\neq$ `u` **or** `y` $\neq$ `v`) **implies** `CellNature(Dig(S,u,v)),x,y` $=$ `CellNature(x,y)`
[Fill]: `CellNature(Fill(S,x,y),x,y)` $=$ **PLT**
forall (x, y) **in** `[0;Width(S)[`$\times$`[0;Height(S)[`,
  (`x` $\neq$ `u` **or** `y` $\neq$ `v`) **implies** `CellNature(Fill(S,u,v)),x,y` $=$ `CellNature(x,y)`

Ecran éditable

Service: EditableScreen **includes** Screen
Observators: `Playable`: [EditableScreen] $\rightarrow$ bool
Operators: `SetNature`: [EditableScreen] $\times$ int $\times$ int $\times$ Cell $\rightarrow$ [EditableScreen]
pre `SetNature(S,x,y,C)` **requires** $0 \leq$ `y` $<$ `Height(S)` **and** $0 \leq$ `x` $<$ `Width(S)`
Observations:
[invariant]: `Playable(S)` **min**
forall (x, y) **in** `[0;Width(S)[`$\times$ `[0;Height(S)[`, `CellNature(S,x,y)` $\neq$ **HOL**
and forall x **in** `[0;Width(S)[`, `CellNature(S,x,0)` $=$ **MTL**
[SetNature]: `CellNature(SetNature(S,x,y,C)),x,y` $=$ C
forall (x, y) **in** `[0;Width(S)[`$\times$ `[0;Height(S)[`,
(`x` $\neq$ `u` **or** `y` $\neq$ `v`) **implies** `CellNature(SetNature(S,u,v,C)),x,y)` $=$ `CellNature(x,y)`

Environnement

Service: Environment **includes** Screen
Observators : `CellContent`: int $\times$ int $\rightarrow$ Set{Character + Item}
pre `CellContent(E,x,y)` **requires** $0 \leq$ `y` $<$ `Height(S)` **and** $0 \leq$ `x` $<$ `Width(S)`
Constructors: `init`: EditableScreen $\rightarrow$ Environment
Observations:
[invariant]: **forall** (x, y) **in** `[0;Width(E)[`$\times$ `[0;Height(E)[`,
forall Character `c1`, `c2` **in** `CellContent(E,x,y)`^{2}, `c1` $=$ `c2`
forall (x, y) **in** `[0;Width(E)[`$\times$ `[0;Height(E)[`,
`CellNature(E,x,y)` **in** {**MTL**, **PLR**} **implies** `CellContent(x,y)` $= \emptyset$
forall (x, y) **in** `[0;Width(E)[`$\times$ `[0;Height(E)[`,
exists Treasure `t` **in** `CellContent(E,x,y)`
implies (`CellNature(E,x,y)` $=$ **EMP and** `CellNature(E,x,y-1)` **in** {**PLT**, **MTL**})
[init]: **forall** (x, y) **in** `[0;Width(E)[`$\times$ `[0;Height(E)[`,
`CellNature(init(S),x,y)` $=$ EditableScreen::`CellNature(S,x,y)`

Personnage

Service: Character

Observators: **const** `Envi`: [Character] → Environment
`Hgt`: [Character] → int
`Wdt`: [Character] → int

Operators: `init`: Screen × int × int → [Character]
pre init(S,x,y) **requires** Environment::CellNature(S,x,y) = **EMP**
`GoLeft`: [Character] → [Character]
`GoRight`: [Character] → [Character]
`GoUp`: [Character] → [Character]
`GoDown`: [Character] → [Character]

Observations:

`[invariant]`: Environment::CellNature(Envi(C),Wdt(C),Hgt(C)) **in** {**EMP**, **HOL**, **LAD**, **HDR**}
exists Character `x` **in** Environment::CellContent(Envi(C),Wdt(C),Hgt(C)) **implies** `x` = `C`

