

04/05/2017

Spécification du jeu Street Fighter



Loïc LAFONTAINE MAXIME LAVASTE

Spécification

Engine

Service: Engine

Types: bool, int, Commande

Observers:

```
const height: [Engine] → int
const width: [Engine] → int
char: [Engine] × int → Character
  pre char(E,i) requires i ≥ 1, 2g
player: [Engine] × int → Player
  pre player(E,i) requires i ≥ 1, 2g
gameOver: [Engine] → bool
```

Constructors:

```
init: int × int × int × Player × Player → [Engine]
  pre init(h,w,s,p1,p2) requires h > 0 && s > 0 && w > s && p1 !=
  p2
```

Operators:

```
step: [Engine] × Commande × Commande → [Engine]
  pre step(E) requires : gameOver(E)
```

Observations:

[invariant]:

```
gameOver(E) (∃i, Character(engine(C), i) = Character ::dead(player(E,
i)))
```

[init]:

```
height(init(h, w, s, p1, p2)) = h
width(init(h, w, s, p1, p2)) = w
space(init(h, w, s, p1, p2)) = s
player(init(h, w, s, p1, p2), 1) = p1
player(init(h, w, s, p1, p2), 2) = p2
Character ::positionX(char(init(h, w, s, p1, p2), 1)) = w//2 - s//2
Character ::positionX(char(init(h, w, s, p1, p2), 2)) = w//2 + s//2
Character ::positionY(char(init(h, w, s, p1, p2), 1)) = 0
Character ::positionY(char(init(h, w, s, p1, p2), 2)) = 0
Character ::faceRight(char(init(h, w, s, p1, p2), 1))
Character ::faceRight(char(init(h, w, s, p1, p2), 2))
```

[step]:

```
char(step(E, C1, C2), 1) = step(char(E, 1), C1)
char(step(E, C1, C2), 2) = step(char(E, 2), C2)
```

Hitbox

Service: Hitbox

Types: bool, int

Observers:

```
PositionX: [Hitbox] → int
PositionY: [Hitbox] → int
Hauteur: [Hitbox] → int
Longueur: [Hitbox] → int
BelongsTo: [Hitbox] × int × int → bool
CollidesWith: [Hitbox] × Hitbox → bool
EqualsTo: [Hitbox] × Hitbox → bool
```

Constructors:

init: $\text{int} \times \text{int} \times \text{int} \times \text{int} \rightarrow [\text{Hitbox}]$
 pre init(x, y, h, l) **requires** $h > 0 \ \&\& \ l > 0$

Operators:

MoveTo: $[\text{Hitbox}] \times \text{int} \times \text{int} \rightarrow [\text{Hitbox}]$
 pre moveTo(h, w) **requires** $h > 0 \ \&\& \ w > 0$
 resize: $[\text{Hitbox}] \times \text{int} \times \text{int} \rightarrow [\text{Hitbox}]$

Observations:**[invariant]:**

CollidesWith(H,H1) = 9 x,y:int \times int, BelongsTo(H,x,y) $\&\&$
 BelongsTo(H1,x,y)
 EqualsTo(H,H1) = 8 x,y:int \times int, BelongsTo(H,x,y) = BelongsTo(H1,x,y)

[init]:

PositionX(init(x, y, h, l)) = x
 PositionY(init(x, y, h, l)) = y
 Hauteur(init(x, y, h, l)) = h
 Largeur(init(x, y, h, l)) = l

[MoveTo]:

PositionX(MoveTo(H,x,y)) = x
 PositionY(MoveTo(H,x,y)) = y

[Resize]:

Hauteur (resize (H,h,w)) = h
 Largeur (resize (H,h,w)) = w

Combo**Service:** Combo**Types :** int**Observers:**

Combo : $[\text{Combo}] \rightarrow \text{int}$
 frameRestantes : $[\text{Combo}] \rightarrow \text{int}$

Constructors:

init: $\text{int} \times \text{int} \rightarrow [\text{Combo}]$

Operators:

reset: $[\text{Combo}] \rightarrow [\text{Combo}]$
 addCombo: $[\text{Combo}] \rightarrow [\text{Combo}]$
 pre addCombo() **requires** frameRestante() > 0
 step: $[\text{Combo}] \rightarrow \text{Boolean} \rightarrow [\text{Combo}]$

Observations:**[invariant]:**

Combo() ≥ 0

[init]:

Combo(init) = 0
 frameRestantes = 96

[Reset]:

Combo() = 0

[addCombo]:

combo(addCombo) = Combo + 1

[step]:**[removeFrame]:**

frameRestante(removeFrame()) = frameRestante - 1

[comboPossible]:

comboPossible == frameRestante > 0

Character

Service: Character

Types: bool, int, Commande, Personnage

Observers:

positionX: [Character] → int
positionY: [Character] → int
hauteur: [Character] → int
longueur: [Character] → int
personnage :[Character] → Personnage
nom :[Character] → String
state: [Character] → State
engine: [Character] → Engine
charBox: [Character] → Hitbox
jump: [Character] → Jump
life: [Character] → int
const speed: [Character] → int
faceRight: [Character] → bool
dead: [Character] → bool
id: [Character] → int

Constructors:

init: Personnage x int x int x bool x Engine → [Character]
pre init(p, l,s,f,e) **requires** l > 0 && s > 0

Operators:

moveLeft: [Character] → [Character]
moveRight: [Character] → [Character]
switchSide: [Character] → [Character]
step: [Character] x Commande → [Character]
pre step() **requires** !dead
bindJump: [Character] x Jump → [Character]

Observations:

[invariant]:

positionX(C) > 0 && positionX(C) < **Engine::** width(engine)
positionY(C) > 0 && positionY(C) < **Engine::** height(engine)
dead(C) = !(life > 0)

[init]:

life(init(p, l, s, f, e)) = l && speed(init(p, l, s, f, e)) = s &&
faceRight(init(p, l, s, f, e)) = f
&& engine(init(p, l, s, f, e)) = e && personage = p
∃h :Hitbox, charbox(init(p, l, s, f, e)) = h
∃j :Jump, jump(init(p, l, s, f, e)) = j

[moveX] : #Tous les mouvements

faceRight(moveLeft(C)) = faceRight(C) ∧ life(moveLeft(C)) = life(C)

[moveLeft]:

(∃i, player(engine(C), i) != C ∧ collisionwith(hitbox(moveLeft(C)),
hitbox(player(engine(C), i)))) ⇒ positionX(moveLeft(C)) = positionX(C)

$\text{positionX}(C) \leq \text{speed}(C) \wedge (\forall i, \text{player}(\text{engine}(C), i) \neq C \Rightarrow \neg \text{collisionWith}(\text{hitbox}(\text{moveLeft}(C)),$

$\text{hitbox}(\text{player}(\text{engine}(C), i)))) \Rightarrow \text{positionX}(\text{moveLeft}(C)) = \text{positionX}(C) - \text{speed}(C)$

$\text{positionX}(C) > \text{speed}(C) \wedge (\forall i, \text{player}(\text{engine}(C), i) \neq C \Rightarrow \neg \text{collisionWith}(\text{hitbox}(\text{moveLeft}(C)),$

$\text{hitbox}(\text{player}(\text{engine}(C), i)))) \Rightarrow \text{positionX}(\text{moveLeft}(C)) = 0$

$\text{positionY}(\text{moveLeft}(C)) = \text{positionY}(C)$

$\text{HitboxState}(\text{moveLeft}(C)) = \text{HitboxState}::\text{STANDING}$

[moveRight]:

$(\exists i, \text{player}(\text{engine}(C), i) \neq C \wedge \text{collisionWith}(\text{hitbox}(\text{moveRight}(C)),$

$\text{hitbox}(\text{player}(\text{engine}(C), i)))) \Rightarrow \text{positionX}(\text{moveRight}(C)) = \text{positionX}(C)$

$\text{positionX}(C) \leq \text{speed}(C) \wedge (\forall i, \text{player}(\text{engine}(C), i) \neq C \Rightarrow$

$\neg \text{collisionWith}(\text{hitbox}(\text{moveRight}(C)), \text{hitbox}(\text{player}(\text{engine}(C), i)))) \Rightarrow \text{positionX}(\text{moveRight}(C)) = \text{positionX}(C) + \text{speed}(C)$

$\text{positionX}(C) \leq \text{speed}(C) \wedge (\forall i, \text{player}(\text{engine}(C), i) \neq C \Rightarrow$

$\neg \text{collisionWith}(\text{hitbox}(\text{moveRight}(C)), \text{hitbox}(\text{player}(\text{engine}(C), i)))) \Rightarrow$

$\text{positionY}(\text{moveLeft}(C)) = \text{positionY}(C)$

$\text{HitboxState}(\text{moveLeft}(C)) = \text{HitboxState}::\text{STANDING}$

[moveDown]:

$\text{HitboxState}(\text{moveLeft}(C)) = \text{HitboxState}::\text{CROUCHING}$

[moveDownLeft]

$\text{HitboxState}(\text{moveLeft}(C)) = \text{HitboxState}::\text{CROUCHING}$

[moveDownRight]

$\text{HitboxState}(\text{moveLeft}(C)) = \text{HitboxState}::\text{CROUCHING}$

[moveUpRight]

[moveUpLeft]

[moveUpNeutral]

[moveDownRight]

POST BAISSMENT HAUTEUR HITBOX

[moveDownLeft]

