**PSUEDOCODE**

**MainMenu extends World**

BEGIN mainProgram()

set background color to Color(5,5,50))

create image with text "Operation Firestorm"

create image displaying highscore

create LevelButton(1)

create LevelButton(2)

create LevelButton (3)

create DifficultyButton difficultyButton

create ClassButton classButton

END mainProgram

**GameEndScreen extends World**

// create game end screen

BEGIN MainProgram(boolean winState, int score)

set background color to Color(5,5,50))

IF (winstate = TRUE) THEN  
 create Image with text “You Win”

ELSE  
 create Image with text “You Lose”

ENDIF

IF (GameSettings.checkScore()) THEN   
 create Image with text “NEW HIGH SCORE” + GameSettings.getCurrentScore()

ELSE  
 create Image with text “Score: “ + GameSettings.getCurrentScore()

create Image with text “High Score: “ + GameSettings.getHighScore()

ENDIF

Create ReturnButton returnButton

END MainProgram

**Level extends World**

// Creates level based on an array, iterates through each number in the array and places it into the world.

int levelXOffset = -1  
int levelYOffset = -1  
int levelLength  
int[] levelStructure

List BoxWall boxWalls  
List NPC npcs  
List Flag flags

BEGIN MainProgram

set background color to Color(5,5,50))

END MainProgram

BEGIN constructLevel(int[] levelStructure, int levelLength)

// create UI elements and initiate createLevel

My levelStructure = levelStructure

My levelLength = levelLength

Call createLevel()

Create HealthBar() healthBar

Create WeaponUI() weaponUI

Create ScoreBoard() scoreboard

END constructLevel

BEGIN createLevel()

// build the level using given information from child classes

int i = 0

FOR (int object IN levelStructure)

addGameObject(object, ((i % levelLength) \* 90) + (90 \* levelXOffset), ((int) Math.floor(i / levelLength) \* 90) + (90 \* levelYOffset))  
i++

ENDFOR

Create Player() player

Create Camera() camera

END createLevel

BEGIN addGameObject(int objectType, int x, int y)

// read each digit of the object list

IF (objectType == 1) THEN

Add (new BoxWall(x,y)) to boxWalls

ELSEIF (objectType == 2) THEN

Add (new NPC(x,y)) to npcs

ELSEIF (objectType == 3) THEN

Add (new NavPoint(x,y)) to navPoints

ELSEIF (objectType == 4) THEN

Add (new Flag(x,y)) to flags

ENDIF

END addGameObject

BEGIN createGameObjects()

// check each list and create the objects to the world

FOR (BoxWall boxWall IN boxWalls)

Create boxWall

ENDFOR

FOR (NPC npc IN npcs)

Create npc

ENDFOR

FOR (Flag flag IN flags)

Create flag

ENDFOR

END createGameObjects

**Level1 extends Level**

int levelLength = 20

int[] levelStructure = {

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

1,0,0,1,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,1,

1,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,1,

1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,1,

1,0,1,1,0,0,0,1,1,0,0,0,1,0,0,0,1,0,0,1,

1,0,0,1,0,0,0,1,1,0,2,0,1,0,0,0,0,0,0,1,

1,0,0,1,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,1,

1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,

1,0,1,1,1,0,0,1,1,0,0,0,0,0,1,0,0,0,2,1,

1,0,0,1,0,0,0,1,1,0,0,0,0,0,1,0,0,1,0,1,

1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,1,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

}

BEGIN MainProgram()

constructLevel(levelStructure, levelLength)

END MainProgram

**Level 2 extends Level**

int levelLength = 40

int[] levelStructure = {

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,

1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,

1,0,1,1,1,1,0,0,1,1,0,0,1,2,2,0,0,0,0,0,1,0,0,0,2,0,0,0,0,0,1,0,0,2,0,0,0,0,0,1,

1,0,0,0,0,1,1,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,0,1,

1,0,0,0,0,0,0,0,0,0,0,2,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,

1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,

1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,2,1,0,0,0,1,0,0,0,0,1,2,0,0,0,0,0,0,0,2,1,2,0,0,1,

