Projet CPS

Street Fighter

MALAMAS Alexis ---- GESNOUIN Kévin

Master 1 Informatique S2
Année 2017

UPMC
SORBONNE UNIVERSITÉS

Sommaire

Manuel d’utilisateur

Notre application s'exécute via le fichier build Xml présent dans notre projet ou en exécutant directement le fichier Main.java

Une fois l’application lancée une image de titre Street Fighter apparaît et il suffit d’appuyer sur n’importe quelle touche du clavier pour passer au choix des personnages. Les deux joueurs doivent alors choisir chacun un personnage parmi les 16 disponibles. Chaque personnage possède des caractéristiques propres (vie, vitesse et dégâts).
Pour choisir un personnage chaque joueur déplace un curseur en utilisant les touches z, q, s et d pour le joueur 1 et les flèches directionnelles pour le joueur 2. La carte de combat peut être choisis en utilisant les touches ‘u’ et ‘i’. Les joueurs 1 et 2 appuient respectivement sur les touches ‘e’ et ‘enter’ pour valider leur personnage. Une fois que les 2 joueurs ont choisi un personnage, le combat débute.

Une fois en combat les touches pour jouer sont les suivantes:

Touches du joueur 1
‘q’ -> déplacement gauche
‘d’ -> déplacement droit
‘z‘ -> sauter
‘s’ -> s’accroupir
‘a’ -> coup de poing
‘e’ -> se protéger
‘r’ -> coup de pied

Touches du joueur 2
flèche haut -> sauter
flèche bas -> s'accroupir
flèche droite -> déplacement droit
flèche gauche -> déplacement gauche
k -> coup de poing
l -> se protéger
m -> coup de pied

Pour afficher les hitBoxs des personnages en jeu, on peut appuyer sur la touche 'p' du clavier. Les hitboxs des coups seront alors également affichés au moment de l'application du coup.

Une fois le combat terminé, la fenêtre de jeu se ferme automatiquement au bout de 5 secondes.

Rapport de projet

Introduction

Le projet a pour objectif la réalisation d'un jeu street fighter spécifié selon la méthode Design-by-Contract. Le projet est développé en langage Java.

Pour réaliser la partie graphique de notre application nous avons choisi d'utiliser la librairie Slick2D car c'est une librairie haut niveau qui simplifie la réalisation de jeu en gérant elle même la boucle de jeu et de rendu (fonctions "update" et "render").

Le code source du projet est également disponible par le lien github suivant:
https://github.com/gesnouin-kevin/JeuDeCombats

Spécifications

Choix des spécifications

Lors de la réalisation du projet, nous avons opté pour différents choix de spécifications. En effet, l'application possède le service Character que nous avons décidé de raffiner en FightChar et le service Hitbox en RectangleHitbox.

En ce qui concerne le service RectangleHitbox, il permet de définir une forme précise à notre box sur le character. Pour cela, nous avons ajouté deux observateurs qui sont height et width qui permettent d'avoir en plus de la position, la hauteur et la largeur du rectangle. Nous avons également ajouté des opérateurs afin de pouvoir modifier la largeur et la hauteur du rectangle lorsque que l'on change d'animation.

Nous avons ajouté des observateurs au service Character qui sont 'jumping', 'running' et 'crouching' permettant d'exprimer le moment où le personnage saute, cours et s'accroupit respectivement. De plus, notre application possède 16 personnages jouables donc l'ajout d'un observator 'numeroCharacter' permet de connaître le personnage joué et ainsi de récuperer dans la classe InformationsCharacters.java les données de ce personnage.

À propos du service FightChar, il permet à notre character de pouvoir attaquer, se défendre et mourir en plus de se déplacer. À cela s'ajoute la mise en place d'une RectangleHitbox nommé 'coupBox' permettant de connaître la position et la taille du coup de notre personnage, de différents observators qui sont 'teching' signifiant que le personnage est en

train de réaliser une technique et ‘stun’ désignant le fait que le joueur ne peut pas bouger pendant quelques secondes.

Nous avons choisi d’ajouter à la fonction step les actions ‘KICK’, ‘PUNCH’ et ‘BLOCK’ qui permettent respectivement de faire un coup de pied, poing et un block. Pour cela, on ajoute les fonctions kick(), punch() et block() permettant de réaliser ces actions.
Pour les fonctions kick() et punch(), nous avons eu des difficultés à trouver des pré et post conditions.
Ainsi pour résoudre notre problème, nous avons d’abord décidé de réaliser l’implémentation afin de trouver plus facilement les pré et post conditions. À la fin de l’implémentation, nous avons ajouté différentes post conditions qui permettent de vérifier si le ‘coupbox’ est à la bonne position et taille. De plus nous avons également ajouté des post conditions afin de vérifier dans le cas d’une collision entre le coup d’un personnage et la hitbox ou la ‘coupBox’ d’un adversaire, si la vie de ce dernier diminue et s’il passe dans un état ‘stun’.

