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

Observators :
   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) =min= points_de_vie(P) ≤ 0
    est_equipe_perso(P) =min= perso_equipe(P) ≠ null
    est_equipe_objet(P) =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_objet(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
Observators :
   const nom : [Object] → 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 → [Object]
      pre(init(n,t,bonus,valeur) require n≠"" ∧ ( ( bonus >0 ∧ valeur = 0) ∨ (bonus = 0 ∧
          valeur> 0) )

Observations :
    [Invariants]
      est_equipable(O) =^min bonus_force > 0
      est_de_valeur(O) =^min valeur_marchande > 0
      est_equipable(O) =^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
Observators :
   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}
Observators :
      estFini : [MoteurJeu] → boolean
      resultat: [MoteurJeu] → RESULTAT
       pre resultat(M) require estFini(M)
      combat : [MoteurJeu] → GestionCombat
      pasCourant : [MoteurJeu] → int
Constructors :
      init : ∅ → [MoteurJeu]
Operators :
      pasJeu : [MoteurJeu] × COMMANDE × COMMANDE → [MoteurJeu]
       pre pasJeu(M, comAlex, comRyan) require : ¬estFini(M)
Observations :
   [Invariants]

estFini(M) =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 ¬Personnage : :est_vaincu(GestionCombat : :alex(combat(M))) ∧ Gangster : :est_vaincu(GestionCombat : :slick(combat(M))) ∧ Personnage : :est_vaincu(GestionCombat : :ryan(combat(M)))
RYANGAGNANT	si ¬Personnage : :est_vaincu(GestionCombat : :ryan(combat(M))) ∧ Gangster : :est_vaincu(GestionCombat : :slick(combat(M))) ∧ Personnage : :est_vaincu(GestionCombat : :alex(combat(M)))
DEUXGAGNANTS	si ¬Personnage : :est_vaincu(GestionCombat : :ryan(combat(M))) ∧ Gangster : :est_vaincu(GestionCombat : :slick(combat(M))) ∧ ¬Personnage : :est_vaincu(GestionCombat : :alex(combat(M)))
SLICKGAGNANT	si Personnage :est_vaincu(GestionCombat : :ryan(combat(M))) ∧ ¬Gangster : :est_vaincu(GestionCombat : :slick(combat(M))) ∧ 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)
pasCourant(pasJeu(M, cA, cR)) = pasCourant(M)+1
```

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}

Observators :

terrain : [GestionCombat] → Terrain

alex : [GestionCombat] → Personnage

ryan : [GestionCombat] → Personnage

slick : [GestionCombat} → Gangster

gangsters : [GestionCombat] → List<Gangster>

actionGangster : [GestionCombat] × Gangster → COMMANDE
pre actionGangster(G, gang) require ¬Gangster::est_vaincu(gang)

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) =^min( -d ≤ posX(G,p1) - posX(G,p2) ≤ d+1) ∧ ( d = Personnage::largeur(p1)/2 + d = Personnage::
        largeur(p2)/2 )

    collisionGauche(G,p1,p2) =^min( -d ≤ posX(G,p2) - posX(G,p1) ≤ d+1) ∧ ( d = Personnage::largeur(p1)/2 + d = Personnage::
        largeur(p2)/2 )

    collisionDevant(G,p1,p2) =^min( -d ≤ posY(G,p1) - posY(G,p2) ≤ d+1) ∧ ( d = Personnage::profondeur(p1)/2 + d = Personnage
        ::profondeur(p2)/2 )
```

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

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

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

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

$$actionGangster(G,g) = RIEN \text{ si } estGele(G,g) \vee est\_vaincu(G,g) \ \forall \ g \in gangsters(G)$$

[init]

