

Examen réparti 1 – Street Fighter II TM!

Une solution

Binh-Minh Bui-Xuan

25 février 2014

Une solution à l'énoncé de l'examen réparti 1 pour le cours CPS 2014.

Personnage

```

service : Personnage
types : String, int, boolean
observers :
  const nom : [Personnage] → String
  const largeur : [Personnage] → int
  const hauteur : [Personnage] → int
  const force : [Personnage] → int
  pointsDeVie : [Personnage] → int
  estVaincu : [Personnage] → boolean
Constructors :
  init : String × int × int × int × int → [Personnage]
    pre init(nom,largeur,hauteur,force,pointsVie) require nom ≠ "" ∧ largeur%2=1 ∧ hauteur%2=1 ∧ 0 < force < pointsVie
Operators :
  retrait : [Personnage] × int → [Personnage]
    pre retrait(P,s) require ¬estVaincu(P) ∧ s>0
Observations :
[invariants]
  estVaincu(P)  $\stackrel{\text{min}}{=}$  pointsDeVie(P) ≤ 0
[init]
  nom(init(n,l,h,f,p))=n
  largeur(init(n,l,h,f,p))=l
  hauteur(init(n,l,h,f,p))=h
  force(init(n,l,h,f,p))=f
  pointsDeVie(init(n,l,h,f,p))=p
[retrait]
  pointsDeVie(retrait(P,s))=pointsDeVie(P) -s

```

MoteurJeu

```

service : MoteurJeu
use : GestionCombat
types : boolean, int, enum RESULTAT{GUILLEGAGNANT, GUILPERDANT, NULLE},
        enum COMMANDE{RIEN, GAUCHE, DROITE, FRAPPE}
observers :
  const maxPasJeu : [MoteurJeu] → int
  pasJeuCourant : [MoteurJeu] → int
  estFini : [MoteurJeu] → boolean
  resultatFinal : [MoteurJeu] → RESULTAT
    pre resultatFinal(M) require estFini(M)
  combat : [MoteurJeu] → GestionCombat
Constructors :
  init : int × int × int → [MoteurJeu]
    pre init(largeur,hauteur,maxPas) require largeur ≥ 256 ∧ hauteur ≥ 240 ∧ maxPas ≥ 0
Operators :
  pasJeu : [MoteurJeu] × COMMANDE × COMMANDE → [MoteurJeu]
    pre pasJeu(M,comGuile,comRyu) require ¬estFini(M)
Observations :
[invariants]
  0 ≤ pasJeuCourant(M) ≤ maxPasJeu(M)
  estFini(M)  $\stackrel{\text{min}}{=}$  (Personnage::estVaincu(GestionCombat::guile(combat(M)))
    ∨ Personnage::estVaincu(GestionCombat::ryu(combat(M)))
    ∨ pasJeuCourant(M)=maxPasJeu(M))
  resultatFinal(M)  $\stackrel{\text{min}}{=}$   $\begin{cases} \text{GUILLEGAGNANT} & \text{si } \text{Personnage::estVaincu(GestionCombat::ryu(combat(M)))} \\ & \wedge \neg \text{Personnage::estVaincu(GestionCombat::guile(combat(M)))} \\ \text{GUILPERDANT} & \text{si } \text{Personnage::estVaincu(GestionCombat::guile(combat(M)))} \\ & \wedge \neg \text{Personnage::estVaincu(GestionCombat::ryu(combat(M)))} \\ \text{NULLE} & \text{sinon} \end{cases}$ 
[init]
  maxPasJeu(init(l,h,m))=m
  pasJeuCourant(init(l,h,m))=0
  combat(init(l,h,m))=GestionCombat::init(l,h)
[pasJeu]
  pasJeuCourant(pasJeu(M,cG,cR))=pasJeuCourant(M) +1
  combat(pasJeu(M,cG,cR))=GestionCombat::gerer(combat(M),cG,cR)

```

GestionCombat

service : GestionCombat

use : Personnage

types : boolean, int, String, enum COMMANDE{RIEN, GAUCHE, DROITE, FRAPPE}

observers :

```
const largeur : [GestionCombat] → int
const hauteur : [GestionCombat] → int
guile : [GestionCombat] → Personnage
ryu : [GestionCombat] → Personnage
estFrappé : [GestionCombat] × String → boolean
  pre estFrappé(C,id) require id="Guile" ∨ id="Ryu"
estGelé : [GestionCombat] × String → boolean
  pre estGelé(C,id) require id="Guile" ∨ id="Ryu"
position : [GestionCombat] × String → int
  pre position(C,id) require id="Guile" ∨ id="Ryu"
collisionGauche : [GestionCombat] × String × String → boolean
  pre collisionGauche(C,id1,id2) require (id1="Guile" ∧ id2="Ryu") ∨ (id1="Ryu" ∧ id2="Guile")
collision : [GestionCombat] → boolean
```

Constructors :

```
init : int × int → [GestionCombat]
```

Operators :

```
gerer : [GestionCombat] × COMMANDE × COMMANDE → [GestionCombat]
```

Observations :