Mise en place de la spécification dans les contrats

En ce qui concerne le service Character, afin de tester si le personnage qui reçoit l’action ‘LEFT’, ‘RIGHT’, ‘NEUTRAL’ fait bien l’action souhaitée qui sont moveLeft(), moveRight() et neutral(), nous testons la position x du personnage. En effet, lorsqu’un personnage se déplace vers la gauche, sa position x diminue. C’est pourquoi, pour l’action ‘LEFT’ par exemple nous enregistrons la position x avant de faire l’action dans ‘positionX_atPre’ ensuite nous faisons l’action et nous testons si la position x ‘getPositionX()’ est inférieure à positionX_atPre.

Pour le service FightChar, l’implémentation des fonctions kick() et punch() dans les contrats se fait de la même façon. En effet, au niveau du contrat il faut tester qu’à la fin de l’action kick ou punch que le ‘coupbox’ soit à la bonne position et taille. C’est pourquoi avant de faire cette action, on enregistre ‘coupbox’ dans ‘coupbox_atPre’. Ensuite on réalise l’action voulue et le ‘coupbox’ est modifié mais pas le ‘coupbox_atPre’. C’est pourquoi on va mettre à jour la position manuellement en utilisant des opérateurs comme ‘setHeight’ afin de simuler l’action et nous vérifions si le ‘coupBox’ à la même position que ‘coupbox_atPre’. Nous répétons cette action pour la taille.

Dans le service RectangleHitbox, il faut pouvoir tester si la hauteur et la largeur soient toujours positives.

Implémentations

Implémentation de Engine
Notre classe engine contient un tableau de Player et une variable command pour chaque joueur. En effet lorsqu'un joueur appuie sur une touche, il ne fait que modifier la valeur de la variable command qui lui correspond. Ce n'est ensuite que dans la boucle de jeu que la fonction step de engine est appelé à chaque frame. Cela nous permet donc de bien associer le fait de traiter deux commandes de joueurs par frame et donc de "contrôler" la fonction keyPressed de slick2D.

Implémentation de Player
La classe player contient le numéro du joueur (1 ou 2) et une instance de FightChar et de Animation. Cette dernière nous permet de gérer l'animation des sprites du personnage et propose également de changer l'animation courante (animation de marche ou animation de mort par exemple).

Implémentation de FightChar
La classe FightChar hérite de la classe Character et ajoute des fonctions liées aux techniques du personnage. La méthode step de Character est enrichie afin de traiter les commandes liées à l'utilisation de techniques.
Dans cette classe nous avons également ajouté les fonctions nextFrameTech et updateDurationStunned qui nous permettent de gérer les frames des techniques. Ainsi on a pu ajouter un temps de préparation de technique, un temps de récupération après son utilisation et les temps liées aux techniques qui "stun" le joueur.

Implémentation de Character
La principale difficulté dans l'implémentation de cette classe venait de la manière dont nous allions ajouter la fonctionnalité de saut pour notre personnage. En effet, dans notre jeu nous n'avons pas de réelle gravité qui est appliquée au joueur à chaque tour de boucle sur les personnages. Pour implémenter le saut nous utilisons donc une fonction parabole inversée qui permet de positionner notre personnage sur l'axe Y où nous le souhaitons afin de recréer un saut.

Tests MBT

Pour mettre en place les tests MBT, nous avons réalisé différents scénarios afin de tester nos classes d'implémentation.

Pour tester l'initialisation dans Engine, nous avons mis en place le scénario suivant. Nous créons tout d'abord notre engine auquel nous affectons deux players. Ensuite, nous associons à ces deux players, deux personnages et nous ajoutons leurs rectangleHitbox. À la suite de ces initialisations nous testons si la width et la height de engine sont correctes. Au niveau des players si leurs positions et face sont correctes et enfin on test si les box sont aux bonnes positions.

Pour tester l'updateFace dans Engine, nous avons mis en place le scénario suivant. Nous créons tout d'abord notre engine auquel nous affectons deux players. Ensuite, nous associons à ces deux players, deux personnages où le personnage 1 à une faceRight à true et le personnage 2 à une faceRight à false que nous faisons bouger dans l'engine afin que leurs faceRight soient modifiés. Nous appelons la fonction updateFace(). Pour finir nous testons si la faceRight du personnage 1 est maintenant à false et celle du personnage 2 à true.

