

Chapitre 1

Projet CPS : Spécifications de River City Ransom

Béatrice CARRE et Steven VAROUMAS

1.1 Le service Personnage

`service` : Personnage
`use` : Objet
`types` : String , int , boolean

Observers :

- `const` nom : [Personnage] → String
- `const` largeur : [Personnage] → int
- `const` hauteur : [Personnage] → int
- `const` profondeur : [Personnage] → int
- `const` force : [Personnage] → int
- points_de_vie : [Personnage] → int
- somme_d_argent : [Personnage] → int
- est_vaincu : [Personnage] → boolean
- est_equipe_objet : [Personnage] → boolean
- est_equipe_perso : [Personnage] → boolean
- objet_equipe : [Personnage] → Objet
 - `pre` objet_equipe(P) `require` est_equipe_objet(P)
- perso_equipe : [Personnage] → Personnage
 - `pre` perso_equipe(P) `require` est_equipe_perso(P)

Constructors :

- init : String × int × int × int × int × int × int → [Personnage]
 - `pre` init(nom, largeur, hauteur, profondeur, force, pdv, argent) `require` nom = "Alex" ∨ nom = "Ryan" ∧ largeur > 0 ∧ hauteur > 0 ∧ profondeur > 0 ∧ force > 0 ∧ pdv > 0 ∧ argent ≥ 0

Operators :

- retrait_vie : [Personnage] × int → [Personnage]
 - `pre` retrait_vie(P, s) `require` ¬est_vaincu(P) ∧ s > 0
- depot_vie : [Personnage] × int → [Personnage]
 - `pre` depot_vie(P, s) `require` ¬est_vaincu(P) ∧ s > 0
- retrait_argent : [Personnage] × int → [Personnage]

```

    pre retrait_argent(P,s) require ¬est_vaincu(P) ∧ s>0 ∧ somme_d_argent(P) ≥ s
    // pour ne pas avoir une somme negative
depot_argent : [Personnage] × int → [Personnage]
    pre depot_argent(P,s) require ¬est_vaincu(P) ∧ s>0
ramasser_objet : [Personnage] × Object → [Personnage]
    pre ramasser_objet(P,o) require ¬est_vaincu(P) ∧ ¬est_equipe_objet(P) ∧ ¬
    est_equipe_perso(P)
    ramasser_perso : [Personnage] × Personnage → [Personnage]
    pre ramasser_perso(P,p) require ¬est_vaincu(P) ∧ ¬est_equipe_objet(P) ∧
    ¬est_equipe_perso(P)
jeter : [Personnage] → [Personnage]
    pre jeter(P) require ¬est_vaincu(P) ∧ ( est_equipe_objet(P) ∨ est_equipe_perso
    (P) )

```

Observations :

[invariants]

```

    est_vaincu(P)  $\stackrel{min}{=}$  points_de_vie(P) ≤ 0
    est_equipe_perso(P)  $\stackrel{min}{=}$  perso_equipe(P) ≠ null
    est_equipe_objet(P)  $\stackrel{min}{=}$  objet_equipe(P) ≠ null

```

[init]

```

    nom(init(n,l,h,p,f,v,a))=n
    largeur(init(n,l,h,p,f,v,a))=l
    hauteur(init(n,l,h,p,f,v,a))=h
    profondeur(init(n,l,h,p,f,v,a))=p
    force(init(n,l,h,p,f,v,a))=f
    points_de_vie(init(n,l,h,p,f,v,a))=v
    somme_d_argent(init(n,l,h,p,f,v,a))=a
    objet_equipe(init(n,l,h,p,f,v,a))=null
    perso_equipe(init(n,l,h,p,f,v,a))=null