[invariants]

$$\text{collisionGauche}(C, id1, id2) \stackrel{\text{min}}{=} \left(-d \leq \text{position}(C, id2) - \text{position}(C, id1) \leq d+1 \right) \\ \wedge \left(d = \text{Personnage::largeur}(\text{guile}(C))/2 + \text{Personnage::largeur}(\text{ryu}(C))/2 \right)$$

$$\text{collision}(C) \stackrel{\text{min}}{=} \text{collisionGauche}(C, "Guile", "Ryu") \vee \text{collisionGauche}(C, "Ryu", "Guile")$$

[init]

```
largeur(init(l,h))=l ∧ hauteur(init(l,h))=h
guile(init(l,h))=Personnage::init("Guile",13,51,16,1664)
ryu(init(l,h))=Personnage::init("Ryu",13,51,16,1664)
estFrappé(init(l,h),id)=false
estGelé(init(l,h),id)=false
position(init(l,h),"Guile")=10 + Personnage::largeur(guile(C))/2 + 1
position(init(l,h),"Ryu")=-10 - Personnage::largeur(ryu(C))/2 - 1
```

[gerer]

$$\text{guile}(\text{gerer}(C, cG, cR)) = \begin{cases} \text{guile}(C) & \text{si } \neg \text{estFrappé}(\text{gerer}(C, cG, cR), "Guile") \\ \text{Personnage::retrait}(\text{guile}(C), \text{Personnage::force}(\text{ryu}(C))) & \text{sinon} \end{cases}$$

$$\text{ryu}(\text{gerer}(C, cG, cR)) = \begin{cases} \text{ryu}(C) & \text{si } \neg \text{estFrappé}(\text{gerer}(C, cG, cR), "Ryu") \\ \text{Personnage::retrait}(\text{ryu}(C), \text{Personnage::force}(\text{guile}(C))) & \text{sinon} \end{cases}$$

```
estFrappé(gerer(C,cG,cR),"Guile") = ¬estGelé(C,"Ryu") ∧ cR=FRAPPE ∧ collision(C)
estFrappé(gerer(C,cG,cR),"Ryu") = ¬estGelé(C,"Guile") ∧ cG=FRAPPE ∧ collision(C)
estGelé(gerer(C,cG,cR),"Guile") = estFrappé(gerer(C,cG,cR),"Guile") ∨ (¬estGelé(C,"Guile") ∧ cG=FRAPPE)
estGelé(gerer(C,cG,cR),"Ryu") = estFrappé(gerer(C,cG,cR),"Ryu") ∨ (¬estGelé(C,"Ryu") ∧ cR=FRAPPE)
```

//Attention : concernant la correction de l'examen 1, on omettera dans ce qui suit les appels à Personnage::largeur car c'est fastidieux...
 //... mais cela ne veut pas dire qu'on a le droit de faire pareil dans le projet de l'UE.

$$\text{position}(\text{gerer}(C, cG, cR), "Guile") = \begin{cases} \min\{\text{position}(C, "Guile")+64, \text{largeur}(C)\} & \text{si } \text{estFrappé}(\text{gerer}(C, cG, cR), "Guile") \\ & \wedge \text{collisionGauche}(C, "Guile", "Ryu") \\ \max\{\text{position}(C, "Guile")-64, 0\} & \text{si } \text{estFrappé}(\text{gerer}(C, cG, cR), "Guile") \\ & \wedge \text{collisionGauche}(C, "Ryu", "Guile") \\ \max\{\text{position}(C, "Guile")-1, 0\} & \text{si } \neg \text{estFrappé}(\text{gerer}(C, cG, cR), "Guile") \\ & \wedge \neg \text{collisionGauche}(C, "Guile", "Ryu") \wedge cG=\text{GAUCHE} \\ \min\{\text{position}(C, "Guile")+1, \text{largeur}(C)\} & \text{si } \neg \text{estFrappé}(\text{gerer}(C, cG, cR), "Guile") \\ & \wedge \neg \text{collisionGauche}(C, "Ryu", "Guile") \wedge cG=\text{DROITE} \\ \text{position}(C, "Guile") & \text{sinon} \end{cases}$$

$$\text{position}(\text{gerer}(C, cG, cR), "Ryu") = \begin{cases} \min\{\text{position}(C, "Ryu")+64, \text{largeur}(C)\} & \text{si } \text{estFrappé}(\text{gerer}(C, cG, cR), "Ryu") \\ & \wedge \text{collisionGauche}(C, "Ryu", "Guile") \\ \max\{\text{position}(C, "Ryu")-64, 0\} & \text{si } \text{estFrappé}(\text{gerer}(C, cG, cR), "Ryu") \\ & \wedge \text{collisionGauche}(C, "Guile", "Ryu") \\ \max\{\text{position}(C, "Ryu")-1, 0\} & \text{si } \neg \text{estFrappé}(\text{gerer}(C, cG, cR), "Ryu") \\ & \wedge \neg \text{collisionGauche}(C, "Ryu", "Guile") \wedge cR=\text{GAUCHE} \\ \min\{\text{position}(C, "Ryu")+1, \text{largeur}(C)\} & \text{si } \neg \text{estFrappé}(\text{gerer}(C, cG, cR), "Ryu") \\ & \wedge \neg \text{collisionGauche}(C, "Guile", "Ryu") \wedge cR=\text{DROITE} \\ \text{position}(C, "Ryu") & \text{sinon} \end{cases}$$