1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,0,0,0,1,

1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,1,1,4,0,1,0,0,0,0,0,1,

1,0,0,1,1,1,1,0,0,1,2,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,1,

1,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,1,0,0,0,1,0,0,2,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,

1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,2,0,1,0,0,0,0,0,1,0,0,0,1,1,1,1,0,0,0,0,0,1,

1,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,1,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

}

BEGIN MainProgram()

constructLevel(levelStructure, levelLength)

END MainProgram

**Level 3 extends Level**

int levelLength = 40

int[] levelStructure = {

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

1,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,1,0,0,0,0,0,0,0,2,1,0,0,0,1,1,1,0,0,1,0,0,0,0,0,1,

1,0,0,0,0,1,1,1,1,1,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,0,2,0,1,1,0,0,0,0,0,0,1,0,0,1,

1,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0,1,1,1,1,

1,0,0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,

1,0,0,1,0,2,0,1,0,0,1,1,1,0,0,2,0,1,2,1,2,1,1,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,2,0,1,

1,0,0,1,1,0,0,1,0,2,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,0,2,0,1,1,1,1,0,0,1,

1,0,0,0,1,1,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,1,2,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,

1,0,0,0,0,1,1,1,1,1,1,0,0,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,2,1,0,0,0,0,0,0,0,0,0,0,0,1,

1,0,2,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,0,1,0,0,0,2,0,0,1,1,1,0,0,0,1,1,0,0,0,0,1,

1,0,1,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,2,1,1,0,1,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,1,

1,0,1,0,0,0,1,1,1,1,0,0,1,2,0,1,0,0,0,0,0,0,1,0,2,0,0,2,2,1,0,0,0,1,1,1,1,1,0,1,1,1,

1,0,1,1,0,0,1,1,1,1,0,0,0,1,0,1,0,0,1,1,0,1,1,0,0,0,0,0,0,1,0,2,0,0,0,0,0,0,0,0,0,1,

1,0,1,1,0,0,0,1,1,0,0,0,0,0,0,1,0,0,1,0,0,2,1,0,0,0,2,0,0,1,0,0,0,0,0,1,1,0,0,0,0,1,

1,0,1,1,0,2,0,0,0,0,1,1,1,0,0,0,0,0,0,0,4,0,0,0,2,0,0,0,2,1,0,0,0,0,0,1,1,0,0,2,0,1,

1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,1,2,0,0,1,1,1,1,1,1,1,1,0,1,1,0,0,1,1,0,0,0,0,1,

1,0,0,0,1,0,0,1,0,1,1,0,1,1,1,0,0,0,1,1,0,1,1,2,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0,1,1,1,

1,0,2,1,1,1,0,1,0,1,1,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,1,1,1,

1,0,0,1,1,1,1,1,0,1,1,0,0,1,1,1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,1,0,0,0,0,1,0,0,0,1,1,1,

1,0,0,0,1,1,1,1,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,1,1,0,0,0,1,1,1,

1,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,1,0,0,2,0,0,1,0,1,0,2,0,0,0,0,0,1,1,1,0,1,0,0,0,1,

1,0,1,0,0,1,1,1,0,1,1,1,0,1,0,0,0,1,1,1,1,1,1,1,0,1,0,0,0,0,0,0,1,1,1,1,0,1,0,2,0,1,

1,0,1,0,0,1,1,1,0,1,1,0,0,1,1,0,0,0,0,1,0,0,0,0,0,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,

1,0,1,0,0,1,2,1,0,0,1,0,1,1,1,1,0,0,0,1,0,0,2,0,0,0,0,0,0,0,1,0,0,1,0,0,0,1,1,1,0,1,

1,0,1,0,0,1,0,1,1,0,1,0,0,1,1,1,1,0,0,0,0,0,1,1,1,0,1,1,1,0,1,0,0,1,0,0,0,1,0,0,0,1,

