

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
est_visible(P) = true
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)) = max(0, 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

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

```

```

[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},

Observators :

```

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 > 50 ∧ hauteur > 100 ∧ prof
    > 50 ∧ 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:: est_vaincu(GestionCombat:: alex(combat(M)))
    ∧ Personnage:: est_vaincu(GestionCombat:: ryan(combat(M)))
    ∨ Gangster:: est_vaincu(GestionCombat:: slick(combat(M)))
```

resultat(M) ^{min}	ALEXGAGNANT	si \neg Personnage :est_vaincu(GestionCombat : :alex(combat(M))) \wedge Gangster :est_vaincu(GestionCombat : :slick(combat(M))) \wedge Personnage :est_vaincu(GestionCombat : :ryan(combat(M)))
	RYANGAGNANT	si \neg Personnage :est_vaincu(GestionCombat : :ryan(combat(M))) \wedge Gangster :est_vaincu(GestionCombat : :slick(combat(M))) \wedge Personnage :est_vaincu(GestionCombat : :alex(combat(M)))
	DEUXGAGNANTS	si \neg Personnage :est_vaincu(GestionCombat : :ryan(combat(M))) \wedge Gangster :est_vaincu(GestionCombat : :slick(combat(M))) \wedge \neg Personnage :est_vaincu(GestionCombat : :alex(combat(M)))
	SLICKGAGNANT	si Personnage :est_vaincu(GestionCombat : :ryan(combat(M))) \wedge \neg Gangster :est_vaincu(GestionCombat : :slick(combat(M))) \wedge Personnage :est_vaincu(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, SAUTDROITE, SAUTGAUCHE, SAUTBAS, RAMASSER, JETER}

```

```

Observers :
    terrain : [GestionCombat] → Terrain

    alex : [GestionCombat] → Personnage

    ryan : [GestionCombat] → Personnage

    slick : [GestionCombat] → Gangster

    gangsters : [GestionCombat] → List<Gangster>

    actionGangster : [GestionCombat] × Gangster → COMMANDE

    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)

