Service: GameEng

use: Level, Lemming
types: Lemming, int, boolean, double

Observateurs:

getLevel: [GameEng] → Level
isObstacle: [GameEng] x int x int → boolean
    pre isObstacle(G, x, y) require x ∈ [0; getHeight(getLevel(G))[
        ∧ y ∈ [0; getWidth(getLevel(G))[
isObstacle2: [GameEng] x int x int → boolean
    pre isObstacle2(G, x, y) require x ∈ [0; getHeight(getLevel(G))[
        ∧ y ∈ [0; getWidth(getLevel(G))[
isLibre: [GameEng] x int x int → boolean
    pre isLibre(G, x, y) require x ∈ [0; getHeight(getLevel(G))[
        ∧ y ∈ [0; getWidth(getLevel(G))[
getSizeColony: [GameEng] → int
getSpawnSpeed: [GameEng] → int
setSpawnSpeed: [GameEng] x int → int
    pre setSpawnSpeed(G, s) require s > 0 ∧ ¬gameOver(G)
gameOver: [GameEng] → boolean
score: [GameEng] → double
    pre score(G) require gameOver(G)
getNombreTours: [GameEng] → int
    pre getNombreTours(G) require ¬Level::isEditing(getLevel(G)) ∧ ¬gameOver(G)
getNombreToursFinal: [GameEng] → int
    pre getNombreTousFinal(G) require ¬gameOver(G)
getLemmingsActifs: [GameEng] → Set<int>
    pre getLemmingsActifs(G) require ¬gameOver(G)
getLemming: [GameEng] x int → Lemming
    pre getLemming(G, i) require ¬gameOver(G) ∧ i ∈ getLemmingsActifs(G)
getNombreSauves: [GameEng] → int
getNombreMorts: [GameEng] → int
getNombreActifs: [GameEng] → int
getNombreCrees: [GameEng] → int
isAnnihilation: [GameEng] → boolean

Constructeurs:

init: int x int → [GameEng]
    pre init(sc, ss) require sc > 0 ∧ ss > 0

Operateurs:

step: [GameEng] → [GameEng]
    pre step(G) require ¬gameOver(G)
tuerLemming: [GameEng] x int → [GameEng]
    pre tuerLemming(G, i) require ¬gameOver(G) ∧ i ∈ getLemmingsActifs(G)

sauverLemming: [GameEng] x int → [GameEng]

pre sauverLemming(G, i) require ¬gameOver(G) ∧ i ∈ getLemmingsActifs(G)

goAnnihilation: [GameEng] → [GameEng]

pre goAnnihilation require ¬gameOver(G)

Observations:

[Invariants]

getNombreActifs(G) =(min) |getLemmingsActifs()|

score(G) =(min) getNombreSauves(G)/getSizeColony(G)

getNombreCrees(G) =(min)
getNombreActifs(G)+getNombreSauves(G)+getNombreMorts(G)+
getNombreTours(G) ≥ 0

getNombreSauves(G) ≥ 0 ∧ getNombreSauves(G) ≤ getNombreCrees(G)

getNombreMorts(G) ≥ 0 ∧ getNombreMorts(G) ≤ getNombreCrees(G)

getNombreActifs(G) ≥ 0 ∧ getNombreActifs(G) ≤ getNombreCrees(G)

getNombreCrees(G) ≥ 0 ∧ getNombreCrees(G) ≤ getSizeColony(G)

[init]

getSpawnSpeed(init(sc, ss)) = ss

getSizeColony(init(sc, ss)) = sc

getNombreSauves(init(sc, ss)) = 0

getNombreMorts(init(sc, ss)) = 0

getNombreActifs(init(sc, ss)) = 0

getNombreCrees(init(sc, ss)) = 0

getNombreTours(init(sc, ss)) = 0

isAnnihilation = false

[setSpawnSpeed]

getSpawnSpeed(setSpawnSpeed(G, s)) = s

[step]

¬isAnnihilation(G) ∧ (getNombreTours(G) % getSpawnSpeed(G) = 0)
∧ getNombreCrees(G) < getSizeColony(G)
⇒ getNombreCrees(step(G)) = getNombreCrees(G)+1
∧ getNombreCrees(step(G)) ∈ getNombreActifs(step(G))
∧ Lemming::getX(getLemming(step(G), getNombreCrees(step(G))) = x
∧ Lemming::getY(getLemming(step(G), getNombreCrees(step(G))) = y
∧ Level::getEntrance(getLevel(step(G), x, y))

isAnnihilation(G) ∧ (getNombreTours(G) % getSpawnSpeed(G) = 0)
∧ getNombreCrees(G) < getSizeColony(G)
⇒ getNombreCrees(step(G)) = getNombreCrees(G)

getNombreTours(step(G)) = getNombreTours(G)+1

[tuerLemming]

getNombreActifs(tuerLemming(G;i)) = getNombreActifs(G)-1

getNombreMorts(tuerLemming(G,i)) = getNombreMorts(G)+1

getLemmingsActifs(tuerLemming(G, i)) = getLemmingsActifs(G)\{i}

[sauverLemming]

getNombreActifs(sauverLemming(G;i)) = getNombreActifs(G)-1

getNombreSauves(sauverLemming(G,i)) = getNombreSauves(G)+1

getLemmingsActifs(sauverLemming(G, i)) = getLemmingsActifs(G)\{i}

[annihilation]

isAnnihilation(annihilation(G)) = true