

# 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

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

```

depot_argent : [Personnage] × int → [Personnage]
  pre depot_argent(P,s) require ¬est_vaincu(P) ∧ s>0

ramasser_argent : [Personnage] × Object → [Personnage]
  pre ramasser_argent(P,o) require ¬est_vaincu(P) ∧ Objet::est_de_valeur(o)

ramasser_objet : [Personnage] × Object → [Personnage]
  pre ramasser_objet(P,o) require ¬est_vaincu(P) ∧ ¬est_equipe_objet(P) ∧ ¬
    est_equipe_perso(P) ∧ Objet::est_equipable(o)

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

```

[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)

```

[ramasser\_argent]

```

somme_d_argent(ramasser_argent(P,objet)) = somme_d_argent(P) + Objet::
  valeur_marchande(objet)

```

[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)
  { 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 → [Gangster]
  pre init(nom, largeur, hauteur, profondeur, force, pdv) require nom ≠ "" ∧ largeur
    >0 ∧ hauteur>0 ∧ profondeur>0 ∧ force>0 ∧ pdv>0

```

```

Observations :
[init]
  nom(init(n,l,h,p,f,v))=n
  largeur(init(n,l,h,p,f,v))=l
  hauteur(init(n,l,h,p,f,v))=h
  profondeur(init(n,l,h,p,f,v))=p
  force(init(n,l,h,p,f,v))=f
  points_de_vie(init(n,l,h,p,f,v))=v
  somme_d_argent(init(n,l,h,p,f,v))=0
  objet_equipe(init(n,l,h,p,f,v))=null
  perso_equipe(init(n,l,h,p,f,v))=null
  action(init(n,l,h,p,f,v)) = RIEN

[retrait_argent]
  somme_d_argent(retrait_argent(G,s)) = argent(G)

[depot_argent]
  somme_d_argent(depot_argent(G,s)) = argent(G)

[ramasser_argent]
  somme_d_argent(ramasser_objet(G, objet)) = somme_d_argent(G)

```

## 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=OBJET ∧ o≠null)

```

**Operators :**

```

retirerObjet : [Bloc] → [Bloc]
  pre retirerObjet(B) require type(B)=OBJET
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 → Bloc
  pre bloc( T, x,y) require 0 ≤ x ≤ largeur ∧ 0 ≤ y ≤ profondeur

```

Constructors :

```

init : int × int × int → [Terrain]
  pre init(largeur, hauteur, prof) require largeur > 0 ∧ hauteur > 0 ∧ prof > 0
    ∧ largeur%50=0 ∧ profondeur%50=0

```

Operators :

```

modifier_bloc : [Terrain] × int × int × Bloc → [Terrain]
  pre bloc( T, x, y, b) require 0 ≤ x ≤ largeur ∧ 0 ≤ y ≤ 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) ≠ NULL

```

[modifier\_bloc]

```

bloc(modifier_bloc(T, x, y, b), x, y) = 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]

```

estFini(M)  $\stackrel{min}{=}$  (Personnage:: estVaincu(GestionCombat:: alex(combat(M)))
  ∧ Personnage:: estVaincu(GestionCombat:: ryan(combat(M)))
  ∨ Gangster:: estVaincu(GestionCombat:: slick(combat(M)))