1,0,0,0,0,0,0,1,1,0,1,1,0,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,0,0,2,0,1,0,0,0,0,0,0,0,1,

1,0,1,0,1,1,1,1,1,0,1,1,0,1,1,1,1,1,1,0,1,0,0,1,1,1,1,1,1,0,1,0,0,1,0,0,1,1,1,1,0,1,

1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,1,1,0,0,1,0,0,1,0,0,0,0,0,0,0,1,

1,0,1,1,1,1,0,2,1,1,1,1,1,1,0,1,1,1,1,0,0,0,1,1,1,1,1,1,1,0,1,0,0,1,1,1,1,1,0,0,0,1,

1,0,0,0,0,1,1,1,1,0,0,2,1,1,0,1,1,1,1,0,2,0,1,1,1,0,1,1,1,0,1,0,0,0,0,0,0,0,2,0,0,1,

1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

}

BEGIN MainProgram()

constructLevel(levelStructure, levelLength)

END MainProgram

**Button extends Actor**

// Base class for buttons, contains information for if mouse is hovering over the button

int clickBoundaryX

int clickBoundaryY

int clickYOffset

int clickXOffset

BEGIN wasClicked(int x, int y)

// is the mouse hovering over the mouse

IF (x + clickXOffset > getX() - (clickBoundaryX / 2) AND

x + clickXOffset < getX() + (clickBoundaryX / 2) AND

y + clickYOffset > getY() - (clickBoundaryY / 2) AND

y + clickYOffset < getY() + (clickBoundaryY / 2)) THEN

RETURN true

ELSE

RETURN false

ENDIF

END wasClicked

**ClassButton extends Button**

// Button to change between classes in game

static int playerClass = 1

BEGIN ClassButton()

super(400,100,0,-50)

END ClassButton

BEGIN mainProgram()

// draw button

GameSettings.setPlayerClass(playerClass)

set my color to new Color(5,5,50)

IF (wasClicked(mouseX, mouseY)) THEN

IF (mouse was clicked) THEN

cycleClass()

ELSE

set my color to new Color(100,100,150)

ENDIF

ENDIF

string className = ""

IF (playerClass == 1) THEN

className = "Assault"

ELSE IF (playerClass = 2) THEN

className = "Recon"

ENDIF

create a box with a white outline and filled currentColor

create text image with text "Class: " + className

END mainProgram

BEGIN cycleClass()

// toggle between player classes

playerClass++

IF (playerClass > 2) THEN

playerClass = 1

ENDIF

END cycleClass

**DifficultyButton extends Button**

// Button to toggle between difficulties

static int difficulty = 1

BEGIN DifficultyButton()

Super(400, 100, 0, -50)

END DifficultyButton

BEGIN mainProgram()

// draw button

GameSettings.setDifficulty(difficulty)

GameSettings.setPlayerClass(playerClass)

set my color to new Color(5,5,50)

IF (wasClicked(mouseX, mouseY)) THEN

IF (mouse was clicked) THEN

cycleDifficulty()

ELSE

set my color to new Color(100,100,150)

ENDIF

ENDIF

String difficultyName = “”

IF (difficulty == 1) THEN

difficultyName = “Easy”

ELSE IF (difficulty == 2) THEN

difficultyName = “Normal”

ELSE IF (difficulty == 3) THEN

difficultyName = “Hard”

ENDIF

create a box with a white outline and filled currentColor

create text image with text "Difficulty: " + difficultyName

END mainProgram

BEGIN cycleDifficulty()

// toggle difficulty

difficulty++

IF (difficulty > 3) THEN

difficulty = 1

ENDIF

END cycleDifficulty

**LevelButton extends Button**

Int level

BEGIN levelButton(int level)

Super(200,100,0,0)

My level = level

Load and set my to ("level" + level + ".png")

END levelButton

BEGIN mainProgram

IF (wasClicked(mouseX, mouseY)) THEN

IF (mouse was clicked) THEN

loadMyLevel()

