Chapitre 1

Projet CPS : Spécifications de River City Ransom

Béatrice CARRE et Steven VAROUMAS
Lien vers l'énoncé du projet : lien.

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

   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) ∧ Objet::est_equipable(o)) ∨ Objet::est_de_valeur(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**]

$$\text{est\_vaincu}(P) \overset{min}{=} \text{points\_de\_vie}(P) \leq 0$$
$$\text{est\_equipe\_perso}(P) \overset{min}{=} \text{perso\_equipe}(P) \neq \text{null}$$
$$\text{est\_equipe\_objet}(P) \overset{min}{=} \text{objet\_equipe}(P) \neq \text{null}$$

[**init**]

$$\text{nom}(\text{init}(n,l,h,p,f,v,a)) = n$$
$$\text{largeur}(\text{init}(n,l,h,p,f,v,a)) = l$$
$$\text{hauteur}(\text{init}(n,l,h,p,f,v,a)) = h$$
$$\text{profondeur}(\text{init}(n,l,h,p,f,v,a)) = p$$
$$\text{force}(\text{init}(n,l,h,p,f,v,a)) = f$$
$$\text{points\_de\_vie}(\text{init}(n,l,h,p,f,v,a)) = v$$
$$\text{somme\_d\_argent}(\text{init}(n,l,h,p,f,v,a)) = a$$
$$\text{objet\_equipe}(\text{init}(n,l,h,p,f,v,a)) = \text{null}$$
$$\text{perso\_equipe}(\text{init}(n,l,h,p,f,v,a)) = \text{null}$$

[**retrait_vie**]

$$\text{points\_de\_vie}(\text{retrait\_vie}(P,s)) = \text{points\_de\_vie}(P) - s$$

[**depot_vie**]

$$\text{points\_de\_vie}(\text{depot\_vie}(P,s)) = \text{points\_de\_vie}(P) + s$$

[**retrait_argent**]

$$\text{somme\_d\_argent}(\text{retrait\_argent}(P,s)) = \text{argent}(P) - s$$

[**depot_argent**]

$$\text{somme\_d\_argent}(\text{depot\_argent}(P,s)) = \text{argent}(P) + s$$

[**ramasser_objet**]

$$\text{objet\_equipe}(\text{ramasser\_objet}(P,\text{objet})) = \begin{cases} \text{objet} & \text{si Objet::est\_equipable(objet\_equipe(P))} \\ \text{null} & \text{sinon} \end{cases}$$

force(ramasser_objet(P, objet)) =
$$\begin{cases} \text{force(P)} + \text{Objet : :bonus\_force(objet)} & \text{si Objet : :est\_equipable(objet)} \\ \text{force(P)} & \text{sinon} \end{cases}$$

somme_d_argent(ramasser_objet(P, objet)) =
$$\begin{cases} \text{somme\_d\_argent(P)} + \text{Objet : :valeur\_marchande(objet)} & \text{si Objet : :est\_de\_valeur(objet)} \\ \text{somme\_d\_argent(P)} & \text{sinon} \end{cases}$$

[**ramasser_perso**]
perso_equipe(ramasser_perso(P, perso)) = perso

[**jeter**]
perso_equipe(jeter(P)) = null
force(jeter(P)) =
$$\begin{cases} \text{force(P) - Objet : :bonus\_force(objet\_equipe(P))} \\ \text{si est\_equipe\_objet(P)} \wedge \text{Objet : :est\_equipable(objet\_equipe(P))} \\ \text{force(P) sinon} \end{cases}$$
objet_equipe(jeter(P)) = null