    estVisible : [GestionCombat] × Personnage → boolean
    pre estVisible(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) ∨ perso1 = ryan(G)) ∧ (perso2 = slick(G) ∨ perso2 ∈
gangsters(G))
```

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

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

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

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

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

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

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

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

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

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

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

actionGangster(G,g) = RIEN si estGele(G,g) ∨ est_vaincu(G,g) ∀ g ∈ gangsters(G)

[init]

terrain(init()) = Terrain::init(1000,1000,1000)

alex(init()) = Personnage::init("Alex",30,30,30,100,100,0)

ryan(init()) = Personnage::init("Ryan",30,30,30,100,100,0)

slick(init()) = Gangster::init("Slick",50,50,50,100,100)

gangsters(init()) = {g = Personnage::init("noname",20,20,20,10,50)}, ∀ g ∈
  gangsters(G)

actionGangster(G,g) = RIEN ∀ g ∈ 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

posY (init () , perso) = random

Bloc :: type (Terrain : bloc (terrain (G) , posX (init () , g) , posY (init () , g) , posZ (init () , g)))
= VIDE $\forall g \in \text{gangsters}(G)$

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)) =

$$\begin{cases} \min(\text{posX}(G, \text{alex}(G)) + 10, \text{Terrain} : \text{largeur}(\text{terrain}(G)) - \text{Personnage} : \text{largeur}(\text{alex}(G))) \\ \text{si } cA = \text{DROITE} \vee cA = \text{SAUTDROITE} \wedge \neg \text{Personnage} : \text{est_vaincu}(\text{alex}(G)) \wedge \neg \text{estGele}(G, \text{alex}(G)) \\ \wedge \neg \text{collisionDroite}(\text{alex}(G), p) \forall p \in \text{gangster} \vee p = \text{slick}(G) \\ \max(\text{posX}(G, \text{alex}(G)) - 10, 0) \\ \text{si } cA = \text{GAUCHE} \vee cA = \text{SAUTGAUCHE} \wedge \neg \text{Personnage} : \text{est_vaincu}(\text{alex}(G)) \wedge \neg \text{estGele}(G, \text{alex}(G)) \\ \wedge \neg \text{collisionGauche}(\text{alex}(G), p) \forall p \in \text{gangster} \vee p = \text{slick}(G) \\ \text{posX}(G, \text{alex}(G)) \text{ sinon} \end{cases}$$

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

$$\begin{cases} \min(\text{posY}(G, \text{alex}(G)) + 10, \text{Terrain} : \text{profondeur}(\text{terrain}(G)) - \text{Personnage} : \text{profondeur}(\text{alex}(G))) \\ \text{si } cA = \text{HAUT} \vee cA = \text{SAUTHAUT} \wedge \neg \text{Personnage} : \text{est_vaincu}(\text{alex}(G)) \wedge \neg \text{estGele}(G, \text{alex}(G)) \wedge \neg \text{collisionDerriere}(\text{alex}(G), p) \forall p \in \text{gangster} \vee p = \text{slick}(G) \\ \max(\text{posY}(G, \text{alex}(G)) - 10, 0) \\ \text{si } cA = \text{BAS} \vee cA = \text{SAUTBAS} \wedge \neg \text{Personnage} : \text{est_vaincu}(\text{alex}(G)) \wedge \neg \text{estGele}(G, \text{alex}(G)) \wedge \neg \text{collisionDevant}(\text{alex}(G), p) \forall p \in \text{gangster} \vee p = \text{slick}(G) \\ \text{posY}(G, \text{alex}(G)) \text{ sinon} \end{cases}$$

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

$$\begin{cases} 100 & \text{si } cA = \text{SAUT} \vee cA = \text{SAUTBAS} \vee cA = \text{SAUTHAUT} \vee cA = \text{SAUTDROITE} \vee cA = \text{SAUTGAUCHE} \\ & \wedge \neg \text{Personnage} : \text{est_vaincu}(\text{alex}(G)) \wedge \neg \text{estGele}(G, \text{alex}(G)) \wedge \neg \text{collisionDessus}(\text{alex}(G), p) \forall p \in \text{gangster} \vee p = \text{slick}(G) \\ \text{pos}(G, \text{alex}(G)) & \text{si } \text{estGele}(G, \text{alex}(G)) \\ 0 & \text{Sinon} \end{cases}$$

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

$$\left\{ \begin{array}{l} \text{- Personnage : :jeter(alex(G))} \\ \text{si } cA = \text{JETER} \wedge \neg \text{Personnage : :est_vaincu(alex(G))} \wedge \neg \text{estGele(G,alex(G))} \wedge \text{Bloc : :type(Terrain : :bloc(terrain(G),posX} \\ \text{- Personnage : :ramasser_objet(alex(G), Bloc : :objet(Terrain : :bloc(terrain(G), posX(alex(G)),posY(alex(G))))} \\ \text{si } cA = \text{RAMASSER} \wedge \text{posZ(alex(G))}=0 \wedge \neg \text{Personnage : :est_vaincu(alex(G))} \wedge \neg \text{estGele(G,alex(G))} \\ \text{- Personnage : :ramasser_perso(alex(G), p)} \\ \text{si } \text{collision(G,alex(G), p)} \wedge cA = \text{RAMASSER} \wedge \neg \text{Personnage : :est_vaincu(alex(G))} \wedge \neg \text{estGele(G,alex(G))} \\ \text{- Personnage : :ramasser_argent(alex(G), Bloc : :objet(Terrain : :bloc(terrain(G), posX(alex(G)),posY(alex(G))))} \\ \text{si } cA = \text{RAMASSER} \wedge \text{posZ(alex(G))}=0 \wedge \neg \text{Personnage : :est_vaincu(alex(G))} \wedge \neg \text{estGele(G,alex(G))} \\ \text{- Personnage : :retrait_vie(alex(G), Personnage : :force(p))} \\ \text{si } \text{collision(G,alex(G),p)} \wedge \text{actionGangster(G,p)} = \text{FRAPPER} \wedge \neg \text{Personnage : :est_vaincu(alex(G))} \wedge \neg \text{estGele(G,alex(G))} \\ \text{- Personnage : :retrait_vie(alex(G), Personnage : :points_de_vie(alex(G))} \\ \text{si } \text{Bloc : :type(Terrain : :bloc(terrain(G), posX(alex(G)), posY(alex(G)),posZ(alex(G))))} = \text{FOSSE} \wedge \neg \text{Personnage : :est_va} \\ \text{- alex(G) Sinon} \end{array} \right.$$

ryan(gerer(G,cA,cR)) =

$$\left\{ \begin{array}{l} \text{- Personnage : :jeter(ryan(G))} \\ \text{si } cR = \text{JETER} \wedge \neg \text{Personnage : :est_vaincu(ryan(G))} \wedge \neg \text{estGele(G,ryan(G))} \wedge \text{Bloc : :type(Terrain : :bloc(terrain(G),posX} \\ \text{- Personnage : :ramasser_objet(ryan(G), Bloc : :objet(Terrain : :bloc(terrain(G), posX(ryan(G)),posY(ryan(G))))} \\ \text{si } cR = \text{RAMASSER} \wedge \text{posZ(ryan(G))}=0 \wedge \neg \text{Personnage : :est_vaincu(ryan(G))} \wedge \neg \text{estGele(G,ryan(G))} \\ \text{- Personnage : :ramasser_perso(ryan(G), p)} \\ \text{si } \text{collision(G,ryan(G), p)} \wedge cR = \text{RAMASSER} \wedge \neg \text{Personnage : :est_vaincu(ryan(G))} \wedge \neg \text{estGele(G,ryan(G))} \\ \text{- Personnage : :ramasser_argent(ryan(G), Bloc : :objet(Terrain : :bloc(terrain(G), posX(ryan(G)),posY(ryan(G))))} \\ \text{si } cR = \text{RAMASSER} \wedge \text{posZ(ryan(G))}=0 \wedge \neg \text{Personnage : :est_vaincu(ryan(G))} \wedge \neg \text{estGele(G,ryan(G))} \\ \text{- Personnage : :retrait_vie(ryan(G), Personnage : :force(p))} \\ \text{si } \text{collision(G,ryan(G),p)} \wedge \text{actionGangster(G,p)} = \text{FRAPPER} \wedge \neg \text{Personnage : :est_vaincu(ryan(G))} \wedge \neg \text{estGele(G,ryan(G))} \\ \text{- Personnage : :retrait_vie(ryan(G), Personnage : :points_de_vie(ryan(G))} \\ \text{si } \text{Bloc : :type(Terrain : :bloc(terrain(G), posX(ryan(G)), posY(ryan(G)),posZ(ryan(G))))} = \text{FOSSE} \wedge \neg \text{Personnage : :est_v} \\ \text{- ryan(G) Sinon} \end{array} \right.$$

gangsters(gerer(G,cA,cR)) = { g =

$$\left\{ \begin{array}{l} \text{- Gangster : :jeter(g)} \\ \text{si } \text{actionGangster(G,g)} = \text{JETER} \wedge \neg \text{Personnage : :est_vaincu(g)} \wedge \neg \text{estGele(G,g)} \wedge \text{Bloc : :type(Terrain : :bloc(terrain(G),} \\ \text{- Gangster : :ramasser_objet(g, Bloc : :objet(Terrain : :bloc(terrain(G), posX(g),posY(g))))} \\ \text{si } \text{actionGangster(G,g)} = \text{RAMASSER} \wedge \text{posZ(g)}=0 \wedge \neg \text{Gangster : :est_vaincu(g)} \wedge \neg \text{estGele(G,g)} \\ \text{- Gangster : :retrait_vie(g, Personnage : :force(p))} \\ \text{si } \text{collision(G,alex(G),g)} \wedge cA = \text{FRAPPER} \wedge \neg \text{Gangster : :est_vaincu(g)} \wedge \neg \text{estGele(G,g)} \\ \text{- Gangster : :retrait_vie(g, Personnage : :force(p))} \\ \text{si } \text{collision(G,ryan(G),g)} \wedge cR = \text{FRAPPER} \wedge \neg \text{Gangster : :est_vaincu(g)} \wedge \neg \text{estGele(G,g)} \\ \text{- Gangster : :retrait_vie(g, Gangster : :points_de_vie(g)} \\ \text{si } \text{Bloc : :type(Terrain : :bloc(terrain(G), posX(g), posY(g),posZ(g)))} = \text{FOSSE} \wedge \neg \text{Gangster : :est_vaincu(g)} \wedge \neg \text{estGele(G} \\ \text{- g Sinon} \end{array} \right. \} \quad \forall g \in \text{gangsters(G)}$$

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

$$\left\{ \begin{array}{ll} \text{true} & \text{si } \neg \text{Personnage : :est_vaincu(alex(G))} \\ \text{false} & \text{sinon} \end{array} \right.$$

estVisible(gerer(G,cA,cR), g) =

$$\left\{ \begin{array}{ll} \text{true} & \text{si } \neg \text{Gangster : :est_vaincu(g)} \\ \text{true} & \text{si } \text{Personnage : :perso_equipe(alex(G))} = g \wedge cA = \text{JETER} \\ \text{false} & \text{si } \text{collision(G,alex(G),g)} \wedge cA = \text{RAMASSER} \\ \text{false} & \text{sinon} \end{array} \right\} \quad \forall g \in \text{gangsters(G)}$$

posX(gerer(G,cA,cR), p) =