ENDIF

ENDIF

END mainProgram

BEGIN loadMyLevel()

// on click load the desired level

IF (level == 1) THEN

Create Level1 level1

Load level1

ELSEIF (level == 2) THEN

Create Level2 level2

Load level2

ELSEIF (level == 3) THEN

Create Level3 level3

Load level3

ENDIF

END loadMyLevel

**ReturnButton extends Button**

// draw the button

BEGIN ReturnButton()

super(400,100,0,-50)

END ReturnButton

BEGIN mainProgram

IF (wasClicked(mouseX, mouseY)) THEN

IF (mouse was clicked) THEN

returnToMenu()

ELSE

set my color to new Color(100,100,150)

ENDIF

ENDIF

create a box with a white outline and filled currentColor

create text image with text "RETURN"

END mainProgram

BEGIN returnToMenu()

// on click return to the menu

Create MainMenu mainMenu

Load mainMenu

END RETURNToMenu

**Camera extends Actor**

// static information about camera position.

static int camX

static int camY

static int targetCamX

static int targetCamY

static double cameraSpeed

BEGIN Camera (int camX, int camY)

my camX = camX

my camY = camY

cameraSpeed = 10

END Camera

BEGIN mainProgram()

// use proportional controller to move camera

camX += round((targetCamX - camX) / cameraSpeed)

camY += round((targetCamY - camY) / cameraSpeed)

END mainProgram

**GameObject extends Actor**

// Base class for any object that moves according to camera position

int fieldX

int fieldY

BEGIN GameObject(int x, int y)

my fieldX = x

my fieldY = y

END GameObject

BEGIN updateLocation()

// set my position on screen based on static camera position and object position

setLocation (fieldX - Camera.getCamX(), fieldY - Camera.getCamY())

END updateLocation

**BloodSplatter extends GameObject**

// Blood particle

BEGIN BloodSplatter(int x, int y, int rot)

super(x,y)

set image scale to (140, 120)

setRotation to rot

END BloodSplatter

BEGIN mainProgram

// decrease transparency and then destroy

updateLocation()

IF (my transparency <= 5) THEN

getWorld().removeObject(this)

ELSE

decrease my transparency by 5

ENDIF

END mainProgram

**BoxWall extends GameObject**

// wall for game, enemies and players cannot walk through this

Int bounds

BEGIN BoxWall(int x, int y)

Super (x,y)

bounds = 60

END

BEGIN mainProgram()

updateLocation

END mainProgram

BEGIN getColliderBounds()

RETURN colliderBounds

END getColliderBounds

**Bullet extends GameObject**

// in game projectile, deals damage on collision with an enemy or player

int speed

World world

int damage

long cullTime

BEGIN Bullet(int x, int y, int direction, int speed, int size, int damage)

super(x,y)

cullTime = current time + 400

setRotation(direction)

my scale = (size, size/2)

my speed = speed

my damage = damage

END Bullet

BEGIN mainProgram()

collisionCheck()

timedDestroy()

updateLocation()

moveX(round(cos(degrees to radians(getRotation())) \* speed)

moveY(round(sin(degrees to radians(getRotation())) \* speed)

END mainProgram

BEGIN timedDestroy()

IF (current time > cullTime) THEN

getWorld().removeObject(this)

ENDIF

END timedDestroy

BEGIN collisionCheck()

//check if touching an object

boolean willRemove = false

IF (touching an NPC) THEN

NPC npc = the NPC that was hit

npc.hit(damage)

willRemove = true

create a BloodSplatter(getFieldX(), getFieldY(), getRotation())

ELSEIF (touching a Player) THEN

Player player = the Player that was hit

player.hit(damage)

willRemove = true

create a BloodSplatter(getFieldX(), getFieldY(), getRotation())

ELSEIF (touching a BoxWall) THEN

willRemove = true

ENDIF

IF (willRemove is TRUE) THEN

getWorld().removeObject(this)