Pour tester le moveLeft dans Character, nous avons mis en place le scénario suivant. Nous créons tout d'abord notre engine auquel nous affectons deux players. Ensuite, nous associons à ces deux players, deux personnages que nous initialisons à une certaine valeur x. Ensuite, nous leur associons leurs rectangleHitbox. Maintenant nous appelons la méthode moveLeft. Nous testons alors que la vie, la vitesse et sa faceRight ne soient pas modifiées. De plus, il faut tester que la position du personnage soit toujours dans l'engine. Donc nous testons si sa position x est entre [0;Engine::width()] et sa position y entre [0;Engine::height()].

Pour tester le kick dans fightChar, nous avons mis en place le scénario suivant. Nous créons tout d'abord notre engine auquel nous affectons deux players. Ensuite, nous associons à ces deux players, deux personnages que nous faisons bouger dans l'engine. Ensuite, nous leur associons leurs rectangleHitbox et nous associons un coupBox à chaque personnage. Maintenant le joueur 2 va donner un kick qui va modifier son coupBox. Alors nous testons que la position du coupBox soit bien mise à jour.

Spécification

Service : Character

Types : bool,int

Observators:
positionX : [Character] -> int
positionY : [Character] -> int
engine : [Character] -> Engine
rectangleHitbox : [Character] -> RectangleHitbox
life : [Character] -> int
const speed : [Character] -> int
faceRight : [Character] -> boolean
dead : [Character] -> boolean
running : [Character] -> boolean
crouching : [Character] -> boolean
jumping : [Character] -> boolean
numeroPlayer : [Character] -> int
numeroCharacter : [Character] -> int

Constructors:
init: int × int × bool × int -> [Character]
pre init(l,s,f,numeroPlayer) **requires** l > 0 ∧ s > 0

Operators:
moveLeft: [Character] → [Character]
moveRight: [Character] → [Character]
switchSide: [Character] → [Character]
step: [Character] × Commande → [Character]
setPositionX: [Character] × int -> [Character]
pre setPositionX(x) requires x >= 0
setPositionY: [Character] × int -> [Character]
pre setPositionY(y) requires y >= 0
setFaceRight: [Character] × boolean -> [Character]
setNumeroPlayer: [Character] × int -> [Character]
pre setNumeroPlayer() requires numeroPlayer>=0 && numeroPlayer<=1
moveUp: [Character] → [Character]
moveDown: [Character] → [Character]
neutral: [Character] → [Character]
updateY: [Character] → [Character]
setLife: [Character] × int -> [Character]
setRectangleHitbox: [Character] × RectangleHitbox -> [Character]
setDead: [Character] × boolean -> [Character]

crouch: [Character] → [Character]

Observations:

[invariant]
getPositionX() > 0 and getPositionX() < Engine::getWidth()
getPositionY() > 0 and getPositionY() < Engine::getHeight()
isDead() = not (getLife() > 0)

[init]
life(init(l, s, f, e)) = l ∧ speed(init(l, s, f, e)) = s ∧ faceRight(init(l, s, f, e)) = f

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

[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))))
⇒ positionX(moveRight(C)) = 0
faceRight(moveRight(C)) = faceRight(C) ∧ life(moveRight(C)) = life(C)
positionY(moveRight(C)) = positionY(C)

[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) = C

[setPositionX]
positionX(C,setPositionX(x)) = x

[setPositionY]
positionY(C,setPositionY(y)) = y

[setFaceRight]
faceRight(C,setFaceRight(fr)) = fr

[setNumeroPlayer]
numeroPlayer(C,setNumeroPlayer(numeroPlayer)) = numeroPlayer

[setNumeroCharacter]
numeroCharacter(C,setNumeroCharacter(numeroCharacter)) = numeroCharacter

[moveDown]
faceRight(C) = faceRight(C)@Pre and Life(C) = Life(C)@Pre
positionY(C) = positionY(C)@Pre

[moveUp]
faceRight(C) = faceRight(C)@Pre and Life(C) = Life(C)@Pre
positionY(C) = positionY(C)@Pre

[neutral]
height(rectangleHitbox(C)) =
InformationsCharacter.getHeightSpritePersoIdle(numeroCharacter(C))
width(rectangleHitbox(C)) =
InformationsCharacter.getWidthSpritePersoIdle(numeroCharacter(C))

[setLife]
life(C,setLife(l))=l

[setDead]
dead(C,setDead(d)) = d

[crouch]
height(rectangleHitbox(C)) =
InformationsCharacter.getHeightSpritePersoIdle(numeroCharacter(C))

width(rectangleHitbox(C)) =
InformationsCharacter.getWidthSpritePersoIdle(numeroCharacter(C))
Service: FightChar **refines** Character

Types: int, boolean

Observators:
blocking: [FightChar] -> bool
hitstunned: [FightChar] -> bool
teching: [FightChar] -> bool
coupBox: [FightChar] -> RectangleHitbox
durationStunned: [FightChar] -> int
frameTech: [FightChar] -> int

Constructors:
init: int × int × bool × int -> [Character]
    pre init(l,s,f,numeroPlayer) **requires** l > 0 ∧ s > 0

