Pseudocode

Class Hitbox(string type, int damage):

Type [pHitbox1, pHitbox2, pHitbox3, pHitboxAir, eHitbox1, …]

Int [5, 3, 7, 5, 4, …] \*hitpoints lost

Class lastCheckpoint(int x, int y)

Position when level starts is always at the beginning

Changes when touching checkpoint

MoveInput():

If canMove == True{

if input == leftKey{

xAxis = -1

}

else if input == rightKey{

xAxis = 1

}

else{

xAxis = 0

}

}

Move():

sprite = runAnimation

if xAxis < 0 {

flipSprite = True

}

If xAxis > 0 {

flipSprite = False

}

xPosition = xAxis \* speed

JumpInput():

If feetPosition == platform{

grounded = True

}

Else{

grounded = False

}

If input == jumpKey && Grounded == True && canJump == True{

yAxis = 1

}

Else{

yAxis = 0

}

Jump():

sprite = jumpAnimation

yVelocity = yAxis \* jumpForce

if xAxis < 0 {

flipSprite = True

}

If xAxis > 0 {

flipSprite = False

}

AttackInput():

If input == fire1Key{

regularAttack = True

}

Else{

regularAttack = False

}

Attack():

attackTime = 0.2f

followUpTime = 0.4f

If regularAttack == True && grounded == True{

If attackTime > 0{

sprite = attackAnimation1

hitboxAppear(pHitbox1.position, radius, eHurtbox)

attackTime -= Time.deltaTime

followUpTime -= Time.deltaTime

}

\*first attack of combo, finish if button isn’t pressed

elseIf attackTime <= 0 && followUpTime > 0{

sprite = attackAnimation2

hitboxAppear(pHitbox2.position, radius, eHurtbox)

\*constant rapid appearing hitbox

followUpTime -= Time.deltaTime

while regularAttack == True && followUpTime > 0{

followUpTime = 0.2f

\*reset timer to continue while button is pressed

sprite = attackAnimation3

hitboxAppear(pHitbox3.position, radius, eHurtbox)

\*if button isn’t pressed during rapid attack, use finish

}

}

If regularAttack == True && grounded == False{

If If attackTime > 0{

sprite = attackAnimationAir

hitboxAppear(pHitboxAir.position, radius, eHurtbox)

attackTime -= Time.deltaTime

}

attackTime = 0.2f

followUpTime = 0.4f

}

GuardInput():

If input == fire2Key && xAxis == 0{

guard = True

}

Else{

guard = False

}

Guard():

While guard == true{

Sprite = blockAnimation

hurtboxAppear(pBlock.position, radius, eHitbox)

}

DodgeInput():

If input == fire2Key && xAxis != 0{

dodge = True

}

Else{

dodge = False

}

Dodge():

dodgeTime = 0.3f

If dodge == True && Grounded == True && canDodge == True{

if xAxis < 0 {

flipSprite = True

}

If xAxis > 0 {

flipSprite = False

}

canDodge = False

sprite = dodgeAnimation

invincible = true

dodgeTime -= Time.deltaTime

invincible = false

canDodge = true

}

SpecialAttackInput():

If input == fire3Key{

If input == upKey{

special = upSpecial

}

Elif input == downKey{

special = downSpecial

}

Else{

special = neutralSpecial

}

}

\*inputs may be more like get functions, as there can be more than one input

SpecialAttack():

If special = neutralSpecial{

If player.energy > 8{

Player.energy -= 8

attackTimer = 0.3f

sprite = nSpecialAnimation

attackTimer -= Time.deltaTime

Instantiate(electricBall, throwPoint.position, throwPoint.rotation)

}

}

\*throw ball of electricity, can’t make other inputs until thrown

If special = upSpecial{

If player.energy > 4{

Player.energy -= 4

attackTimer = 1.0f

sprite = upSpecialAnimation

hitboxAppear(pUpSpec.position, radius, eHurtbox)

attackTimer -= Time.deltaTime

while attackTimer > 0 & attackTimer < 0.8f{

xPosition = xAxis \* speed

yPosition = speed

}

}

}

\*launch in direction for time specified

If special = downSpecial{

If player.energy > 8{

Player.energy -= 8

attackTimer = 0.6f

sprite = downSpecialAnimation

hitboxAppear(pDownSpec.position, radius, eHurtbox)

}

}

\*single large launching hitbox

HurtboxCollision():

