

04/05/2017

Loïc LAFONTAINE MAXIME LAVASTE

Spécification du jeu Street Fighter



Table des matières

[Spécification 1](#_Toc481802135)

[Engine 1](#_Toc481802136)

[Hitbox 2](#_Toc481802137)

[Combo 2](#_Toc481802138)

[Character 3](#_Toc481802139)

[FighterCharacter 5](#_Toc481802140)

[Techniques 6](#_Toc481802141)

# Introduction

Dans le cadre de l’enseignement de l’UE CPS, nous avons dû spécifier, créer des test afin d’implémenter un Street Fighter via le pattern design decorator/ contract

# Spécification

## Engine

**Service**: Engine

**Types**: bool, int, Commande

**Observators**:

const height: [Engine] 🡪 int

const width: [Engine] 🡪 int

char: [Engine] × int 🡪 Character

**pre** char(E,i) **requires** i 2 f1, 2g

player: [Engine] × int 🡪Player

**pre** player(E,i) **requires** i 2 f1, 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 && Player::Character(0).isFaceRight() != Player::Character(1).isFaceRight()

**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))

Character(1).isFaceRight() != character(2).isFaceRight()

Character::PositionX(1) < Character::PositionX(2) ⇒ Character(1).isFaceRight() = true

Character::PositionX(1) > Character::PositionX(2) ⇒ Character(1).isFaceRight() = false

[**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)) = h - Character ::largeur(1)

Character ::positionY(char(init(h, w, s, p1, p2), 2)) = h - Character ::largeur(2)

Character ::faceRight(char(init(h, w, s, p1, p2), 1)) = true

Character :::faceRight(char(init(h, w, s, p1, p2), 2)) = false

[**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

**Observators**:

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: int × int x int x int 🡪 [Hitbox]

**pre** init(x, y, h, l) **requires** h > 0 && l > 0

**Operators**:

MoveTo: [Hitbox] × int × int 🡪 [Hitbox]

**pre** moveTo(h, w) **requires** h > 0 && w > 0

resize: [Hitbox] × int × int 🡪 [Hitbox]

**Observations**:

[**invariant**]:

PositionX > 0 && PositionY > 0 && largeur > 0 && hauteur > 0

[**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

**Observators**:

Combo : [Combo] 🡪 int

frameRestantes : [Combo] 🡪 int

**Constructors**:

init: int × int 🡪 [Combo]

**Operators:**

reset: [Combo] 🡪 [Combo]

addCombo: [Combo] 🡪 [Combo]

**pre** addCombo() **requires** frameRestante() > 0

step: [Combo] 🡪 Boolean 🡪 [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

**Observators**:

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 × int × bool × 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] × 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)

positionX(C) ≤ speed(C) ∧ (∀i, player(engine(C), i) != C ⇒ ¬collisionwith(hitbox(moveLeft(C)),

hitbox(player(engine(C), i)))) ⇒ positionX(moveLeft(C)) = positionX(C) − speed(C)

positionX(C) > speed(C) ∧(∀i, player(engine(C), i) != C ⇒ ¬collisionwith(hitbox(moveLeft(C)),

hitbox(player(engine(C), i)))) ⇒ positionX(moveLeft(C)) = 0

positionY(moveLeft(C)) = positionY(C)

HitboxState(moveLeft(C)) = HitboxState::STANDING

[**moveRight**]:

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

positionX(C) <= speed(C) ∧ (∀i, player(engine(C), i) != C ⇒

¬collisionwith(hitbox(moveRight(C)), hitbox(player(engine(C), i)))) ⇒ positionX(moveRight(C)) = positionX(C) + speed(C)

positionX(C) ≤ speed(C) ∧(∀i, player(engine(C), i) != C ⇒ ¬collisionwith(hitbox(moveRight(C)), hitbox(player(engine(C), i)))) ⇒

positionY(moveLeft(C)) = positionY(C)

HitboxState(moveLeft(C)) = HitboxState::STANDING

[**moveDown**]:

HitboxState(moveLeft(C)) = HitboxState::CROUCHING

[**moveDownLeft**]

HitboxState(moveLeft(C)) = HitboxState::CROUCHING

#Même postCondition que [**moveLeft**] sauf pour HitboxState

[**moveDownRight**]