```

[retrait_vie]

```

    points_de_vie(retrait_vie(P,s)) = points_de_vie(P) - s

```

[depot_vie]

```

    points_de_vie(depot_vie(P,s)) = points_de_vie(P) + s

```

[retrait_argent]

```

    somme_d_argent(retrait_argent(P,s)) = argent(P) - s

```

[depot_argent]

```

    somme_d_argent(depot_argent(P,s)) = argent(P) + s

```

[ramasser_objet]

```

    objet_equipe(ramasser_objet(P,objet)) = objet
    force(ramasser_objet(P,objet)) =
    { force(P) + Objet : :bonus_force(objet) si Objet : :est_equipable(objet)
    { force(P) sinon

    somme_d_argent(ramasser_objet(P,objet)) =
    { somme_d_argent(P) + Objet : :valeur_marchande(objet) si Objet : :est_de_valeur(objet)
    { somme_d_argent(P) sinon

```

[ramasser_perso]

```
perso_equipe(ramasser_perso(P, perso)) = perso
```

[jeter]

```
perso_equipe(jeter(P)) = null
force(jeter(P)) =
  { force(P) - Objet : :bonus_force(objet_equipe(P))
    si est_equipe_objet(P) ∧ Objet : :est_equipable(objet_equipe(P))
    force(P) sinon
  }
objet_equipe(jeter(P)) = null
```

1.2 Gangster

service : Gangster

Refine : Personnage

use : enum ACTION{RIEN, FRAPPE, SAUTE, HAUT, BAS, GAUCHE, DROITE}

Observers :

```
action : [Gangster] → ACTION
pre action(G) require ¬estVaincu(G)
```

Constructors :

```
init : String × int × int × int × int × int × int → [Gangster]
pre init(nom, largeur, hauteur, profondeur, force, pdv, argent) require nom ≠ ""
  ∧ largeur > 0 ∧ hauteur > 0 ∧ profondeur > 0 ∧ force > 0 ∧ pdv > 0 ∧ argent ≥ 0
```

Observations :

[init]

```
nom(init(n, l, h, p, f, v, a)) = n
largeur(init(n, l, h, p, f, v, a)) = l
hauteur(init(n, l, h, p, f, v, a)) = h
profondeur(init(n, l, h, p, f, v, a)) = p
force(init(n, l, h, p, f, v, a)) = f
points_de_vie(init(n, l, h, p, f, v, a)) = v
somme_d_argent(init(n, l, h, p, f, v, a)) = a
objet_equipe(init(n, l, h, p, f, v, a)) = null
perso_equipe(init(n, l, h, p, f, v, a)) = null
action(init(n, l, h, p, f, v, a)) = RIEN
```

1.3 Bloc

service : Bloc

use : Objet

types : enum TYPE{VIDE, FOSSE, OBJET},

Observers :

```
const type : [Bloc] → TYPE
const objet : [Bloc] → Objet
```

Constructors :

```
init : TYPE × Objet → [Bloc]
pre init(t, o) require
  (t = VIDE ∨ t = FOSSE) ∧ o = null ∨ (t = OBJ ∧ o ≠ null)
```

Operators :

```
retirerObjet : [Bloc] → [Bloc]
pre retirerObjet(B) require type(B) = OBJ ∧
poserObjet : [Bloc] × Objet → [Bloc]
pre poserObjet(B, o) require type(B) = VIDE
```

Observations :

```
[init]
  type(init(t,o)) = t
  objet(init(t,o)) = o
[retirerObjet]
  type(retirerObjet(B)) = VIDE
  objet(retirerObjet(B)) = null
[poserObjet]
  type(poserObjet(B,o)) = OBJET
  objet(poserObjet(B,o)) = o
```

1.4 Objet

```
service : Objet
types : String, boolean, int
Observers :
  const nom : [Objet] → String
  est_equipable : [Objet] → boolean
  est_de_valeur : [Objet] → boolean
  bonus_force : [Objet] → int
  pre bonus_force(O) require est_equipable(O)
  valeur_marchande : [Objet] → int
  pre valeur_marchande(O) require est_de_valeur(O)
```