```

$$\text{resultat}(M) \stackrel{\text{min}}{=} \left\{ \begin{array}{ll} \text{ALEXGAGNANT} & \begin{array}{l} \text{si } \neg \text{Personnage} : \text{estVaincu}(\text{GestionCombat} : \text{:alex}(\text{combat}(M))) \\ \wedge \text{Gangster} : \text{estVaincu}(\text{GestionCombat} : \text{:slick}(\text{combat}(M))) \\ \wedge \text{Personnage} : \text{estVaincu}(\text{GestionCombat} : \text{:ryan}(\text{combat}(M))) \end{array} \\ \text{RYANGAGNANT} & \begin{array}{l} \text{si } \neg \text{Personnage} : \text{estVaincu}(\text{GestionCombat} : \text{:ryan}(\text{combat}(M))) \\ \wedge \text{Gangster} : \text{estVaincu}(\text{GestionCombat} : \text{:slick}(\text{combat}(M))) \\ \wedge \text{Personnage} : \text{estVaincu}(\text{GestionCombat} : \text{:alex}(\text{combat}(M))) \end{array} \\ \text{DEUXGAGNANTS} & \begin{array}{l} \text{si } \neg \text{Personnage} : \text{estVaincu}(\text{GestionCombat} : \text{:ryan}(\text{combat}(M))) \\ \wedge \text{Gangster} : \text{estVaincu}(\text{GestionCombat} : \text{:slick}(\text{combat}(M))) \\ \wedge \neg \text{Personnage} : \text{estVaincu}(\text{GestionCombat} : \text{:alex}(\text{combat}(M))) \end{array} \\ \text{SLICKGAGNANT} & \begin{array}{l} \text{si } \text{Personnage} : \text{estVaincu}(\text{GestionCombat} : \text{:ryan}(\text{combat}(M))) \\ \wedge \neg \text{Gangster} : \text{estVaincu}(\text{GestionCombat} : \text{:slick}(\text{combat}(M))) \\ \wedge \text{Personnage} : \text{estVaincu}(\text{GestionCombat} : \text{:alex}(\text{combat}(M))) \end{array} \\ \text{NULLE} & \text{sinon} \end{array} \right.$$

[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>

actionGangster : [GestionCombat] (\$\*times\$\*) Gangster → ACTION

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)

collisionDroite : [GestionCombat] × Personnage × Gangster → boolean

pre collisionDroite(G, perso1, perso2) require (perso1 = alex(G) ∧ perso2 ∈ gangsters(G))

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

```

    ∨ (perso1 = ryan(G) ∧ perso2 ∈ gangsters(G))
    ∨ (perso1 = ryan(G) ∧ perso2 = slick(G))
collisionGauche : [GestionCombat] × Personnage × Gangster → boolean
    pre collisionGauche(G, perso1, perso2) require (perso1 = alex(G) ∧ perso2
        ∈ gangsters(G))
    ∨ (perso1 = alex(G) ∧ perso2 = slick(G))
    ∨ (perso1 = ryan(G) ∧ perso2 ∈ gangsters(G))
    ∨ (perso1 = ryan(G) ∧ perso2 = slick(G))
collisionDevant : [GestionCombat] × Personnage × Gangster → boolean
    pre collisionDevant(G, perso1, perso2) require (perso1 = alex(G) ∧ perso2
        ∈ gangsters(G))
    ∨ (perso1 = alex(G) ∧ perso2 = slick(G))
    ∨ (perso1 = ryan(G) ∧ perso2 ∈ gangsters(G))
    ∨ (perso1 = ryan(G) ∧ perso2 = slick(G))
collisionDerriere : [GestionCombat] × Personnage × Gangster → boolean
    pre collisionDerriere(G, perso1, perso2) require (perso1 = alex(G) ∧
        perso2 ∈ gangsters(G))
    ∨ (perso1 = alex(G) ∧ perso2 = slick(G))
    ∨ (perso1 = ryan(G) ∧ perso2 ∈ gangsters(G))
    ∨ (perso1 = ryan(G) ∧ perso2 = slick(G))
collisionDessus : [GestionCombat] × Personnage × Gangster → boolean
    pre collisionDessus(G, perso1, perso2) require (perso1 = alex(G) ∧ perso2
        ∈ gangsters(G))
    ∨ (perso1 = alex(G) ∧ perso2 = slick(G))
    ∨ (perso1 = ryan(G) ∧ perso2 ∈ gangsters(G))
    ∨ (perso1 = ryan(G) ∧ perso2 = slick(G))
collisionDessous : [GestionCombat] × Personnage × Gangster → boolean
    pre collisionDessous(G, perso1, perso2) require (perso1 = alex(G) ∧
        perso2 ∈ gangsters(G))
    ∨ (perso1 = alex(G) ∧ perso2 = slick(G))
    ∨ (perso1 = ryan(G) ∧ perso2 ∈ gangsters(G))
    ∨ (perso1 = ryan(G) ∧ perso2 = slick(G))
collision : [GestionCombat] × Personnage × Gangster → boolean
    pre collision(G, perso1, perso2) require
        (perso1 = alex(G) ∧ perso2 ∈ gangsters(G))
    ∨ (perso1 = alex(G) ∧ perso2 = slick(G))
    ∨ (perso1 = ryan(G) ∧ perso2 ∈ gangsters(G))
    ∨ (perso1 = ryan(G) ∧ perso2 = slick(G))