If hitbox.owner == player{

If collider(hitbox.owner) == enemy{

EnemyHealth.loseHealth(hitbox.damage)

if hitbox.type == knockback{

Enemy.falldown()

}

If hitbox.type == flinch{

Enemy.TakeDamageE()

}

}

}

elif hitbox.owner == enemy{

If collider(hitbox.owner) == player{

PlayerHealth.loseHealth(hitbox.damage)

if hitbox.type == knockback{

Player.falldown()

}

If hitbox.type == flinch{

Player.TakeDamageAni()

}

}

}

HitboxCollision():

\*\*

TakeDamageAni():

flinchTime = 0.5f

if player.position < eHitbox.position{

flipSprite = True

}

Elif player.position > eHitbox.position{

flipSprite = False

}

\*if same, stick to current sprite flip position

sprite = flinchAnimationPlayer

flinchTime -= Time.deltaTime

TakeDamageE():

flinchTime = 0.5f

if enemy.position < pHitbox.position{

flipSprite = True

}

Elif enemy.position > pHitbox.position{

flipSprite = False

}

\*if same, stick to current sprite flip position

sprite = getflinchAnimation(enemyType) \*animation depends on enemy

flinchTime -= Time.deltaTime

FallDown():

if player.position < eHitbox.position{

flipSprite = True

xAxis = -1

}

Elif player.position > eHitbox.position{

flipSprite = False

xAxis = 1

}

distanceLaunched = xAxis \* 3

downtime = 1.5f

invincible = true

sprite = fallAnimationPlayer()

flinchTime -= Time.deltaTime

invincible = false

GetUp():

If input = Any{

sprite = getUpAnimationPlayer()

}

MoveE():

sprite = runAnimationE

if player.position < enemy.position {

flipSprite = True

xAxisE = -1

}

elif player.position > enemy.position {

flipSprite = False

xAxisE = 1

}

Else{

xAxisE = 0

}

xPosition = xAxisE \* speed

RandomAction():

xAxisDistance = 2

If enemy.position >= player.position – xAxisDistance || enemy.position <= player.position + xAxisDistance{ \*enemy will perform an action when in distance with the player

randomNumber = generateRandomNumber(10)

if randomNumber < 2{

GuardE()

}

Elif randomNumber > 8{

If player.grounded == True{

AirAttackE()

}

Else{

AttackE()

}

Else{

Elif randomNumber > 8{

If player.grounded == True{

AttackE()

}

Else{

AirAttackE()

}

}

}

GuardE(): \*block for a few seconds

blockTime = 1.5f

Sprite = blockAnimation

hurtboxAppear(pBlock.position, radius, eHitbox)

blockTime -= Time.deltaTime()

AttackE():

attackTime = enemy.attackTime

sprite = attackAnimationE

hitboxAppear(eHitbox.position, radius, pHurtbox)

attackTime -= Time.deltaTime

AirAttackE():

attackTime = enemy.attackTime

enemyHeight = enemy.attackHeight

sprite = airAttackAnimationE

hitboxAppear(eHitbox.position, radius, pHurtbox)

attackTime -= Time.deltaTime

JumpE(): \*if enemy is running into a wall

If enemy.collider() == wall{

sprite = jumpAnimation

yVelocity = enemy.jumpForce

}

LoseHealth(damage){

hitPoints -= damage

if hitPoints <= 0{

if PlayerHealth{

Death()

}

If EnemyHealth{

DeathE()

}

Death():

Sprite = fallAnimationPlayer()

screenFade()

player.position = lastCheckpoint.position

Restore()

Restore():

hitPoints.player == maxHitPoints.player

Heal():

If player.position == checkpoint.position{

Restore()

}

\*Best if hurtbox collision responds when passed by

SaveLast():

If player.position == checkpoint.position{

lastCheckpoint.poisition = checkpoint.position

}

UpdateEnergy():

energyGrow = 1.0f

if player.energy < 10{

energyGrow -= Time.deltaTime

if energyGrow <= 0{

player.energy += 1

energyGrow = 1.0f

}

}

EnergyMeter():

energyPointsText.text = “Energy Points: “ + energyPoints.points.ToString()

HealthBar():

healthPointsText.text = “Energy Points: “ + healthPoints.points.ToString()

PauseInput():

If input == pauseKey{

If isPaused == True{

ResumeGame()

}

Else{

PauseGame()

}

}

PauseGame():

Time.timescale = 0

pausePanel.setActive(True)

isPaused = True

ResumeGame():

Time.timescale = 1

pausePanel.setActive(False)

isPaused = False