Constructors :

```
init : String × int × int → [Objet]
  pre(init(n,t,bonus,valeur) require n≠"" ∧ ( ( bonus >0 ∧ valeur = 0) ∨ (bonus
    = 0 ∧ valeur > 0) )
```

Observations :

```
[Invariants]
  est_equipable(O)  $\stackrel{min}{=}$  bonus_force > 0
  est_de_valeur(O)  $\stackrel{min}{=}$  valeur_marchande > 0
  est_equipable(O)  $\stackrel{min}{=}$  ¬est_de_valeur(O)
```

```
[init]
  nom(init(n,bonus,valeur)) = n
  bonus_force(init(n,bonus,valeur)) = bonus
  valeur_marchande(init(n,bonus,valeur)) = valeur
```

1.5 Terrain

```
service : Terrain
use : Bloc
types : int
Observers :
  const largeur : [Terrain] → int
  const hauteur : [Terrain] → int
  const profondeur : [Terrain] → int
  bloc : [Terrain] × int × int × int → Bloc
```

```
pre bloc( T, i, j, k) require 0 ≤ i ≤ largeur ∧ 0 ≤ j ≤ hauteur ∧ 0 ≤ k ≤
profondeur
```

Constructors :

```
init : int × int × int → [Terrain]
pre init(largeur, hauteur, prof) require largeur > 0 ∧ hauteur > 0 ∧ prof > 0
```

Operators :

```
modifier_bloc : [Terrain] × int × int × int × Bloc → [Terrain]
pre bloc( T, i, j, k, b) require 0 ≤ i ≤ largeur ∧ 0 ≤ j ≤ hauteur ∧ 0 ≤ k ≤
profondeur ∧ b ≠ null
```

Observations :

[Invariants]

[init]

```
largeur(init(l, h, p)) = l
hauteur(init(l, h, p)) = h
profondeur(init(l, h, p)) = p
bloc(init(l, h, p), x, y, z) ≠ NULL
```

[modifier_bloc]

```
bloc(modifier_bloc(T, x, y, z, b), x, y, z) = b
```

1.6 Moteur de jeu

service : MoteurJeu

use : GestionCombat

types : boolean, int, enum RESULTAT{DEUXGAGNANTS, RYANGAGNANT, ALEXGAGNANT, SLICKGAGNANT, NULLE},
enum COMMANDE{RIEN, GAUCHE, DROITE, BAS, HAUT, FRAPPE, SAUT, SAUTHAUT, SAUTDROIT, SAUTGAUCHE, SAUTBAS, RAMASSER, JETER}

Observers :

```
estFini : [MoteurJeu] → boolean
resultat : [MoteurJeu] → RESULTAT
pre resultat(M) require estFini(M)
combat : [MoteurJeu] → GestionCombat
```

Constructors :

```
init : ∅ → [MoteurJeu]
```

Operators :

```
pasJeu : [MoteurJeu] × COMMANDE × COMMANDE → [MoteurJeu]
pre pasJeu(M, comAlex, comRyan) require ¬estFini(M)
```

Observations :

[Invariants]

$$\text{estFini}(M) \stackrel{\text{min}}{=} \left\{ \begin{array}{l} (\text{Personnage} : : \text{estVaincu}(\text{GestionCombat} : : \text{alex}(\text{combat}(M))) \\ \wedge \text{Personnage} : : \text{estVaincu}(\text{GestionCombat} : : \text{ryan}(\text{combat}(M)))) \\ \vee \text{Gangster} : : \text{estVaincu}(\text{GestionCombat} : : \text{slick}(\text{combat}(M))) \end{array} \right.$$