ENDIF

END collisionCheck

**Flag extends GameObject**

// player wins on touching this

BEGIN Flag(int x, int y)

Super(x,y)

Set my image to green square

END Flag

BEGIN mainProgram()

updateLocation()

END mainProgram

**MuzzleFlash extends GameObject**

// particle effects

BEGIN MuzzleFlash(int x, int y, int rot)

super(x,y)

set image scale to (100, 80)

setRotation to rot

END BloodSplatter

BEGIN mainProgram

// decrease sensitivity and then destroy

updateLocation()

IF (my transparency <= 45) THEN

getWorld().removeObject(this)

ELSE

decrease my transparency by 45

ENDIF

END mainProgram

**NPC extends GameObject**

// base class for enemies

int speed

int colliderRadius = 30

int health

long currentTime = System.currentTimeMillis()

long nextShotDue

int shootSpread

int fireRate

int difficulty

int direction

int FOV

int reactionTime

long reactionDue

int turnSpeed

boolean hasSeenPlayer

long relaxTime

int damage

boolean alerted

int checkFrame = 0

BEGIN NPC(int x, int y)

Super(x,y)

health = 100

set my image scale to (240, 200)

speed = 1  
 direction = 4

FOV = 120

turnSpeed = 5

difficulty = GameSettings.getDifficulty()

IF (difficulty == 1) THEN

difficulty = 1

reactionTime = 1000

damage = 10

fireRate = 150

shootSpread = 35

ELSEIF (difficulty == 2) THEN

difficulty = 2

reactionTime = 500

damage = 25

fireRate = 150

shootSpread = 27

ELSEIF (difficulty == 3) THEN

difficulty = 3

reactionTime = 100

damage = 35

fireRate = 20

shootSpread = 15

ENDIF

END NPC

BEGIN mainProgram()

getWorld().setPaintOrder(NPC class)

currentTime = System. currentTimeMillis()

updateLocation()

IF (difficulty == 1) THEN

runDifficultyEasy()

ELSEIF (difficulty == 2) THEN

runDifficultyMedium()

ELSEIF (difficulty == 3) THEN

runDifficultyHard()

ENDIF

checkFrame = checkFrame + 1

IF (checkFrame >= 10) THEN

checkFrame = 1

ENDIF

END mainProgram

BEGIN runDifficultyEasy()

// AI for easy difficulty

IF (health > 0) THEN

IF (hasSeenPlayer OR alerted) THEN

aimAtPlayer()

IF (currentTime > relaxTime) THEN

alerted = false

ENDIF

ELSEIF (currentTime > relaxTime) THEN

aimAtRotation(direction\*90)

ENDIF

IF (Greenfoot.getRandomNumber(50) == 1) THEN

direction = Greenfoot.getRandomNumber(4) + 1

ENDIF

IF (currentTime > nextShotDue && checkFrame == 1) THEN

IF (findTarget(1200)) THEN

shoot()

ENDIF

nextShotDue = currentTime + fireRate

ENDIF

ELSE

fadeAway()

ENDIF

END runDifficultlyEasy

BEGIN runDifficultyMedium

// AI for Medium difficulty

IF (health > 0) THEN

IF (hasSeenPlayer OR alerted) THEN

aimAtPlayer()

IF (currentTime > relaxTime) THEN

alerted = false

ENDIF

ELSEIF (currentTime > relaxTime) THEN

aimAtRotation(direction\*90)

ENDIF

IF (!collisionCheck(direction)) THEN

IF (direction == 1) THEN

moveY(-speed)

ENDIF

IF (direction == 2) THEN

moveX(speed)

ENDIF

if (direction == 3) THEN

moveY(speed)

ENDIF

if (direction == 4) THEN

moveX(-speed)

ENDIF

ELSE

direction = Greenfoot.getRandomNumber(4) + 1

ENDIF

IF (currentTime > nextShotDue && checkFrame == 1) THEN

IF (findTarget(1200)) THEN

shoot()