Operators:
step: [FightChar] x Command -> [FightChar]
    pre step(c) **require** not isDead()
kick: [FightChar] -> [FightChar]
punch: [FightChar] -> [FightChar]
block: [FightChar] -> [FightChar]
hit: [FightChar] -> [FightChar]
dead: [FightChar] -> [FightChar]
nextFrameTech: [FightChar] -> [FightChar]
setDurationStunned: [FightChar] x int -> [FightChar]
    pre setDurationStunned(bs) **require** bs>0
setCoupBox: [FightChar] -> [FightChar]
updateDurationStunned: [FightChar] -> [FightChar]

Observators:

[step]
step(KICK) = kick()
step(PUNCH) = punch()
step(BLOCK_PRESSED) = block()

[kick]

RectangleHitbox::getPosX()
RectangleHitbox::getPosY()
RectangleHitbox::setPosX()
RectangleHitbox::setPosY()

height(rectangleHitbox(FC))=setHeight(rectangleHitbox(FC)@Pre,InformationsCharacter.getHeightSpritePersoIdle(numeroCharacter(FC)))

width(rectangleHitbox(FC))=setWidth(rectangleHitbox(FC)@Pre,InformationsCharacter.getWidthSpritePersoIdle(numeroCharacter(FC)))

height(coupBox(FC))=setHeight(coupBox(FC)@Pre,InformationsCharacter.getHeightSpritePersoFoot(numeroCharacter(FC)))

width(coupBox(FC))=setWidth(coupBox(FC)@Pre,InformationsCharacter.getWidthSpritePersoFoot(numeroCharacter(FC)))

getPosY(coupBox(FC))=setPosY(coupBox(FC)@Pre,positionY(FC)+InformationsCharacter.getPosYSpritePersoFoot(numeroCharacter(FC))-InformationsCharacter.getHeightSpritePersoFoot(numeroCharacter(FC)))

FaceRight()=>getPosX(coupBox(FC))=setPosX(coupBox(FC)@Pre,positionX(FC)+InformationsCharacter.getWidthSpritePersoKick(numeroCharacter(FC))-InformationsCharacter.getWidthSpritePersoFoot(numeroCharacter(FC)))

not faceRight()
=>getPosX(coupBox(FC))=setPosX(coupBox(FC)@Pre,positionX(FC)+InformationsCharacter.getWidthSpritePersoIdle(numeroCharacter(FC))-InformationsCharacter.getPosXSpritePersoFoot(numeroCharacter(FC))-InformationsCharacter.getWidthSpritePersoFoot(numeroCharacter(FC)))

∃ i in {0,1}
collidesWith(coupBox(),coupBox(fightCharacter(player(engine(FC),i)))) ||
isCollidesWith(coupBox(),coupBox(fightCharacter(player(engine(FC),i))))
	=>!.blocking(fightCharacter(player(engine(FC),i))))
		=>durationStunned(fightCharacter(player(engine(FC),i))))=3

[punch]
Pareil que kick sauf pour :
qu'il faut remplacer Foot par Arm et Kick par Punch

∃ i in {0,1}
collidesWith(coupBox(),coupBox(fightCharacter(player(engine(FC),i)))) ||
isCollidesWith(coupBox(),coupBox(fightCharacter(player(engine(FC),i))))
	=>!.blocking(fightCharacter(player(engine(FC),i))))
		=>durationStunned(fightCharacter(player(engine(FC),i))))=2

[block]
height(rectangleHitbox(FC))=setHeight(rectangleHitbox(FC),InformationsCharacter.getHeightSpritePersoBlocking(numeroCharacter(FC)))

width(rectangleHitbox(FC))=setWidth(rectangleHitbox(FC),InformationsCharacter.getWidthSpritePersoBlocking(numeroCharacter(FC)))

[hit]
otherPlayer correspond au joueur non courant
life(FC)=life()@Pre-InformationsCharacter.getDamage(numeroCharacter(fightCharacter(otherPlayer())))

[dead]
∃ i && j in {0,1}, j!=i , life(fightCharacter(player(engine(),i)))<=0
=> dead(fightCharacter(player(engine(),i)))

[nextFrameTech]
techFrame()<2 => techFrame()==echFrame()@Pre+1
techFrame()>=2 => techFrame()=0 && teching()

[setDurationStunned]
durationStunned()=bs

[setCoupBox]
coupBox(FC,setCoupBox(rectangleHitbox))=rectangleHitbox

[updateDurationStunned]
durationStunned()>0 => durationStunned()=durationStunned()@Pre -1

Service: Hitbox

Types: bool, int

Observators:
positionX: [Hitbox] ->int
positionY: [Hitbox] ->int
belongsTo: [Hitbox] x int x int -> bool
collidesWith: [Hitbox] x Hitbox -> bool
equalsTo: [Hitbox] x Hitbox -> bool