resultat(M) ^{min}	ALEXGAGNANT	si Personnage : :!estVaincu(GestionCombat : :alex(combat(M))) ∧ Gangster : :estVaincu(GestionCombat : :slick(combat(M))) ∧ Personnage : :estVaincu(GestionCombat : :ryan(combat(M)))
	RYANGAGNANT	si Personnage : :!estVaincu(GestionCombat : :ryan(combat(M))) ∧ Gangster : :estVaincu(GestionCombat : :slick(combat(M))) ∧ Personnage : :estVaincu(GestionCombat : :alex(combat(M)))
	DEUXGAGNANTS	si Personnage : :!estVaincu(GestionCombat : :ryan(combat(M))) ∧ Gangster : :estVaincu(GestionCombat : :slick(combat(M))) ∧ Personnage : :!estVaincu(GestionCombat : :alex(combat(M)))
	SLICKGAGNANT	si Personnage : :estVaincu(GestionCombat : :ryan(combat(M))) ∧ Gangster : :!estVaincu(GestionCombat : :slick(combat(M))) ∧ Personnage : :estVaincu(GestionCombat : :alex(combat(M)))
	NULLE	sinon

[init]

combat(init()) = GestionCombat::init()

[pasJeu]

combat(pasJeu(M, cA, cR)) = GestionCombat::gerer(combat(M), cA, cR)

1.7 GestionCombat

service : GestionCombat

use : Terrain, Personnage, Gangster

types : string, boolean, enum COMMANDE{RIEN, GAUCHE, DROITE, BAS, HAUT, FRAPPER, SAUT, SAUTHAUT, SAUTDROIT, SAUTGAUCHE, SAUTBAS, RAMASSER, JETER}

Observers :

terrain : [GestionCombat] → Terrain

alex : [GestionCombat] → Personnage

ryan : [GestionCombat] → Personnage

slick : [GestionCombat] → Gangster

gangsters : [GestionCombat] → Set<Gangster>

estGele : [GestionCombat] × Personnage → boolean

pre estGele(G, perso) require perso = alex(G) ∨ perso = ryan(G) ∨ perso = slick(G) ∨ perso ∈ gangsters(G)

estFrappe : [GestionCombat] × Personnage → boolean

pre estFrappe(G, perso) require perso = alex(G) ∨ perso = ryan(G) ∨ perso = slick(G) ∨ perso ∈ gangsters(G)

posX : [GestionCombat] × Personnage → int

pre posX(G, perso) require perso = alex(G) ∨ perso = ryan(G) ∨ perso = slick(G) ∨ perso ∈ gangsters(G)

posY : [GestionCombat] × Personnage → int

pre posY(G, perso) require perso = alex(G) ∨ perso = ryan(G) ∨ perso = slick(G) ∨ perso ∈ gangsters(G)

posZ : [GestionCombat] × Personnage → int

pre posZ(G, perso) require perso = alex(G) ∨ perso = ryan(G) ∨ perso = slick(G) ∨ perso ∈ gangsters(G)

collision : [GestionCombat] × Personnage × Personnage → boolean

pre collision(G, perso1, perso2) require

∨ (perso1 = alex(G) ∧ perso2 ∈ gangsters(G))

∨ (perso1 = alex(G) ∧ perso2 = slick(G))

$$\begin{aligned} & \vee (\text{perso1} = \text{ryan}(G) \wedge \text{perso2} \in \text{gangsters}(G)) \\ & \vee (\text{perso1} = \text{ryan}(G) \wedge \text{perso2} = \text{slick}(G)) \end{aligned}$$

Constructors :

$\text{init} : \emptyset \rightarrow [\text{GestionCombat}]$

Operators :

$\text{gerer} : [\text{GestionCombat}] \times \text{COMMANDE} \times \text{COMMANDE} \rightarrow [\text{GestionCombat}]$

Observations :

[Invariants]

$$\begin{aligned} 0 & \leq \text{posX}(G, s) \leq \text{Terrain}::\text{largeur}(\text{terrain}) \\ 0 & \leq \text{posY}(G, s) \leq \text{Terrain}::\text{profondeur}(\text{terrain}) \\ 0 & \leq \text{posZ}(G, s) \leq \text{Terrain}::\text{hauteur}(\text{terrain}) \\ \text{collision}(G, \text{perso1}, \text{perso2}) & \stackrel{\text{min}}{=} \text{collision}(G, \text{perso2}, \text{perso1}) \\ \text{collision}(G, p1, p2) & \stackrel{\text{min}}{=} \text{A FAIRE} \end{aligned}$$