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

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

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

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

$$gangsters(init()) = \{g = Personnage::init("noname",20,20,20,10,50)\}, \ \forall \ g \in gangsters(G)$$

$$actionGangster(G,g) = RIEN \ \forall \ g \in gangsters(G)$$

$$estGele(init(), s) = false$$

$$collisionGauche(init(),p1,p2) = false$$

$$collisionDroite(init(),p1,p2) = false$$

$$collisionDevant(init(),p1,p2) = false$$

$$collisionDerriere(init(),p1,p2) = false$$

$$collisionDessous(init(),p1,p2) = false$$

$$collisionDessus(init(),p1,p2) = false$$

$$collision(init(),p1,p2) = false$$

$$estFrappe(init(), s) = false$$

$$posX(init(), alex(G)) < 50$$

$$posX(init(), slick(G)) > Terrain::largeur(terrain(G)) - 50$$

$$posX(init(), ryan(G)) < 50$$

$$posZ(init(), p) = 0$$

$$posY(init(), perso) = random$$

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

$$Bloc::type(Terrain:bloc(terrain(G), posX(init(), slick(G)), posY(init(), slick(G)), posZ(init(), slick(G)))) = VIDE$$

$$Bloc::type(Terrain:bloc(terrain(G), posX(init(), alex(G)), posY(init(), alex(G)), posZ(init(), alex(G)))) \neq FOSSE$$

$$Bloc::type(Terrain:bloc(terrain(G), posX(init(), ryan(G)), posY(init(), ryan(G)), posZ(init(), ryan(G)))) \neq FOSSE$$

[gerer]

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

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

% VIRER LES CONDITIONS DE COLLISION (affreuses)

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\text{posZ}(\text{gerer}(G, cA, cR), \text{gangsters}(G)) = \{ \ g = \begin{cases} 100 \text{ si actionGangster(G,g) = SAUT} \vee \text{actionGangster(G,g) = SAUTBAS} \vee \text{actionGangster(G,g) = SAUTHAUT} \vee \text{actionGangster(G,g) = SAUTDROITE} \\ \vee \text{actionGangster(G,g) = SAUTGAUCHE} \wedge \neg \text{Personnage : :est\_vaincu(g)} \wedge \neg \text{estGele(G,g)} \wedge \neg \text{collisionDessus(g,p)} \ \forall \ p \in \text{gangster} \vee p = g \\ \text{pos(G,g) si estGele(G,g)} \\ 0 \text{ Sinon} \end{cases} \} \ \forall \ g \in \text{gangsters}(G)$$

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

{ - Personnage : :jeter(alex(G))
si cA = JETER ∧ ¬Personnage : :est_vaincu(alex(G)) ∧ ¬estGele(G,alex(G)) ∧ Bloc : :type(Terrain : :bloc(terrain(G),posX(G,alex(G)),posY(G,alex(G))))=VIDE
- Personnage : :ramasser_objet(alex(G), Bloc : :objet(Terrain : :bloc(terrain(G), posX(G,alex(G)),posY(G,alex(G)))))
si cA = RAMASSER ∧ posZ(G,alex(G))=0 ∧ ¬Personnage : :est_vaincu(alex(G)) ∧ ¬estGele(G,alex(G))
- Personnage : :ramasser_perso(alex(G), p)
si collision(G,alex(G), p) ∧ cA = RAMASSER ∧ ¬Personnage : :est_vaincu(alex(G)) ∧ ¬estGele(G,alex(G))
- Personnage : :ramasser_argent(alex(G), Bloc : :objet(Terrain : :bloc(terrain(G), posX(G,alex(G)),posY(G,alex(G)))))
si cA = RAMASSER ∧ posZ(G,alex(G))=0 ∧ ¬Personnage : :est_vaincu(alex(G)) ∧ ¬estGele(G,alex(G))
- Personnage : :retrait_vie(alex(G), Personnage : :force(p))
si collision(G,alex(G),p) ∧ actionGangster(G,p) = FRAPPER ∧ ¬Personnage : :est_vaincu(alex(G)) ∧ ¬estGele(G,alex(G))
- Personnage : :retrait_vie(alex(G), Personnage : :points_de_vie(alex(G))
si Bloc : :type(Terrain : :bloc(terrain(G), posX(G,alex(G)), posY(G,alex(G)))) = FOSSE ∧ ¬Personnage : :est_vaincu(alex(G)) ∧ ¬estGele(G,alex(G))
- alex(G) Sinon