ENDIF

nextShotDue = currentTime + fireRate

ENDIF

ELSE

fadeAway()

ENDIF

END runDifficultyMedium

BEGIN runDifficultyHard()

// AI for hard difficulty

IF (health > 0) THEN

IF (hasSeenPlayer OR alerted) THEN

aimAtPlayer()

IF (currentTime > relaxTime) THEN

alerted = false

ENDIF

ELSEIF (currentTime > relaxTime) THEN

aimAtRotation(direction\*90)

ENDIF

IF (!collisionCheck(direction)) THEN

IF (direction == 1) THEN

moveY(-speed)

ENDIF

IF (direction == 2) THEN

moveX(speed)

ENDIF

if (direction == 3) THEN

moveY(speed)

ENDIF

if (direction == 4) THEN

moveX(-speed)

ENDIF

ELSE

direction = Greenfoot.getRandomNumber(4) + 1

ENDIF

IF (currentTime > nextShotDue AND checkFrame == 1) THEN

IF (findTarget(1200)) THEN

shoot()

ENDIF

nextShotDue = currentTime + fireRate

ENDIF

ELSE

fadeAway()

ENDIF

END runDifficultyHard

BEGIN hit(int damage)  
 // decrease health when called

health -= damage

END hit

BEGIN fadeAway()

// decrease transparency and then destroy

IF (my transparency <= 30) THEN

getWorld().removeObject(this)

ELSE

decrease my transparency by 30

ENDIF

END fadeAway()

BEGIN getColliderRadius()

// return the bounds in which this object occupies

RETURN colliderRadius

END getColliderRadius

BEGIN aimAtPlayer()

// aim at the player using a P-controller

List players = getWorld().getObjects(Player.class)

IF (players.size() > 0)

int rotationForce

Player player = players.get(0)

int targetRotation = rotation from me to player

int currentRotation = get my rotation

IF (currentRotation > 360) THEN

currentRotation = currentRotation – 360

ENDIF

IF (currentRotation >= 270 AND targetRotation <= 90) THEN

rotationForce = (targetRotation + 360 - currentRotation) / turnSpeed

ELSEIF (currentRotation <= 90 AND targetRotation >= 270) THEN

rotationForce = (-currentRotation + -(360 - targetRotation)) / turnSpeed

ELSE

rotationForce = (targetRotation - currentRotation) / turnSpeed

ENDIF

setRotation(getRotation() + rotationForce)

ENDIF

END aimAtPlayer

BEGIN aimAtRotation

// aim at a defined rotation using a P-Controller

Int rotationForce

Int targetRotation = rotation

int currentRotation = get my rotation

IF (currentRotation > 360) THEN

currentRotation = currentRotation – 360

ENDIF

IF (currentRotation >= 270 AND targetRotation <= 90) THEN

rotationForce = (targetRotation + 360 - currentRotation) / turnSpeed ELSEIF (currentRotation <= 90 AND targetRotation >= 270) THEN

rotationForce = (-currentRotation + -(360 - targetRotation)) / turnSpeed

ELSE

rotationForce = (targetRotation - currentRotation) / turnSpeed

ENDIF

setRotation(getRotation() + rotationForce)

END aimAtRotation

BEGIN shoot()

// fire a bullet in current rotation

playSound(“HK416.mp3”)

int barrelOffset

int barrelXOffset = 16

int barrelYOffset = 55

double alpha

double theta = Math.toDegrees(Math.atan(barrelYOffset / barrelXOffset))

double h = Math.sqrt(Math.pow(barrelXOffset, 2) + Math.pow(barrelYOffset, 2))

alpha = getRotation() – theta

int worldXOffset = Math.round(Math.cos(Math.toRadians(alpha)) \* h)

int worldYOffset = Math.round(Math.sin(Math.toRadians(alpha)) \* h)

create Bullet(getFieldX() + worldXOffset, getFieldY() + worldYOffset, getRotation() - 90 + ((shootSpread / 2) - getRandomNumber(shootSpread)), 50, 40, damage)

create muzzleFlash(getFieldX() + worldXOffset, getFieldY() + worldYOffset, getRotation() - 90 + (int) (5 - Greenfoot.getRandomNumber(5))