1.2 Gangster

service: Gangster
Refine : Personnage
use : enum ACTION{RIEN,FRAPPE,SAUTE,HAUT,BAS,GAUCHE,DROITE}
Observators :
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},
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=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
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 > 0 ∧ hauteur > 0 ∧ prof > 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
Constructors :
      init : ∅ → [MoteurJeu]
Operators :
      pasJeu : [MoteurJeu] × COMMANDE × COMMANDE → [MoteurJeu]
            pre pasJeu(M, comAlex, comRyan) require : ¬estFini(M)
Observations :
   [Invariants]
```

$$\text{estFini(M)} \overset{min}{=} \begin{cases} (\text{Personnage : : estVaincu(GestionCombat : :alex(combat(M)))} \\ \wedge\ \text{Personnage : :estVaincu(GestionCombat : :ryan(combat(M))))} \\ \vee\ \text{Gangster : :estVaincu(GestionCombat : :slick(combat(M))} \end{cases}$$

$$\text{resultat(M)} \overset{min}{=} \begin{cases} \text{ALEXGAGNANT} & \text{si Personnage : :!estVaincu(GestionCombat : :alex(combat(M)))} \\ & \wedge\ \text{Gangster : :estVaincu(GestionCombat : :slick(combat(M)))} \\ & \wedge\ \text{Personnage : :estVaincu(GestionCombat : :ryan(combat(M)))} \\ \text{RYANGAGNANT} & \text{si Personnage : :!estVaincu(GestionCombat : :ryan(combat(M)))} \\ & \wedge\ \text{Gangster : :estVaincu(GestionCombat : :slick(combat(M)))} \\ & \wedge\ \text{Personnage : :estVaincu(GestionCombat : :alex(combat(M)))} \\ \text{DEUXGAGNANTS} & \text{si Personnage : :!estVaincu(GestionCombat : :ryan(combat(M)))} \\ & \wedge\ \text{Gangster : :estVaincu(GestionCombat : :slick(combat(M)))} \\ & \wedge\ \text{Personnage : :!estVaincu(GestionCombat : :alex(combat(M)))} \\ \text{SLICKGAGNANT} & \text{si Personnage : estVaincu(GestionCombat : :ryan(combat(M)))} \\ & \wedge\ \text{Gangster : :!estVaincu(GestionCombat : :slick(combat(M)))} \\ & \wedge\ \text{Personnage : :estVaincu(GestionCombat : :alex(combat(M)))} \\ \text{NULLE} & \text{sinon} \end{cases}$$

[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}

Observators :
      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 × 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)
        collision(G,p1,p2) =min A FAIRE

    [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)}, ∀ g ∈
      gangsters(G)
  estGele(init(), s) = false
  collision(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 ∀ g ∈ 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))))≠FOSSE
  Bloc::type(Terrain:bloc(terrain(G),posX(init(),ryan(G)),posY(init(),ryan(G)),posZ(
      init(),ryan(G))))≠FOSSE


    [gerer]
  posX(gerer(G,cA,cR),alex(G)) =
      { posX(G,alex(G)) + 10   si cA = DROIT ∨ cA = SAUTDROIT ∧ ¬Personnage : :estVaincu(alex(G))
      { posX(G,alex(G)) - 10   si cA = GAUCHE ∨ cA = SAUTGAUCHE ∧ ¬Personnage : :estVaincu(alex(G))
      { posX(G,alex(G))        sinon

  posY(gerer(G,cA,cR),alex(G)) =
      { posY(G,alex(G)) + 10   si cA = HAUT ∨ cA = SAUTHAUT ∧ ¬Personnage : :estVaincu(alex(G))
      { posY(G,alex(G)) - 10   si cA = BAS ∨ cA = SAUTBAS ∧ ¬Personnage : :estVaincu(alex(G))
      { posY(G,alex(G))        sinon
```

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

$$
\begin{array}{l}
\mathrm{posX}(\mathrm{gerer}(G,\mathrm{cA},\mathrm{cR}),\mathrm{ryan}(G)) = \\
\quad \left\{ \begin{array}{ll}
\mathrm{posX}(G,\mathrm{ryan}(G)) + 10 & \text{si } \mathrm{cR} = \mathrm{DROIT} \vee \mathrm{cR} = \mathrm{SAUTDROIT} \wedge \neg \mathrm{Personnage} : : \mathrm{estVaincu}(\mathrm{ryan}(G)) \\
\mathrm{posX}(G,\mathrm{ryan}(G)) - 10 & \text{si } \mathrm{cR} = \mathrm{GAUCHE} \vee \mathrm{cR} = \mathrm{SAUTGAUCHE} \wedge \neg \mathrm{Personnage} : : \mathrm{estVaincu}(\mathrm{ryan}(G)) \\
\mathrm{posX}(G,\mathrm{ryan}(G)) & \text{sinon}
\end{array} \right.
\end{array}
$$

$$
\begin{array}{l}
\mathrm{posY}(\mathrm{gerer}(G,\mathrm{cA},\mathrm{cR}),\mathrm{ryan}(G)) = \\
\quad \left\{ \begin{array}{ll}
\mathrm{posY}(G,\mathrm{ryan}(G)) + 10 & \text{si } \mathrm{cR} = \mathrm{HAUT} \vee \mathrm{cR} = \mathrm{SAUTHAUT} \wedge \neg \mathrm{Personnage} : : \mathrm{estVaincu}(\mathrm{ryan}(G)) \\
\mathrm{posY}(G,\mathrm{ryan}(G)) - 10 & \text{si } \mathrm{cR} = \mathrm{BAS} \vee \mathrm{cR} = \mathrm{SAUTBAS} \wedge \neg \mathrm{Personnage} : : \mathrm{estVaincu}(\mathrm{ryan}(G)) \\
\mathrm{posY}(G,\mathrm{ryan}(G)) & \text{sinon}
\end{array} \right.
\end{array}
$$