```

#### Constructors :

```
init : ∅ → [GestionCombat]
```

#### Operators :

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

#### Observations :

##### [Invariants]

0 ≤ posX(G,s) ≤ Terrain::largeur(terrain)

0 ≤ posY(G,s) ≤ Terrain::profondeur(terrain)

0 ≤ posZ(G,s) ≤ Terrain::hauteur(terrain)

collisionDroite(G,p1,p2)  $\stackrel{min}{=}$  ( -d ≤ posX(G,p1) - posX(G,p2) ≤ d+1 ) ∧ ( d =  
Personnage::largeur(p1)/2 + d = Personnage::largeur(p2)/2 )

collisionGauche(G,p1,p2)  $\stackrel{min}{=}$  ( -d ≤ posX(G,p2) - posX(G,p1) ≤ d+1 ) ∧ ( d =  
Personnage::largeur(p1)/2 + d = Personnage::largeur(p2)/2 )

$\text{collisionDevant}(G, p1, p2) \stackrel{\text{min}}{=} ( -d \leq \text{posY}(G, p1) - \text{posY}(G, p2) \leq d+1) \wedge ( d = \text{Personnage}::\text{profondeur}(p1)/2 + d = \text{Personnage}::\text{profondeur}(p2)/2 )$

$\text{collisionDerriere}(G, p1, p2) \stackrel{\text{min}}{=} ( -d \leq \text{posY}(G, p2) - \text{posY}(G, p1) \leq d+1) \wedge ( d = \text{Personnage}::\text{profondeur}(p1)/2 + d = \text{Personnage}::\text{profondeur}(p2)/2 )$

$\text{collisionDessous}(G, p1, p2) \stackrel{\text{min}}{=} ( -d \leq \text{posZ}(G, p1) - \text{posZ}(G, p2) \leq d+1) \wedge ( d = \text{Personnage}::\text{hauteur}(p1)/2 + d = \text{Personnage}::\text{hauteur}(p2)/2 )$

$\text{collisionDessus}(G, p1, p2) \stackrel{\text{min}}{=} ( -d \leq \text{posZ}(G, p2) - \text{posZ}(G, p1) \leq d+1) \wedge ( d = \text{Personnage}::\text{hauteur}(p1)/2 + d = \text{Personnage}::\text{hauteur}(p2)/2 )$

$\text{collision}(G, p1, p2) \stackrel{\text{min}}{=} \text{collisionDroite}(G, p1, p2) \vee \text{collisionGauche}(G, p1, p2) \vee \text{collisionDevant}(G, p1, p2) \vee \text{collisionDerriere}(G, p1, p2) \vee \text{collisionDessous}(G, p1, p2) \vee \text{collisionDessus}(G, p1, p2)$