`[GoLeft]`: Hgt(GoLeft(C)) = Hgt(C)
Wdt(C) = 0 **implies** Wdt(GoLeft(C)) = Wdt(C)
Environment::CellNature(Envi(C),Wdt(C)-1,Hgt(C)) **in** {**MTL**, **PLT** } **implies** Wdt(GoLeft(C)) = Wdt(C)
Environment::CellNature(Envi(C),Wdt(C),Hgt(C)) **not in** {**LAD**, **HDR**}
and Environment::CellNature(Envi(C),Wdt(C),Hgt(C)-1) **not in** {**PLT**, **MTL**, **LAD** }
and not exists Character c **in** Environment::CellContent(Envi(C),Wdt(C),Hgt(C)-1)
implies Wdt(GoLeft(C)) = Wdt(C)
exists Character c **in** Environment::CellContent(Envi(C),Wdt(C)-1,Hgt(C))
implies Wdt(GoLeft(C)) = Wdt(C)
(Wdt(C) ≠ 0) **and** Environment::CellNature(Envi(C),Wdt(C)-1,Hgt(C)) **not in** {**MTL**, **PLT** }
and (Environment::CellNature(Envi(C),Wdt(C),Hgt(C)) **in** {**LAD**, **HDR**}
or Environment::CellNature(Envi(C),Wdt(C),Hgt(C)-1) **in** {**PLT**, **MTL**, **LAD**}
or exists Character c **in** Environment::CellContent(Envi(C),Wdt(C),Hgt(C)-1))
and not (**exists** Character c **in** Environment::CellContent(Envi(C),Wdt(C)-1,Hgt(C)))
implies Wdt(GoLeft(C)) = Wdt(C)-1

Garde

Service: Guard **includes** Character

Observators: **const** `Id`: [Guard] → int

`Behaviour`: [Guard] → Move

`Target`: [Guard] → Character

`TimeInHole`: [Guard] → int

Operators: `ClimbLeft`: [Guard] → [Guard]

pre `ClimbLeft(G)` **requires** `Environment::CellNature(Envi(G),Hgt(G),Wdt(G))` = **HOL**

`Step`: [Guard] → [Guard]

Observations:

[invariant]: Environment::CellNature(Envi(G),Wdt(G),Hgt(G)) = **LAD**
and Hgt(G) < Character::Hgt(Target(G))
and (Environment::CellNature(Envi(G),Wdt(G),Hgt(G)-1) **not in** {**PLT**, **MTL** }
or exists Character c **in** Environment::CellContent(Envi(G),Wdt(G),Hgt(G)-1)
implies Environment::Hgt(Target(G)) - Hgt(G) < |Environment::Wdt(Target(G)) - Wdt(G))|)
implies Behaviour(G) = **Up**

(. . .)

[init]: (. . .)

[ClimbLeft]: Wdt(C) = 0 **implies** Wdt(ClimbLeft(C)) = Wdt(C) **and** Hgt(ClimbLeft(C)) = Hgt(C)
Screen::CellNature(Envi(C),Wdt(C)-1,Hgt(C) +1) **in** {**MTL**, **PLT** }
implies Wdt(ClimbLeft(C)) = Wdt(C) **and** Hgt(ClimbLeft(C)) = Hgt(C)
exists Character c **in** Environment::CellContent(Envi(C),Wdt(C)-1,Hgt(C)+1)
implies Wdt(ClimbLeft(C)) = Wdt(C) **and** Hgt(ClimbLeft(C)) = Hgt(C)
Wdt(C) ≠ 0 **and** Screen::CellNature(Envi(C),Wdt(C)-1,Hgt(C)+1) **notin** {**MTL**, **PLT** }
and not exists Character c **in** Environment::CellContent(Envi(C),Wdt(C)-1,Hgt(C)+1)
implies Wdt(ClimbLeft(C)) = Wdt(C)-1 **and** Hgt(ClimbLeft(C)) = Hgt(C)+1

[Step]: (. . .)

définir des prédicats et les réutiliser dans les gardes permet de rendre plus lisible la spécification, par exemple:
WillFall`(C)` **defined by** (Environment::CellNature(Envi(C),Wdt(C),Hgt(C)-1) **in** {**HOL**, **EMP** }
and not exists Character c **in** Environment::CellContent(Envi(C),Wdt(C),Hgt(C)-1)
and Environment::CellNature(Envi(C),Wdt(C),Hgt(C)) **not in** {**LAD**, **HDR** })

(. . .)