END shoot()  
  
BEGIN findTarget()

// check if player is within vision

Double checkRotationRadians = 0

Boolean isVisible = true

hasSeenPlayer = false

List players getWorld().getObjects(Player.class)  
 Player player = null

IF (players.size() > 0) THEN

player = players.get(0)

checkRotationRadians = Math.atan2(player.getFieldY() - this.getFieldY(), player.getFieldX() - this.getFieldX()) + Math.toRadians(90)  
 int yDifference = player.getFieldY() - this.getFieldY()

int xDifference = player.getFieldX() - this.getFieldX()

int distanceToPlayer = Math.round(Math.sqrt(Math.pow(yDifference,2) + Math.pow(xDifference,2)))  
 IF (distanceToPlayer < range) THEN

int rotationToPlayer = Math.round (Math.toDegrees(Math.atan2(player.getFieldY() - this.getFieldY(), player.getFieldX() - this.getFieldX())))  
 List BoxWalls boxWalls = getWorld().getObjects(BoxWall.class)

IF (walls != null) THEN

int rotationToPlayer = Math.round (Math.toDegrees(Math.atan2(player.getFieldY() - this.getFieldY(), player.getFieldX() - this.getFieldX())))

List BoxWalls walls = getWorld().getObjects(BoxWall.class)  
 IF (walls != null) THEN

FOR (BoxWall boxWall IN walls)

int[][] cornerPositions = new int[4][2]

cornerPositions[0][0] = boxWall.getFieldX() - boxWall.getColliderBounds()

cornerPositions[0][1] = boxWall.getFieldY() + boxWall.getColliderBounds()

cornerPositions[1][0] = boxWall.getFieldX() + boxWall.getColliderBounds()

cornerPositions[1][1] = boxWall.getFieldY() + boxWall.getColliderBounds()

cornerPositions[2][0] = boxWall.getFieldX() + boxWall.getColliderBounds()

cornerPositions[2][1] = boxWall.getFieldY() - boxWall.getColliderBounds()

cornerPositions[3][0] = boxWall.getFieldX() - boxWall.getColliderBounds()

cornerPositions[3][1] = boxWall.getFieldY() - boxWall.getColliderBounds()

int[] rotationCorners = new int[4]

rotationCorners[0] = Math.round (Math.toDegrees(Math.atan2(cornerPositions[0][1] - this.getFieldY(), cornerPositions[0][0] - this.getFieldX())))

rotationCorners[1] = Math.round (Math.toDegrees(Math.atan2(cornerPositions[1][1] - this.getFieldY(), cornerPositions[1][0] - this.getFieldX())))

rotationCorners[2] = (int) Math.round (Math.toDegrees(Math.atan2(cornerPositions[2][1] - this.getFieldY(), cornerPositions[2][0] - this.getFieldX())))

rotationCorners[3] = (int) Math.round (Math.toDegrees(Math.atan2(cornerPositions[3][1] - this.getFieldY(), cornerPositions[3][0] - this.getFieldX())))

int largest = rotationCorners[0]

int smallest = rotationCorners[0]

int i

WHILE (i < rotationCorners.length)

IF (rotationCorners[i] > largest) THEN

largest = rotationCorners[i]

ENDIF  
 i = i + 1

ENDWHILE

WHILE (i < rotationCorners.length)

IF (rotationCorners[i] < smallest) THEN

smallest = rotationCorners[i]

ENDIF

i = i + 1

ENDWHILE

double distanceToWall = Math.sqrt((Math.pow(boxWall.getFieldX() - this.getFieldX(), 2) + (Math.pow(boxWall.getFieldY() - this.getFieldY(), 2))))