**[init]**

```

terrain(init()) = Terrain::init(1000,1000,1000)
alex(init()) = Personnage::init("Alex",10,10,10,100,100,0)
ryan(init()) = Personnage::init("Ryan",10,10,10,100,100,0)
slick(init()) = Gangster::init("Slick",10,10,10,100,100,0)
gangsters(init()) = {g = Personnage::init("???",10,10,10,100,100,0)},  $\forall g \in$ 
  gangsters(G)
estGele(init(), s) = false
  collisionGauche(init(), p1, p2) = false
  collisionDroite(init(), p1, p2) = false
  collisionDevant(init(), p1, p2) = false
  collisionDerriere(init(), p1, p2) = false
  collisionDessous(init(), p1, p2) = false
  collisionDessus(init(), p1, p2) = false
collision(init(), p1, p2) = false
estFrappe(init(), s) = false
posX(init(), alex(G)) < 50
posX(init(), slick(G)) > Terrain::largeur(terrain(G))-50
posX(init(), ryan(G)) < 50
posZ(init(), p) = 0

```

```

Bloc::type(Terrain:bloc(terrain(G),posX(init(),g),posY(init(),g),posZ(init(),g)))
  = VIDE  $\forall g \in$  gangsters(G)
Bloc::type(Terrain:bloc(terrain(G),posX(init(),slick(G)),posY(init(),slick(G)),
  posZ(init(),slick(G)))) = VIDE
Bloc::type(Terrain:bloc(terrain(G),posX(init(),alex(G)),posY(init(),alex(G)),posZ(
  init(),alex(G))))  $\neq$  FOSSE
Bloc::type(Terrain:bloc(terrain(G),posX(init(),ryan(G)),posY(init(),ryan(G)),posZ(
  init(),ryan(G))))  $\neq$  FOSSE

```

**[gerer]**

```

posX(gerer(G,cA,cR), alex(G)) =
  {
    posX(G,alex(G)) + 10    si cA = DROIT  $\vee$  cA = SAUTDROIT  $\wedge$   $\neg$ Personnage::estVaincu(alex(G))
    posX(G,alex(G)) - 10    si cA = GAUCHE  $\vee$  cA = SAUTGAUCHE  $\wedge$   $\neg$ Personnage::estVaincu(alex(G))
    posX(G,alex(G))         sinon
  }

```

```

posY(gerer(G,cA,cR), alex(G)) =

```



$$\begin{cases} \text{posY}(G, \text{alex}(G)) + 10 & \text{si } cA = \text{HAUT} \vee cA = \text{SAUTHAUT} \wedge \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \\ \text{posY}(G, \text{alex}(G)) - 10 & \text{si } cA = \text{BAS} \vee cA = \text{SAUTBAS} \wedge \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \\ \text{posY}(G, \text{alex}(G)) & \text{sinon} \end{cases}$$

$$\text{posZ}(\text{gerer}(G, cA, cR), \text{alex}(G)) = \begin{cases} 10 \text{ si } cA = \text{SAUT} \vee cA = \text{SAUTBAS} \vee cA = \text{SAUTHAUT} \vee cA = \text{SAUTDROIT} \vee cA = \text{SAUTGAUCHE} \\ \wedge \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \\ 0 \text{ Sinon} \end{cases}$$

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

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

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

$$\text{alex}(\text{gerer}(G, cA, cR)) = \begin{cases} - \text{Personnage} : \text{:jeter}(\text{alex}(G)) \text{ si } cA = \text{JETER} \wedge \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \wedge \text{Bloc} : \text{:type}(\text{Terrain} : \text{:bloc}(\text{terrain}(G), \\ - \text{Personnage} : \text{:ramasser\_objet}(\text{alex}(G), \text{Bloc} : \text{:objet}(\text{Terrain} : \text{:bloc}(\text{terrain}(G), \text{posX}(\text{alex}(G)), \text{posY}(\text{alex}(G)))))) \text{ si } cA = \text{RAMASSER} \\ - \text{Personnage} : \text{:ramasser\_perso}(\text{alex}(G), p) \\ \text{si collision}(\text{alex}(G), p) \wedge cA = \text{RAMASSER} \wedge \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \\ - \text{Personnage} : \text{:ramasser\_argent}(\text{alex}(G), \text{Bloc} : \text{:objet}(\text{Terrain} : \text{:bloc}(\text{terrain}(G), \text{posX}(\text{alex}(G)), \text{posY}(\text{alex}(G)))))) \text{ si } cA = \text{RAMASSER\_ARGENT} \\ - \text{Personnage} : \text{:retrait\_vie}(\text{alex}(G), \text{Personnage} : \text{:force}(p)) \\ \text{si collision}(G, \text{alex}(G), p) \wedge \text{actionGangster}(G, p) = \text{FRAPPER} \wedge \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \\ - \text{Personnage} : \text{:retrait\_vie}(\text{alex}(G), \text{Personnage} : \text{:points\_de\_vie}(\text{alex}(G)) \\ \text{si Bloc} : \text{:type}(\text{Terrain} : \text{:bloc}(\text{terrain}(G), \text{posX}(\text{alex}(G)), \text{posY}(\text{alex}(G)), \text{posZ}(\text{alex}(G)))) = \text{FOSSE} \\ \wedge \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \\ - \text{alex}(G) \text{ Sinon} \end{cases}$$