ryan(gerer(G,cA,cR)) =
{ - Personnage : :jeter(ryan(G))
si cR = JETER ∧ ¬Personnage : :est_vaincu(ryan(G)) ∧ ¬estGele(G,ryan(G)) ∧ Bloc : :type(Terrain : :bloc(terrain(G),posX(G,ryan(G)),posY(G,ryan(G))))=VIDE
- Personnage : :ramasser_objet(ryan(G), Bloc : :objet(Terrain : :bloc(terrain(G), posX(G,ryan(G)),posY(G,ryan(G)))))
si cR = RAMASSER ∧ posZ(G,ryan(G))=0 ∧ ¬Personnage : :est_vaincu(ryan(G)) ∧ ¬estGele(G,ryan(G))
- Personnage : :ramasser_perso(ryan(G), p)
si collision(G,ryan(G), p) ∧ cR = RAMASSER ∧ ¬Personnage : :est_vaincu(ryan(G)) ∧ ¬estGele(G,ryan(G))
- Personnage : :ramasser_argent(ryan(G), Bloc : :objet(Terrain : :bloc(terrain(G), posX(G,ryan(G)),posY(G,ryan(G)))))
si cR = RAMASSER ∧ posZ(G,ryan(G))=0 ∧ ¬Personnage : :est_vaincu(ryan(G)) ∧ ¬estGele(G,ryan(G))
- Personnage : :retrait_vie(ryan(G), Personnage : :force(p))
si collision(G,ryan(G),p) ∧ actionGangster(G,p) = FRAPPER ∧ ¬Personnage : :est_vaincu(ryan(G)) ∧ ¬estGele(G,ryan(G))
- Personnage : :retrait_vie(ryan(G), Personnage : :points_de_vie(ryan(G))
si Bloc : :type(Terrain : :bloc(terrain(G), posX(G,ryan(G)), posY(G,ryan(G)))) = FOSSE ∧ ¬Personnage : :est_vaincu(ryan(G)) ∧ ¬estGele(G,ryan(G))
- ryan(G) Sinon

gangsters(gerer(G,cA,cR)) = { g =
{ - Gangster : :jeter(g)
si actionGangster(G,g) = JETER ∧ ¬Personnage : :est_vaincu(g) ∧ ¬estGele(G,g) ∧ Bloc : :type(Terrain : :bloc(terrain(G),posX(G,g),posY(G,g)))=VIDE
- Gangster : :ramasser_objet(g, Bloc : :objet(Terrain : :bloc(terrain(G), posX(G,g),posY(G,g))))
si actionGangster(G,g) = RAMASSER ∧ posZ(G,g)=0 ∧ ¬Gangster : :est_vaincu(g) ∧ ¬estGele(G,g)
- Gangster : :retrait_vie(g, Personnage : :force(p))
si collision(G,alex(G),g) ∧ cA = FRAPPER ∧ ¬Gangster : :est_vaincu(g) ∧ ¬estGele(G,g)
- Gangster : :retrait_vie(g, Personnage : :force(p))
si collision(G,ryan(G),g) ∧ cR = FRAPPER ∧ ¬Gangster : :est_vaincu(g) ∧ ¬estGele(G,g)
- Gangster : :retrait_vie(g, Gangster : :points_de_vie(g)
si Bloc : :type(Terrain : :bloc(terrain(G), posX(G,g), posY(G,g))) = FOSSE ∧ ¬Gangster : :est_vaincu(g) ∧ ¬estGele(G,g)
- g Sinon
} ∀ g ∈ gangsters(G)