[switchSide]:

$\text{faceRight}(\text{switchSide}(C)) \neq \text{faceRight}(C)$

$\text{positionX}(\text{switchSide}(C)) = \text{positionX}(C)$

[step]:

$\text{step}(C, \text{LEFT}) = \text{moveLeft}(C)$

$\text{step}(C, \text{RIGHT}) = \text{moveRight}(C)$

$\text{step}(C, \text{NEUTRAL}) = \text{neutral}(C)$

$\text{step}(C, \text{UPRIGHT}) = \text{moveUpRight}(C)$

$\text{step}(C, \text{UPLEFT}) = \text{moveUpLeft}(C)$

$\text{step}(C, \text{UPNEUTRAL}) = \text{moveUpNeutral}(C)$

$\text{step}(C, \text{DOWNRIGHT}) = \text{moveDownRight}(C)$

$\text{step}(C, \text{DOWNLEFT}) = \text{moveDownLeft}(C)$

$\text{step}(C, \text{DOWNNEUTRAL}) = \text{moveDownNeutral}(C)$

```

[bindHitbox]:
    Charbox(bindHitbox(h)) = h
[bindJump]:
    Jump(bindJump(j)) = j

```

FighterCharacter

Service: FighterCharacter refines Character

Observers:

```

isBlocking: [FightChar] → bool
isBlockstunned: [FightChar] → bool
isHitstunned: [FightChar] → bool
isTeching: [FightChar] → bool
techniqueCourante: [FightChar] → Tech
    pre tech(C) requires isTeching(C)
getComboService → Combo
getCombo → int
isCombo → bool
getCombo → int

```

Operators:

```

startTech(): [FightChar] x Technique → void
    pre startTech requires isTeching()
endTechnique(): [FightChar] → void
    pre startTech requires isTeching()
takeAttack(): [FightChar] x damage x hstun x bstun → void
    pre takeAttack requires damage > 0 && hstun > 0 && bstun > 0 &&
!dead
stepCombo(): [FightChar] → void

```

Observations:

```

[invariant]:
    isTeching() → techniqueCourante() != null
    isTeching → ¬isBlocking
    isHitStunned → (frameHitStun() > 0)
    isBlocking → ¬(isHitStunned || isBlockStun) = false
    isBlockStunned → (frameBlockStun() > 0)

```

Observers

[startTech]:

```

    TechniqueCourante(startTech()) != null && isTeching(startTech(t)) =
true

```

[stepCombo]:

```

    (h && Combo::isComboPossible(stepCombo(h))) =>
(getCombo(stepCombo(h))) = getCombo() + 1
    (h && ¬ Combo::isComboPossible(stepCombo(h))) =>
(getCombo(stepCombo(h))) = 1

```

[init]:

```

    ∃t :Technique, techniques(init(p, l, s, f, e)) = t
    ∃c :Technique, combo(init(p, l, s, f, e)) = c

```

[moveX] : #Tous les mouvements

```

(isTeaching() || isBlocking() || isHitStunned() || isBlockstunned()) =
false
combo(moveX(C)) = combo(c)

```

```

[takeAttack] :
    isBlocking(takeAttack()) = false
    isBlocking(C) → isBlockstunned(takeAttack(C))
    !isBlocking(C) → isHitstunned(takeAttack(C))
    isBlocking(C) → life(C) = life(takeAttack(C))
    !isBlocking(C) → life(C) = life(takeAttack(C)) + damage
    Combo(takeAttack()) = 0
    Rajouter hitstun pre = hstunpre + hstun + block?

```

Techniques

Service: Technique

Type : Hitbox

Observers:

```

damage: [Technique] → int
hitstun: [Technique] → int
blockstun: [Technique] → int
startuptime: [Technique] → int
hittime: [Technique] → int
recoverytime: [Technique] → int
box: [Technique] -> Hitbox
frame: [Technique] → int

```

Operators:

```

init : int x int x int x int x int x int
step : [Technique] -> [Technique]
    pre step(T) requires frame < startuptime + hittime + recoverytime
launchTechnique: [Technique] -> [Technique]

```

Observation:

```

[init] :
    damage(init(d,hs,bs,s,ht,r,bo)) = d
    hitstun(init(d,hs,bs,s,ht,r,bo)) = hs
    blockstun(init(d,hs,bs,s,ht,r,bo)) = bs
    startuptime(init(d,hs,bs,s,ht,r,bo)) = s
    hittime(init(d,hs,bs,s,ht,r,bo)) = ht
    recoverytime(init(d,hs,bs,s,ht,r,bo)) = r
    box(init(d,hs,bs,s,ht,r,bo)) = bo

```

```

[step]:
    frame(step(T)) = frame(T)+1

```

```

[launchTechnique] :
    frame = 0

```