$$\text{posX}(\text{gerer}(G, cA, cR), p) = \begin{cases} \text{posX}(G, \text{alex}(G)) + 10 & \text{si } cA = \text{JETER} \wedge \text{Personnage} : \text{:perso\_equipe}(\text{alex}()) = p \wedge \\ \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \\ \text{posX}(G, p) & \text{sinon} \end{cases}$$

$$\text{posY}(\text{gerer}(G, cA, cR), p) = \begin{cases} \text{posY}(G, \text{alex}(G)) & \text{si } cA = \text{JETER} \wedge \text{Personnage} : \text{:perso\_equipe}(\text{alex}()) = p \wedge \\ \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \\ \text{posY}(G, p) & \text{sinon} \end{cases}$$

$$\text{posZ}(\text{gerer}(G, cA, cR), p) = \begin{cases} 0 & \text{si } cA = \text{JETER} \wedge \text{Personnage} : \text{:perso\_equipe}(\text{alex}()) = p \wedge \neg \text{Personnage} : \text{:estVaincu}(\text{alex}(G)) \\ \text{posZ}(G, p) & \text{sinon} \end{cases}$$

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

$$\left\{ \begin{array}{l} - \text{Gangster} : \text{retrait\_vie}(\text{slick}(\text{G}), \text{Personnage} : \text{force}(\text{alex}(\text{G}))) \\ \quad \text{si } \text{collision}(\text{alex}(\text{G}), \text{slick}(\text{G})) \wedge \text{cA} = \text{FRAPPER} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{alex}(\text{G})) \\ - \text{Gangster} : \text{retrait\_vie}(\text{slick}(\text{G}), \text{Personnage} : \text{force}(\text{ryan}(\text{G}))) \\ \quad \text{si } \text{collision}(\text{ryan}(\text{G}), \text{slick}(\text{G})) \wedge \text{cR} = \text{FRAPPER} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ - \text{slick}(\text{G}) \text{ sinon} \end{array} \right.$$