[init]

$$\begin{aligned} \text{terrain}(\text{init}()) &= \text{Terrain}::\text{init}(1000, 1000, 1000) \\ \text{alex}(\text{init}()) &= \text{Personnage}::\text{init}(\text{"Alex"}, 10, 10, 10, 100, 100, 0) \\ \text{ryan}(\text{init}()) &= \text{Personnage}::\text{init}(\text{"Ryan"}, 10, 10, 10, 100, 100, 0) \\ \text{slick}(\text{init}()) &= \text{Gangster}::\text{init}(\text{"Slick"}, 10, 10, 10, 100, 100, 0) \\ \text{gangsters}(\text{init}()) &= \{g = \text{Personnage}::\text{init}(\text{"???"}, 10, 10, 10, 100, 100, 0)\}, \forall g \in \\ & \quad \text{gangsters}(G) \\ \text{estGele}(\text{init}(), s) &= \text{false} \\ \text{collision}(p1, p2) &= \text{false} \\ \text{estFrappe}(\text{init}(), s) &= \text{false} \\ \text{posX}(\text{init}(), \text{alex}(G)) &< 50 \\ \text{posX}(\text{init}(), \text{slick}(G)) &> \text{Terrain}::\text{largeur}(\text{terrain}(G)) - 50 \\ \text{posX}(\text{init}(), \text{ryan}(G)) &< 50 \\ \text{posZ}(\text{init}(), p) &= 0 \end{aligned}$$

$$\begin{aligned} \text{Bloc}::\text{type}(\text{Terrain}::\text{bloc}(\text{terrain}(G), \text{posX}(\text{init}(), g), \text{posY}(\text{init}(), g), \text{posZ}(\text{init}(), g))) \\ &= \text{VIDE} \forall g \in \text{gangsters}(G) \\ \text{Bloc}::\text{type}(\text{Terrain}::\text{bloc}(\text{terrain}(G), \text{posX}(\text{init}(), \text{slick}(G)), \text{posY}(\text{init}(), \text{slick}(G)), \\ & \quad \text{posZ}(\text{init}(), \text{slick}(G)))) = \text{VIDE} \\ \text{Bloc}::\text{type}(\text{Terrain}::\text{bloc}(\text{terrain}(G), \text{posX}(\text{init}(), \text{alex}(G)), \text{posY}(\text{init}(), \text{alex}(G)), \text{posZ}(\text{init}(), \\ & \quad \text{alex}(G)))) \neq \text{FOSSE} \\ \text{Bloc}::\text{type}(\text{Terrain}::\text{bloc}(\text{terrain}(G), \text{posX}(\text{init}(), \text{ryan}(G)), \text{posY}(\text{init}(), \text{ryan}(G)), \text{posZ}(\text{init}(), \\ & \quad \text{ryan}(G)))) \neq \text{FOSSE} \end{aligned}$$

[gerer]

$$\begin{aligned} \text{posX}(\text{gerer}(G, cA, cR), \text{alex}(G)) &= \\ \begin{cases} \text{posX}(G, \text{alex}(G)) + 10 & \text{si } cA = \text{DROIT} \vee cA = \text{SAUTDROIT} \\ \text{posX}(G, \text{alex}(G)) - 10 & \text{si } cA = \text{GAUCHE} \vee cA = \text{SAUTGAUCHE} \\ \text{posX}(G, \text{alex}(G)) & \text{sinon} \end{cases} \end{aligned}$$