Constructors:
init: int x int -> [Hitbox]

Operators:
moveTo: [Hitbox] x int x int -> [Hitbox]
setPosX: [Hitbox] x int -> [Hitbox]
    pre setPosX(x) **require** x>0
setPosY: [Hitbox] x int -> [Hitbox]
    pre setPosY(y) **require** y>0

Observations:

[invariant]:
collidesWith(H,H1) = ∃ x,y:int int, belongsTo(H,x,y) ^ belongsTo(H1,x,y)
equalsTo(H,H1) = ∀ x,y:int int, belongsTo(H,x,y) = belongsTo(H1,x,y)

[init]:
positionX(init(x,y)) = x
positionY(init(x,y)) = y

[MoveTo]:
positionX(moveTo(H,x,y)) = x
positionY(moveTo(H,x,y)) = y
∀ u,v:int int, belongsTo(moveTo(H,x,y),u,v) =
belongsto(H,u-(x-positionX(H)),v-(y-positionY(H))

[setPosX]
positionX(H,setPosX(x)) = x

[setPosY]
positionY(H,setPosY(y)) = y

Service : RectangleHitbox

Types: bool, int

Observators:
height: [RectangleHitbox] ->int
width: [RectangleHitbox] ->int
collidesWith: [RectangleHitbox] x Hitbox -> bool
equalsTo: [RectangleHitbox] x Hitbox -> bool
posX: [RectangleHitbox] ->int
posY: [RectangleHitbox] ->int

Constructor:
init: int x int x int x int -> [Hitbox]
	pre init(x,y,w,h) **requires** w>0 && h>0

Operators:
setPosX: [Hitbox] x int -> [Hitbox]
	pre setPosX(x) **require** x>0
setPosY: [Hitbox] x int -> [Hitbox]
	pre setPosY(y) **require** y>0
setHeight: [Hitbox] x int -> [Hitbox]
	pre setHeight(h) **require** h>0
setWidth: [Hitbox] x int -> [Hitbox]
	pre setWidth(w) **require** w>0
setWidthHeight: [Hitbox] x int x int -> [Hitbox]
	pre setWidthHeight(w,h) **require** w>0 && h>0

Observations:

[invariant]:
width()>0
height()>0

[setPosX]
posX(RH,setPosX(x)) = x

[setPosY]
posY(RH,setPosY(y)) = y

[setHeight]
height(RH,setHeight(h)) = h

[setWidth]
width(RH,setWidth(w)) = w

[setWidthHeight]
width(RH,setWidthHeight(w,h)) = w
height(RH,setWidthHeight(w,h)) = h

Service: Player

Type: int

Observators:
numeroPlayer: [Player] -> int
animationPlayer: [Player] -> Animation
fightCharacter: [Player] -> FightChar

Constructor:
init : Engine x int -> Player
    pre init(es,numeroPlayer) **requires** numeroPlayer=0 || numeroPlayer=1

Operator:
setAnimationPlayer: [Player] x Animation -> Player

Observations:

[invariant]
numeroPlayer()= 0 || numeroPlayer()= 1

[init]
numeroPlayer(P,init(es,numeroPlayer)) = numeroPlayer
engine(P,init(es,numeroPlayer)) = es

[setAnimationPlayer]
animationPlayer(P,setAnimationPlayer(animationPlayer))=animationPlayer

Service: Engine

Types: bool, int, Command

Observators:
const height: [Engine] -> int
const width: [Engine] -> int
player: [Engine] int -> Player
    pre player(E,i) **requires** i ∈ {1, 2}
gameOver: [Engine] -> bool
commandPlayer1: [Engine] -> Command
commandPlayer2: [Engine] -> Command

Constructors:
init: int x int x int x Player x Player -> [Engine]
    pre init(h,w,s,p1,p2) **requires** h > 0 ^ s > 0 ^ w > s ^ p1 != p2

Operators:
step: [Engine] -> [Engine]
    pre step(E) requires :gameOver(E)
updateFace: [Engine] -> [Engine]
setCommandPlayer1: [Engine] x Command -> [Engine]
    pre setCommandPlayer1(c) require c ∈ Command
setCommandPlayer2: [Engine] x Command -> [Engine]
    pre setCommandPlayer2(c) require c ∈ Command

Observations:

[invariant]:
gameOver(E) = 9i 2 f1; 2g Character ::dead(player(E; i))

[init]:
height(init(h; w; s; p1; p2)) = h
width(init(h; w; s; p1; p2)) = w
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]:
commandPlayer1()==NEUTRAL
commandPlayer2()==NEUTRAL