$$\text{ryan}(\text{gerer}(\text{G}, \text{cA}, \text{cR})) = \left\{ \begin{array}{l} - \text{Personnage} : \text{jeter}(\text{ryan}(\text{G})) \text{ si } \text{cR} = \text{JETER} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ - \text{Personnage} : \text{ramasser\_objet}(\text{ryan}(\text{G}), \text{Bloc} : \text{objet}(\text{Terrain} : \text{bloc}(\text{terrain}(\text{G}), \text{posX}(\text{ryan}(\text{G})), \\ \text{posY}(\text{ryan}(\text{G})), \text{posZ}(\text{ryan}(\text{G})))) \text{ si } \text{cR} = \text{RAMASSER} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ - \text{Personnage} : \text{ramasser\_perso}(\text{ryan}(\text{G}), \text{p}) \\ \quad \text{si } \text{collision}(\text{ryan}(\text{G}), \text{p}) \wedge \text{cR} = \text{RAMASSER} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ - \text{Personnage} : \text{retrait\_vie}(\text{ryan}(\text{G}), \text{Personnage} : \text{force}(\text{p})) \\ \quad \text{si } \text{collision}(\text{ryan}(\text{G}), \text{p}) \wedge \text{Gangster} : \text{action}(\text{p}) = \text{FRAPPER} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ - \text{Personnage} : \text{retrait\_vie}(\text{ryan}(\text{G}), \text{Personnage} : \text{points\_de\_vie}(\text{ryan}(\text{G}))) \\ \quad \text{si } \text{Bloc} : \text{type}(\text{Terrain} : \text{bloc}(\text{terrain}(\text{G}), \text{posX}(\text{ryan}(\text{G})), \text{posY}(\text{ryan}(\text{G})), \text{posZ}(\text{ryan}(\text{G})))) = \text{FOSSE} \\ \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ - \text{ryan}(\text{G}) \text{ Sinon} \end{array} \right.$$

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

$$\text{posY}(\text{gerer}(\text{G}, \text{cA}, \text{cR}), \text{p}) = \text{posY}(\text{ryan}(\text{G})) \text{ si } \text{cR} = \text{JETER} \wedge \text{Personnage} : \text{perso\_equipe}(\text{ryan}()) = \text{p} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ \text{posY}(\text{G}, \text{p}) \text{ sinon}$$

$$\text{posZ}(\text{gerer}(\text{G}, \text{cA}, \text{cR}), \text{p}) = 0 \text{ si } \text{cR} = \text{JETER} \wedge \text{Personnage} : \text{perso\_equipe}(\text{ryan}()) = \text{p} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ \text{posZ}(\text{G}, \text{p}) \text{ sinon}$$

$$\text{terrain}(\text{gerer}(\text{G}, \text{cA}, \text{cR})) = \begin{array}{l} - \text{Bloc} : \text{retirerObjet}(\text{Terrain} : \text{bloc}(\text{terrain}(\text{G}), \text{posX}(\text{alex}(\text{G})), \text{posY}(\text{alex}(\text{G})), \text{posZ}(\text{alex}(\text{G}))) \\ \quad \text{si } \text{cA} = \text{RAMASSER} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{alex}(\text{G})) \\ - \text{Bloc} : \text{poserObjet}(\text{Terrain} : \text{bloc}(\text{terrain}(\text{G}), \text{posX}(\text{alex}(\text{G})), \text{posY}(\text{alex}(\text{G})), \text{posZ}(\text{alex}(\text{G}))), \text{Personnage} : \text{objet\_equipe}(\text{alex}()) \\ \quad \text{si } \text{cA} = \text{JETER} \wedge \text{Personnage} : \text{est\_equipe\_objet}(\text{alex}()) = \text{true} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{alex}(\text{G})) \\ - \text{Bloc} : \text{retirerObjet}(\text{Terrain} : \text{bloc}(\text{terrain}(\text{G}), \text{posX}(\text{ryan}(\text{G})), \text{posY}(\text{ryan}(\text{G})), \text{posZ}(\text{ryan}(\text{G}))) \\ \quad \text{si } \text{cR} = \text{RAMASSER} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ - \text{Bloc} : \text{poserObjet}(\text{Terrain} : \text{bloc}(\text{terrain}(\text{G}), \text{posX}(\text{ryan}(\text{G})), \text{posY}(\text{ryan}(\text{G})), \text{posZ}(\text{ryan}(\text{G}))), \text{Personnage} : \text{objet\_equipe}(\text{ryan}()) \\ \quad \text{si } \text{cR} = \text{JETER} \wedge \text{Personnage} : \text{est\_equipe\_objet}(\text{ryan}()) = \text{true} \wedge \neg \text{Personnage} : \text{estVaincu}(\text{ryan}(\text{G})) \\ - \text{terrain}(\text{G}) \text{ sinon} \end{array}$$