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

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

$$\mathrm{estVisible}(\mathrm{gerer}(G,\mathrm{cA},\mathrm{cR}),\mathrm{alex}(G)) = \begin{cases} \text{true} & \text{si } \neg\text{Personnage : :est\_vaincu(alex(G))} \\ \text{false} & \text{sinon} \end{cases}$$

$$\mathrm{estVisible}(\mathrm{gerer}(G,\mathrm{cA},\mathrm{cR}),\mathrm{ryan}(G)) = \begin{cases} \text{true} & \text{si } \neg\text{Personnage : :est\_vaincu(ryan(G))} \\ \text{false} & \text{sinon} \end{cases}$$

$$\mathrm{estVisible}(\mathrm{gerer}(G,\mathrm{cA},\mathrm{cR}),g) = \left\{ \begin{array}{ll} \text{true} & \text{si } \neg\text{Gangster : :est\_vaincu(g)} \\ \text{true} & \text{si Personnage :perso\_equipe(alex(G)) = g} \wedge \text{cA = JETER} \\ \text{false} & \text{si collision(G,alex(G),g)} \wedge \text{cA = RAMASSER} \\ \text{false} & \text{sinon} \end{array} \right\} \quad \forall\ g \in \mathrm{gangsters}(G)$$

$$\mathrm{estVisible}(\mathrm{gerer}(G,\mathrm{cA},\mathrm{cR}),\mathrm{slick}(G)) = \begin{cases} \text{true} & \text{si } \neg\text{Gangster : :est\_vaincu(slick(G))} \\ \text{true} & \text{si Personnage :perso\_equipe(alex(G)) = slick(G)} \wedge \text{cA = JETER} \\ \text{false} & \text{si collision(G,alex(G),slick(G))} \wedge \text{cA = RAMASSER} \\ \text{false} & \text{sinon} \end{cases}$$

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

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

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

$$\mathrm{posX}(G, \mathrm{gerer}(G, cA, cR), p) =$$
$$\begin{cases} \mathrm{posX}(G,\mathrm{ryan}(G)) & \text{si } cA = \mathrm{JETER} \wedge \mathrm{Personnage::perso\_equipe}(\mathrm{ryan}(G)) = p \wedge \neg\, \mathrm{Personnage::est\_vaincu}(\mathrm{ryan}(G)) \\ \mathrm{posX}(G,p) & \text{sinon} \end{cases}$$

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

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

$$\mathrm{Terrain::bloc}(\mathrm{terrain}(\mathrm{gerer}(G, cA, cR)),\ \mathrm{posX}(G, \mathrm{alex}(G)),\ \mathrm{posY}(G, \mathrm{alex}(G))) =$$
$$\begin{cases} \text{- Bloc : :retirerObjet(Terrain : :bloc(terrain(G), posX(G,alex(G)), posY(G,alex(G))))} \\ \text{si } cA = \mathrm{RAMASSER} \wedge \neg\, \mathrm{Personnage::est\_vaincu}(\mathrm{alex}(G)) \wedge \neg \mathrm{estGele}(G,\mathrm{alex}(G)) \\ \text{- Bloc : :poserObjet(Terrain : :bloc(terrain(G), posX(G,alex(G)), posY(G,alex(G)))), Personnage :objet\_equipe(alex(G))} \\ \text{si } cA = \mathrm{JETER} \wedge \mathrm{Personnage::est\_equipe\_objet}(\mathrm{alex}(G)) = \mathrm{true} \wedge \neg\, \mathrm{Personnage::est\_vaincu}(\mathrm{alex}(G)) \wedge \neg \mathrm{estGele}(G,\mathrm{alex}(G)) \\ \text{- Terrain : :bloc(terrain(G), posX(G,alex(G)), posY(G,alex(G))) Sinon} \end{cases}$$