HitboxState(moveLeft(C)) = HitboxState::CROUCHING

#Même postCondition que [**moveRight**] sauf pour HitboxState

[moveUp]

[**switchSide**]:

faceRight(switchSide(C)) = !faceRight(C)

positionX(switchSide(C)) = positionX(C)

[**step**]:

step(C, LEFT) = moveLeft(C)

step(C, RIGHT) = moveRight(C)

step(C, NEUTRAL) = neutral(C)

step(C, UPRIGHT) = moveUpRight(C)

step(C, UPLEFT) = moveUpLeft(C)

step(C, UPNEUTRAL) = moveUpNeutral(C)

step(C, DOWNRIGHT) = moveDownRight(C)

step(C, DOWNLEFT) = moveDownLeft(C)

step(C, DOWNNEUTRAL) = moveDownNeutral(C)

[**bindHitbox**]:

Charbox(bindHitbox(h)) = h

[**bindJump**]:

Jump(bindJump(j)) = j

## FighterCharacter

**Service**: FighterCharacter refines Character

**Observators**:

isBlocking: [FightChar] 🡪 bool

isBlockstunned: [FightChar] 🡪 bool

isHitstunned: [FightChar] 🡪 bool

isTeching: [FightChar] 🡪 bool

techniqueCourante: [FightChar] 🡪 Tech

**pre** tech(C) **requires** isTeching(C)

hitStun: [FightChar] 🡪 int

blockStun: [FightChar] 🡪 int

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() 🡪 techniqueCourrante() != null

isTeching 🡪 ¬isBlocking

isHitStunned 🡪 (frameHitStun() > 0)

isBlocking 🡪 ¬(isHitStunned || isBlockStun) = false

isBlockStunned 🡪 (frameBlockStun() > 0)

**Observators**

**[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)

[**step**]

Combo::frameRestante(step(C)) = Combo::frameRestante() -1 || Combo::frameRestante(step(C)) = 95 #Restart temps combo

isHitFrame

[**takeAttack**] :

isBlocking(takeAttack(d, hs, bs)) = false

isBlocking(C)  ⇒ isBlockstunned(takeAttack(d, hs, bs)))

!isBlocking(C) ⇒ isHitstunned(takeAttack(d, hs, bs)))

isBlocking(C) ⇒ (life(C) = life(takeAttack(d, hs, bs)) &&

bs == getBlockStun (takeAttack(d, hs, bs))

!isBloking(C) ⇒ (life(C) = life(takeAttack(d, hs, bs)) + damage) &&

hs == getHitStun (takeAttack(d, hs, bs))

Combo(takeAttack (d, hs, bs)) = 0

[**endTechnique**]:

isTeching() = false

## Techniques

**Service: Technique**

Type : Hitbox

**Observators**:

damage: [**Technique**] 🡪 int

hitstun: [**Technique**] 🡪 int

blockstun: [**Technique**] 🡪 int

startuptime: [**Technique**] 🡪 int

hittime: [**Technique**] 🡪 int

recoverytime: [**Technique**] 🡪 int

box: [**Technique**] -> Hitbox

frame: [**Technique**] 🡪 int

alreadyTouch : [**Technique**] 🡪 int

**Operators**:

init : int x int x int x int x int x int

step : [**Technique**] x FightChar x FightChar 🡪 [**Technique**]

**pre** step(T) **requires** frame < startuptime + hittime + recoverytime

launchTechnique: [**Technique**] 🡪 [**Technique**]

endTechnique: [**Technique**] 🡪 [**Technique**]

**pre** endTechnique **requires** isTeching()

**Observation**:

[**invariant**] :

Frame <= startUpTime + HitTime + RecoveryTime

[**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(me, other)) = frame(T)+1

(isRecovery(step(me,other)) || isStartUp(step(me,other))) && other::getLife(step(me, other)) = other::getLife()

IsHitTime() && box.collidesWith(other.charBox()) ⇒

alreadyTouch (step(me, other)) = true

IsHitTime() && box.collidesWith(other.charBox()) && !alreadyTouch ⇒ other.takeDamage(damage, hs, bs)

AJOUTER ETAT HITBOX

[**launchTechnique**] :

frame = 0

isTeching() = true