[updateFace]
faceRight(Player(numeroPlayer(Math.max(posX(player(0)) , posX(player(1))))) = false
faceRight(Player(numeroPlayer(Math.min(posX(player(0)) , posX(player(1))))) = true

[setCommandPlayer1]
commandPlayer1(setCommandPlayer1(commandPlayer1)) = commandPlayer1

[setCommandPlayer2]
commandPlayer2(setCommandPlayer2(commandPlayer2)) = commandPlayer2

Test MBT

Engine

TestInit

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()

Opérations:
L0 = engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 200, true, 0)
fightchar2=init(100, 200, false, 1)
p1.setPositionX(100)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)

Oracle:
height(L0)=800
width(L0)=400
FightChar::positionX(fightchar(L0,player(L0,0))) = 100
FightChar::positionX(fightchar(L0,player(L0,1))) = 300
FightChar::positionY(fightchar(L0,player(L0,0))) = 0
FightChar::positionY(fightchar(L0,player(L0,1))) = 0
FightChar::faceRight(fightchar(L0,player(L0,0))) = true
FightChar::faceRight(fightchar(L0,player(L0,1))) = false
RectangleHitbox::positionX(fightchar(L0,player(L0,0))) = 100
RectangleHitbox::positionX(fightchar(L0,player(L0,1))) = 300
RectangleHitbox::positionY(fightchar(L0,player(L0,0))) = 0
RectangleHitbox::positionY(fightchar(L0,player(L0,1))) = 0
FightChar::dead(fightchar(L0,player(L0,1))) = false
FightChar::dead(fightchar(L0,player(L0,1))) = false

TestStep

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 200, true, 0)
fightchar2=init(100, 200, false, 1)

Opérations:
L0= step()

Oracle:
commandPlayer1=NEUTRAL
commandPlayer2=NEUTRAL
FightChar::dead(fightchar(L0,player(L0,1))) = false
FightChar::dead(fightchar(L0,player(L0,1))) = false

TestUpdateFace

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 200, true, 0)
fightchar2=init(100, 200, false, 1)
p1.setPositionX(100)
p2.setPositionX(400)

Opérations:
L0=updateFace()

Oracle:
FightChar::faceRight(fightchar(L0,player(L0,0))) = true
FightChar::faceRight(fightchar(L0,player(L0,1))) = false
FightChar::dead(fightchar(L0,player(L0,1))) = false
FightChar::dead(fightchar(L0,player(L0,1))) = false

Character

TestInit

Conditions initiales:
char = new Character()

Opération:
L0=char.init(100, 5, true, 0)

Oracle:
FightChar::life(L0)=100
FightChar::speed(L0)=5
FightChar::faceRight(L0)=true

TestMoveLeft

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 5, true, 0)
fightchar2=init(100, 5, false, 1)
p1.setPositionX(5)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)

Opération:
L0=fightchar1.moveLeft()

Oracle:
FightChar::life(fightchar(L0,player(L0,0))) = 100
FightChar::speed(fightchar(L0,player(L0,0)))=5
FightChar::faceRight(fightchar(L0,player(L0,0)))=true
FightChar::positionX(fightchar(L0,player(L0,0)))=0
FightChar::positionY(fightchar(L0,player(L0,0)))=0

FightChar::positionX(fightchar(L0,player(L0,0)))>=0
FightChar::positionY(fightchar(L0,player(L0,0)))>=0
FightChar::positionX(fightchar(L0,player(L0,0)))<Engine::width
FightChar::positionY(fightchar(L0,player(L0,0)))<Engine::height

TestMoveRight

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 6, true, 0)
fightchar2=init(100, 200, false, 1)
p1.setPositionX(6)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)

Opération:
L0=fightchar1.moveRight()

Oracle:
FightChar::life(fightchar(L0,player(L0,0))) = 100
FightChar::speed(fightchar(L0,player(L0,0)))=6
FightChar::faceRight(fightchar(L0,player(L0,0)))=true
FightChar::positionX(fightchar(L0,player(L0,0)))=12
FightChar::positionY(fightchar(L0,player(L0,0)))=0
FightChar::positionX(fightchar(L0,player(L0,0)))>=0
FightChar::positionY(fightchar(L0,player(L0,0)))>=0
FightChar::positionX(fightchar(L0,player(L0,0)))<Engine::width
FightChar::positionY(fightchar(L0,player(L0,0)))<Engine::height

TestMoveUp

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 200, true, 0)
fightchar2=init(100, 200, false, 1)
p1.setPositionX(100)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)

Opération:
L0=fightchar1.moveUp()