$$\begin{aligned} \text{posY}(\text{gerer}(G, cA, cR), \text{alex}(G)) &= \\ \begin{cases} \text{posY}(G, \text{alex}(G)) + 10 & \text{si } cA = \text{HAUT} \vee cA = \text{SAUTHAUT} \\ \text{posY}(G, \text{alex}(G)) - 10 & \text{si } cA = \text{BAS} \vee cA = \text{SAUTBAS} \\ \text{posY}(G, \text{alex}(G)) & \text{sinon} \end{cases} \end{aligned}$$

$$\text{posZ}(\text{gerer}(G, cA, cR), \text{alex}(G)) =$$

$$\begin{cases} 10 & \text{si } cA = \text{SAUT} \vee cA = \text{SAUTBAS} \vee cA = \text{SAUTHAUT} \vee cA = \text{SAUTDROIT} \vee cA = \text{SAUTGAUCHE} \\ 0 & \text{Sinon} \end{cases}$$

$$\text{posX}(\text{gerer}(G, cA, cR), \text{ryan}(G)) = \begin{cases} \text{posX}(G, \text{ryan}(G)) + 10 & \text{si } cR = \text{DROIT} \vee cR = \text{SAUTDROIT} \\ \text{posX}(G, \text{ryan}(G)) - 10 & \text{si } cR = \text{GAUCHE} \vee cR = \text{SAUTGAUCHE} \\ \text{posX}(G, \text{ryan}(G)) & \text{sinon} \end{cases}$$

$$\text{posY}(\text{gerer}(G, cA, cR), \text{ryan}(G)) = \begin{cases} \text{posY}(G, \text{ryan}(G)) + 10 & \text{si } cR = \text{HAUT} \vee cR = \text{SAUTHAUT} \\ \text{posY}(G, \text{ryan}(G)) - 10 & \text{si } cR = \text{BAS} \vee cR = \text{SAUTBAS} \\ \text{posY}(G, \text{ryan}(G)) & \text{sinon} \end{cases}$$

$$\text{posZ}(\text{gerer}(G, cA, cR), \text{ryan}(G)) = \begin{cases} 10 & \text{si } cR = \text{SAUT} \vee cR = \text{SAUTBAS} \vee cR = \text{SAUTHAUT} \vee cR = \text{SAUTDROIT} \vee cR = \text{SAUTGAUCHE} \\ 0 & \text{Sinon} \end{cases}$$

$$\text{alex}(\text{gerer}(G, cA, cR)) = \begin{cases} - \text{Personnage} : \text{jeter}(\text{alex}(G)) \text{ si } cA = \text{JETER} \\ - \text{Personnage} : \text{ramasser_objet}(\text{alex}(G), \text{Bloc} : \text{objet}(\text{Terrain} : \text{bloc}(\text{terrain}(G), \text{posX}(\text{alex}(G)), \\ \text{posY}(\text{alex}(G)), \text{posZ}(\text{alex}(G)))) \text{ si } cA = \text{RAMASSER} \\ - \text{Personnage} : \text{ramasser_perso}(\text{alex}(G), p) \text{ si } \text{collision}(\text{alex}(G), p) \wedge cA = \text{RAMASSER} \\ - \text{Personnage} : \text{retrait_vie}(\text{alex}(G), \text{Personnage} : \text{force}(p)) \text{ si } \text{collision}(\text{alex}(G), p) \wedge \text{Gangster} : \text{action}(p) = \text{FRAPPER} \\ - \text{alex}(G) \text{ Sinon} \end{cases}$$

$$\begin{aligned} \text{posX}(\text{gerer}(G, cA, cR), p) &= \text{posX}(G, \text{alex}(G)) + 10 \text{ si } cA = \text{JETER} \wedge \text{Personnage} : \text{perso_equipe}(\text{alex}()) = p \\ &\quad \text{posX}(G, p) \text{ sinon} \\ \text{posY}(\text{gerer}(G, cA, cR), p) &= \text{posY}(G, \text{alex}(G)) \text{ si } cA = \text{JETER} \wedge \text{Personnage} : \text{perso_equipe}(\text{alex}()) = p \\ &\quad \text{posY}(G, p) \text{ sinon} \\ \text{posZ}(\text{gerer}(G, cA, cR), p) &= 0 \text{ si } cA = \text{JETER} \wedge \text{Personnage} : \text{perso_equipe}(\text{alex}()) = p \\ &\quad \text{posZ}(G, p) \text{ sinon} \end{aligned}$$