```

posX(G, alex(G)) si cA = JETER  ∧ Personnage::perso_equipe(alex(G)) = p ∧
    ¬Personnage::est_vaincu(alex(G))
posX(G,p) sinon

```

```

posY(gerer(G,cA,cR),p) =
    posY(G, alex(G)) si cA = JETER ∧ Personnage::perso_equipe(alex(G)) = p ∧
        ¬Personnage::est_vaincu(alex(G))
posY(G,p) sinon

```

```

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

```

TODO SLICK AND GANGSTERS AND RYAN

```

Terrain::bloc(terrain(gerer(G,cA,cR)), posX(alex(G)), posY(alex(G))) =
- Bloc::retirerObjet(Terrain::bloc(terrain(G), posX(alex(G)), posY(alex(G))))
    si cA = RAMASSER ∧ ¬Personnage::est_vaincu(alex(G))
- Bloc::poserObjet(Terrain::bloc(terrain(G), posX(alex(G)), posY(alex(G))),
    Personnage:objet_equipe(alex(G))
    si cA = JETER ∧ Personnage::est_equipe_objet(alex(G)) = true ∧ ¬Personnage::
        est_vaincu(alex(G))
- Bloc::poserObjet(Terrain::bloc(terrain(G), posX(g),posY(g)), Objet::init("
    Recompense",0,1000)) si Gangster::est_vaincu(g)
- Terrain::bloc(terrain(G), posX(alex(G)), posY(alex(G))) Sinon

```