Oracle:
FightChar::life(fightchar(L0,player(L0,0))) = 100
FightChar::speed(fightchar(L0,player(L0,0)))=5
FightChar::faceRight(fightchar(L0,player(L0,0)))=true
FightChar::positionX(fightchar(L0,player(L0,0)))=6
FightChar::positionY(fightchar(L0,player(L0,0)))=0
FightChar::positionX(fightchar(L0,player(L0,0)))>=0
FightChar::positionY(fightchar(L0,player(L0,0)))>=0
FightChar::positionX(fightchar(L0,player(L0,0)))<Engine::width
FightChar::positionY(fightchar(L0,player(L0,0)))<Engine::height

TestMoveDown

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 200, true, 0)
fightchar2=init(100, 200, false, 1)

p1.setPositionX(100)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)

Opération:
L0=fightchar1.moveDown()

Oracle:
FightChar::life(fightchar(L0,player(L0,0))) = 100
FightChar::speed(fightchar(L0,player(L0,0)))=5
FightChar::faceRight(fightchar(L0,player(L0,0)))=true
FightChar::positionX(fightchar(L0,player(L0,0)))=6
FightChar::positionY(fightchar(L0,player(L0,0)))=0
FightChar::positionX(fightchar(L0,player(L0,0)))>=0
FightChar::positionY(fightchar(L0,player(L0,0)))>=0
FightChar::positionX(fightchar(L0,player(L0,0)))<Engine::width
FightChar::positionY(fightchar(L0,player(L0,0)))<Engine::height

TestCrouch

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 200, true, 0)
fightchar2=init(100, 200, false, 1)
p1.setPositionX(100)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)

Opération:
L0=fightchar1.crouch()

Oracle:
FightChar::life(fightchar(L0,player(L0,0))) = 100
FightChar::speed(fightchar(L0,player(L0,0)))=5
FightChar::faceRight(fightchar(L0,player(L0,0)))=true
FightChar::positionX(fightchar(L0,player(L0,0)))=6
FightChar::positionY(fightchar(L0,player(L0,0)))=0
FightChar::positionX(fightchar(L0,player(L0,0)))>=0
FightChar::positionY(fightchar(L0,player(L0,0)))>=0
FightChar::positionX(fightchar(L0,player(L0,0)))<Engine::width
FightChar::positionY(fightchar(L0,player(L0,0)))<Engine::height

TestStepNeutral

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 200, true, 0)
fightchar2=init(100, 200, false, 1)
p1.setPositionX(100)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)

Opération:
L0=fightchar1.step(NEUTRAL)

Oracle:
FightChar::life(fightchar(L0,player(L0,0))) = 100
FightChar::speed(fightchar(L0,player(L0,0)))=5
FightChar::faceRight(fightchar(L0,player(L0,0)))=true
FightChar::positionX(fightchar(L0,player(L0,0)))=6
FightChar::positionY(fightchar(L0,player(L0,0)))=0
FightChar::positionX(fightchar(L0,player(L0,0)))>=0
FightChar::positionY(fightchar(L0,player(L0,0)))>=0
FightChar::positionX(fightchar(L0,player(L0,0)))<Engine::width
FightChar::positionY(fightchar(L0,player(L0,0)))<Engine::height

FightChar

TestInit

Conditions initiales:
char = new FightCharacter()

Opération:
L0=char.init(100, 5, true, 0)

Oracle:
FightChar::life(L0)=100
FightChar::speed(L0)=5
FightChar::faceRight(L0)=true

TestPunch

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 5, true, 0)
fightchar2=init(100, 5, false, 1)
p1.setPositionX(5)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)
r1.getCoupBox().init(5, 0, 100, 200)
r2.getCoupBox().init(5, 0, 100, 200)

Opération:
L0=fightchar2.punch()

Oracle:
width(coupBox(fightchar(L0,player(L0,0))))=
InformationsCharacter.getHeightSpritePersoArm(0)

height(coupBox(fightchar(L0,player(L0,0)))) =
InformationsCharacter.getWidthSpritePersoArm(0)

positionX(coupBox(fightchar(L0,player(L0,0))))=positionX()+InformationsCharacter.getWidth
SpritePersoIdle(0)-InformationsCharacter.getPosXSpritePersoArm(0)-InformationsCharacter
.getWidthSpritePersoArm(0)

positionY(coupBox(fightchar(L0,player(L0,0))))=positionY()+InformationsCharacter.getPosY
SpritePersoArm(0)-InformationsCharacter.getHeightSpritePersoArm(0)

life(fightchar(L0,player(L0,0))) = 100
speed(fightchar(L0,player(L0,0)))=5
faceRight(fightchar(L0,player(L0,0)))=true

TestKick

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 5, true, 0)
fightchar2=init(100, 5, false, 1)
p1.setPositionX(5)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)
r1.getCoupBox().init(5, 0, 100, 200)
r2.getCoupBox().init(5, 0, 100, 200)

Opération:
L0=fightchar2.kick()