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

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

$$
\begin{array}{l}
\mathrm{posX}(\mathrm{gerer}(G,\mathrm{cA},\mathrm{cR}),\mathrm{p}) = \\
\quad \mathrm{posX}(G,\mathrm{alex}(G))+10 \text{ si } \mathrm{cA} = \mathrm{JETER} \wedge \mathrm{Personnage} : : \mathrm{perso\_equipe}(\mathrm{alex}()) = \mathrm{p} \wedge \\
\qquad \neg \mathrm{Personnage} : : \mathrm{estVaincu}(\mathrm{alex}(G)) \\
\quad \mathrm{posX}(G,\mathrm{p}) \text{ sinon} \\
\mathrm{posY}(\mathrm{gerer}(G,\mathrm{cA},\mathrm{cR}),\mathrm{p}) = \\
\quad \mathrm{posY}(G,\mathrm{alex}(G)) \text{ si } \mathrm{cA} = \mathrm{JETER} \wedge \mathrm{Personnage} : : \mathrm{perso\_equipe}(\mathrm{alex}()) = \mathrm{p} \wedge \\
\qquad \neg \mathrm{Personnage} : : \mathrm{estVaincu}(\mathrm{alex}(G)) \\
\quad \mathrm{posY}(G,\mathrm{p}) \text{ sinon} \\
\mathrm{posZ}(\mathrm{gerer}(G,\mathrm{cA},\mathrm{cR}),\mathrm{p}) = \\
\quad 0 \text{ si } \mathrm{cA} = \mathrm{JETER} \wedge \mathrm{Personnage} : : \mathrm{perso\_equipe}(\mathrm{alex}()) = \mathrm{p} \wedge \neg \mathrm{Personnage} : : \mathrm{estVaincu} \\
\qquad (\mathrm{alex}(G)) \\
\quad \mathrm{posZ}(G,\mathrm{p}) \text{ sinon}
\end{array}
$$

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

```
ryan(gerer(G,cA,cR)) =
{ - Personnage : :jeter(ryan(G)) si cR = JETER ∧ ¬Personnage : :estVaincu(ryan(G))
  - Personnage : :ramasser_objet(ryan(G), Bloc : :objet(Terrain : :bloc(terrain(G), posX(ryan(G)),
  posY(ryan(G)),posZ(ryan(G)))) si cR = RAMASSER ∧ ¬Personnage : :estVaincu(ryan(G))
  - Personnage : :ramasser_perso(ryan(G), p)
   si collision(ryan(G), p) ∧ cR = RAMASSER ∧ ¬Personnage : :estVaincu(ryan(G))
  - Personnage : :retrait_vie(ryan(G), Personnage : :force(p))
   si collision(ryan(G),p) ∧ Gangster : :action(p) = FRAPPER ∧ ¬Personnage : :estVaincu(ryan(G))
  - Personnage : :retrait_vie(ryan(G), Personnage : :points_de_vie(ryan(G))
   si Bloc : :type(Terrain : :bloc(terrain(G), posX(ryan(G)), posY(ryan(G)),posZ(ryan(G)))) = FOSSE
  ∧ ¬Personnage : :estVaincu(ryan(G))
  - ryan(G) Sinon

posX(gerer(G,cA,cR),p) =
    posX(ryan(G))+10 si cR = JETER ∧ Personnage::perso_equipe(ryan()) = p ∧
        ¬Personnage::estVaincu(ryan(G))
    posX(G,p) sinon

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

posZ(gerer(G,cA,cR),p) =
    0 si cR = JETER ∧ Personnage::perso_equipe(ryan()) = p ∧ ¬Personnage::estVaincu
        (ryan(G))
    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 ∧ ¬Personnage::estVaincu(alex(G))
- 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 ∧ ¬Personnage::
        estVaincu(alex(G))
- Bloc::retirerObjet(Terrain::bloc(terrain(G), posX(ryan(G)), posY(ryan(G)),posZ(
    ryan(G)))
    si cR = RAMASSER ∧ ¬Personnage::estVaincu(ryan(G))
- 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 ∧ ¬Personnage::
        estVaincu(ryan(G))
- terrain(G) sinon
```