$$\text{slick}(\text{gerer}(G, cA, cR)) = \begin{cases} - \text{Gangster} : \text{retrait_vie}(\text{slick}(G), \text{Personnage} : \text{force}(h)) \text{ si } \text{collison}(\text{alex}(G), \text{slick}(G)) \wedge cA = \text{FRAPPER} \\ - \text{Gangster} : \text{retrait_vie}(\text{slick}(G), \text{Personnage} : \text{force}(h)) \text{ si } \text{collison}(\text{ryan}(G), \text{slick}(G)) \wedge cR = \text{FRAPPER} \\ - \text{slick}(G) \text{ sinon} \end{cases}$$

$$\text{ryan}(\text{gerer}(G, cA, cR)) = \begin{cases} - \text{Personnage} : \text{jeter}(\text{ryan}(G)) \text{ si } cR = \text{JETER} \\ - \text{Personnage} : \text{ramasser_objet}(\text{ryan}(G), \text{Bloc} : \text{objet}(\text{Terrain} : \text{bloc}(\text{terrain}(G), \text{posX}(\text{ryan}(G)), \\ \text{posY}(\text{ryan}(G)), \text{posZ}(\text{ryan}(G)))) \text{ si } cR = \text{RAMASSER} \\ - \text{Personnage} : \text{ramasser_perso}(\text{ryan}(G), p) \text{ si } \text{collision}(\text{ryan}(G), p) \wedge cR = \text{RAMASSER} \\ - \text{Personnage} : \text{retrait_vie}(\text{ryan}(G), \text{Personnage} : \text{force}(p)) \text{ si } \text{collision}(\text{ryan}(G), p) \wedge \text{Gangster} : \text{action}(p) = \text{FRAPPER} \\ - \text{ryan}(G) \text{ Sinon} \end{cases}$$

$$\text{posX}(\text{gerer}(G, cA, cR), p) = \text{posX}(\text{ryan}(G)) + 10 \text{ si } cR = \text{JETER} \wedge \text{Personnage} : \text{perso_equipe}(\text{ryan}()) = p \\ \text{posX}(G, p) \text{ sinon}$$

$$\text{posY}(\text{gerer}(G, cA, cR), p) = \text{posY}(\text{ryan}(G)) \text{ si } cR = \text{JETER} \wedge \text{Personnage} : \text{perso_equipe}(\text{ryan}()) = p$$


```

posY(G,p) sinon

posZ(gerer(G,cA,cR),p) =
  0 si cR = JETER ∧ Personnage::perso_equipe(ryan()) = p
  posZ(G,p) sinon

terrain(gerer(G,cA,cR)) =
- Bloc::retirerObjet(Terrain::bloc(terrain(G), posX(alex(G)), posY(alex(G)),posZ(
  alex(G))) si cA = RAMASSER
- Bloc::poserObjet(Terrain::bloc(terrain(G), posX(alex(G)), posY(alex(G)),posZ(
  alex(G))), Personnage:objet_equipe(alex()) si cA = JETER ∧ Personnage::
  est_equipe_objet(alex()) = true
- Bloc::retirerObjet(Terrain::bloc(terrain(G), posX(ryan(G)), posY(ryan(G)),posZ(
  ryan(G))) si cR = RAMASSER
- Bloc::poserObjet(Terrain::bloc(terrain(G), posX(ryan(G)), posY(ryan(G)),posZ(
  ryan(G))), Personnage:objet_equipe(ryan()) si cR = JETER ∧ Personnage::
  est_equipe_objet(ryan()) = true
- terrain(G) sinon // Faut il faire aussi les deux ?????

```