Oracle:
width(coupBox(fightchar(L0,player(L0,0))))=
InformationsCharacter.getHeightSpritePersoFoot(0)
height(coupBox(fightchar(L0,player(L0,0)))) =
InformationsCharacter.getWidthSpritePersoFoot(0)

positionX(coupBox(fightchar(L0,player(L0,0))))=positionX()+InformationsCharacter.getWidth
SpritePersoIdle(0)-InformationsCharacter.getPosXSpritePersoFoot(0)-InformationsCharacter
.getWidthSpritePersoFoot(0)

positionY(coupBox(fightchar(L0,player(L0,0))))=positionY()+InformationsCharacter.getPosY
SpritePersoFoot(0)-InformationsCharacter.getHeightSpritePersoFoot(0)

life(fightchar(L0,player(L0,0))) = 100
speed(fightchar(L0,player(L0,0)))=5
faceRight(fightchar(L0,player(L0,0)))=true

TestBlock

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 5, true, 0)
fightchar2=init(100, 5, false, 1)
p1.setPositionX(5)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)
r1.getCoupBox().init(5, 0, 100, 200)
r2.getCoupBox().init(5, 0, 100, 200)

Opération:
L0=fightchar2.block()

Oracle:
RectangleHitbox::width(fightchar(L0,player(L0,1))) = 200
RectangleHitbox::height(fightchar(L0,player(L0,1))) = 100

life(fightchar(L0,player(L0,0))) = 100
speed(fightchar(L0,player(L0,0)))=5
faceRight(fightchar(L0,player(L0,0)))=true

TestDead

Conditions initiales:
engine = new EngineImpl()
p1 = new PlayerImpl()
p2 = new PlayerImpl()
engine.init(800, 400, 200, p1, p2)
p1=(engine,0)
p2=(engine,1)
fightchar1=init(100, 5, true, 0)
fightchar2=init(100, 5, false, 1)
p1.setPositionX(5)
p2.setPositionX(300)
r1=new RectangleHitboxImpl()
r2=new RectangleHitboxImpl()
p1.setRectangleHitbox(r1)
p2.setRectangleHitbox(r2)
r1.init(100, 0, 100, 200)
r2.init(300, 0, 100, 200)
r1.getCoupBox().init(5, 0, 100, 200)
r2.getCoupBox().init(5, 0, 100, 200)

Opération:
L0=fightchar2.dead()

Oracle:
life(fightchar(L0,player(L0,0))) = 100
speed(fightchar(L0,player(L0,0)))=5
faceRight(fightchar(L0,player(L0,0)))=true

Hitbox

TestInit

Conditions initiales:
hitbox=new Hitbox()

Opération:
L0=hitbox.init(10,10)

Oracle:
positionX(L0)=10
positionY(L0)=10

TestMoveTo

Conditions initiales:
hitbox=new Hitbox()
hitbox.init(10,10)

Opération:
L0=hitbox.moveTo(100,100)

Oracle:
positionX(L0)=100
positionY(L0)=100

TestSetPosX

Conditions initiales:
hitbox=new Hitbox()
hitbox.init(10,10)

Opération:
L0=hitbox.setPosX(42)

Oracle:
positionX(L0)=42

TestSetPosY

Conditions initiales:
hitbox=new Hitbox()
hitbox.init(10,10)

Opération:
L0=hitbox.setPosY(42)

Oracle:
positionY(L0)=42

RectangleHitbox

TestInit

Conditions initiales:
hitbox=new RectangleHitbox()

Opération:
L0=hitbox.init(10,10,100,100)

Oracle:
positionX(L0)=10
positionY(L0)=10
width(L0)=100
height(L0)=100
width(L0)>0
height(L0)>0

TestSetters

Conditions initiales:
hitbox=new RectangleHitbox()
hitbox.init(10,10,100,100)

Opération:
L0=hitbox.setPosX(24)
L0=hitbox.setPosY(42)
L0=hitbox.setWidth(450)
L0=hitbox.setHeight(365)
L1=hitbox.setWidthHeight(874,456)

Oracle:
positionX(L0)=24
positionY(L0)=42
width(L0)=450
height(L0)=365

width(L1)=874
height(L1)=456
width(L1)>0
height(L1)>0

Player

TestInit

Conditions initiales:
player1 = new Player()
player2 = new Player()
engine = new Engine()
engine.init(800, 400, 300, player1, player2)

Opération:
L0=player1.init(engine, 0)
L0=player2.init(engine, 1)

Oracle:
numeroPlayer(L0,player1)=0
numeroPlayer(L0,player2)=1

TestSetA

Conditions initiales:
player1 = new Player()
player2 = new Player()
engine = new Engine()
engine.init(800, 400, 300, player1, player2)

Opération:
L0=player1.init(engine, 0)
L0=player2.init(engine, 1)

Oracle:
numeroPlayer(L0,player1)=0
numeroPlayer(L0,player2)=1