$$\mathrm{Terrain::bloc}(\mathrm{terrain}(\mathrm{gerer}(G, cA, cR)),\ \mathrm{posX}(G, \mathrm{ryan}(G)),\ \mathrm{posY}(G, \mathrm{ryan}(G))) =$$
$$\begin{cases} \text{- Bloc : :retirerObjet(Terrain : :bloc(terrain(G), posX(G,ryan(G)), posY(G,ryan(G))))} \\ \text{si } cR = \mathrm{RAMASSER} \wedge \neg\, \mathrm{Personnage::est\_vaincu}(\mathrm{ryan}(G)) \wedge \neg \mathrm{estGele}(G,\mathrm{ryan}(G)) \\ \text{- Bloc : :poserObjet(Terrain : :bloc(terrain(G), posX(G,ryan(G)), posY(G,ryan(G)))), Personnage :objet\_equipe(ryan(G))} \\ \text{si } cR = \mathrm{JETER} \wedge \mathrm{Personnage::est\_equipe\_objet}(\mathrm{ryan}(G)) = \mathrm{true} \wedge \neg\, \mathrm{Personnage::est\_vaincu}(\mathrm{ryan}(G)) \wedge \neg \mathrm{estGele}(G,\mathrm{ryan}(G)) \\ \text{- Terrain : :bloc(terrain(G), posX(G,ryan(G)), posY(G,ryan(G))) Sinon} \end{cases}$$

$$\mathrm{Terrain::bloc}(\mathrm{terrain}(\mathrm{gerer}(G, cA, cR)),\ \mathrm{posX}(G, \mathrm{slickG})),\ \mathrm{posY}(G, \mathrm{slick}(G))) =$$
$$\begin{cases} \text{- Bloc : :retirerObjet(Terrain : :bloc(terrain(G), posX(G,slick(G)), posY(G,slick(G))))} \\ \text{si actionGangster(G,slick(G)) = RAMASSER} \wedge \neg\, \mathrm{Personnage::est\_vaincu}(\mathrm{slick}(G)) \wedge \neg \mathrm{estGele}(G,\mathrm{slick}(G)) \\ \text{- Bloc : :poserObjet(Terrain : :bloc(terrain(G), posX(G,slick(G)), posY(G,slick(G)))), Personnage :objet\_equipe(slick(G))} \\ \text{si actionGangster(G,slick(G)) = JETER} \wedge \mathrm{Personnage::est\_equipe\_objet}(\mathrm{slick}(G)) = \mathrm{true} \wedge \neg\, \mathrm{Personnage::est\_vaincu}(\mathrm{slick}(G)) \wedge \neg \mathrm{estGele}(G,\mathrm{slick}(G)) \\ \text{- Terrain : :bloc(terrain(G), posX(G,slick(G)), posY(G,slick(G))) Sinon} \end{cases}$$

$$\mathrm{Terrain::bloc}(\mathrm{terrain}(\mathrm{gerer}(G, cA, cR)),\ \mathrm{posX}(G, g),\ \mathrm{posY}(G, g)) =$$
$$\begin{cases} \text{- Bloc : :retirerObjet(Terrain : :bloc(terrain(G), posX(G,g), posY(G,g)))} \\ \text{si actionGangster(G,g) = RAMASSER} \wedge \neg\, \mathrm{Personnage::est\_vaincu}(g) \wedge \neg \mathrm{estGele}(G,g) \\ \text{- Bloc : :poserObjet(Terrain : :bloc(terrain(G), posX(G,g), posY(G,g))), Personnage :objet\_equipe(g)} \\ \text{si actionGangster(G,g) = JETER} \wedge \mathrm{Personnage::est\_equipe\_objet}(g) = \mathrm{true} \wedge \neg\, \mathrm{Personnage::est\_vaincu}(g) \wedge \neg \mathrm{estGele}(G,g) \quad \forall\ g \in \mathrm{gangsters}(G) \\ \text{- Bloc : :poserObjet(Terrain : :bloc(terrain(G), posX(G,g),posY(G,g)), Objet : :init("Recompense",0,1000))} \\ \text{si Gangster : :est\_vaincu(g)} \\ \text{- Terrain : :bloc(terrain(G), posX(G,g), posY(G,g)) Sinon} \end{cases}$$