IF (distanceToWall < distanceToPlayer) THEN

IF (smallest < 0 AND largest > 0) THEN

IF (smallest <= - 90) THEN

IF (rotationToPlayer < smallest AND rotationToPlayer >= -180 OR rotationToPlayer > largest AND rotationToPlayer <= 180) THEN

hasSeenPlayer = false

isVisible = false

ENDIF

ELSEIF (smallest > -90) THEN

IF (rotationToPlayer > smallest AND rotationToPlayer <= 0 OR rotationToPlayer < largest AND rotationToPlayer >= 0) THEN

hasSeenPlayer = false

isVisible = false

ENDIF

ENDIF

ELSEIF (rotationToPlayer > smallest AND rotationToPlayer < largest AND distanceToWall < distanceToPlayer) THEN

hasSeenPlayer = false

isVisible = false

ENDIF

ENDFOR  
 IF (rotationToPlayer > 180 AND getRotation() - 90 < 0) THEN

IF ((getRotation() - 90) + (360 - rotationToPlayer) > (FOV/2)) THEN

hasSeenPlayer = false

isVisible = false

ENDIF

ELSEIF (rotationToPlayer < 0 && getRotation() - 90 > 180) THEN

IF ((rotationToPlayer - 90) + (360 - getRotation()) > (FOV/2)) THEN

hasSeenPlayer = false

isVisible = false

ENDIF

ELSEIF (getRotation() - 90 > rotationToPlayer + (FOV/2) || getRotation() - 90 < rotationToPlayer - (FOV/2)) THEN

hasSeenPlayer = false  
 isVisible = false

ENDIF

IF (hasSeenPlayer)

IF (currentTime < reactionDue) THEN

isVisible = false

ENDIF

relaxTime = currentTime+4000

ELSE

reactionDue = currentTime+reactionTime

ENDIF

RETURN isVisible

ELSE

RETURN false

ENDIF

ENDIF

ENDIF

RETURN false

END findTarget()

BEGIN collisionCheck(int direction)

// check if there is a wall in the way

List BoxWall boxWalls = getWorld().getObjects(BoxWall.class)

IF (direction == 1) THEN

IF (boxWalls != null) THEN

FOR (BoxWall boxWall IN boxWalls)

IF (getFieldY() - 5 < boxWall.getFieldY() + boxWall.getColliderBounds() AND getFieldY() - 5 > boxWall.getFieldY() - boxWall.getColliderBounds() AND getFieldX() < boxWall.getFieldX() + boxWall.getColliderBounds() AND getFieldX() > (boxWall.getFieldX() - boxWall.getColliderBounds())) THEN

RETURN true

ENDIF

ENDFOR

RETURN false

ENDIF

RETURN false

ELSEIF (direction == 2) THEN

IF (boxWalls != null)

FOR (BoxWall boxWall IN boxWalls)

IF (getFieldX() + 5 > boxWall.getFieldX() - boxWall.getColliderBounds() AND getFieldX() + 5 < boxWall.getFieldX() + boxWall.getColliderBounds() AND getFieldY() < boxWall.getFieldY() + boxWall.getColliderBounds() AND getFieldY() > (boxWall.getFieldY() - boxWall.getColliderBounds())) THEN

RETURN true

ENDIF

ENDFOR

RETURN false

ENDIF

RETURN false

ELSEIF (direction == 3) THEN

IF (boxWalls != null) THEN

FOR (BoxWall boxWall IN boxWalls)

IF (getFieldY() + 5 > boxWall.getFieldY() - boxWall.getColliderBounds() AND getFieldY() + 5 < boxWall.getFieldY() + boxWall.getColliderBounds() AND getFieldX() < boxWall.getFieldX() + boxWall.getColliderBounds() AND getFieldX() > (boxWall.getFieldX() - boxWall.getColliderBounds())) THEN

RETURN true

ENDIF

ENDFOR

RETURN false

ENDIF

RETURN false

ELSEIF (direction == 4) THEN

IF (boxWall != null) THEN

FOR (BoxWall boxWall in boxWalls)

IF (getFieldX() - 5 < boxWall.getFieldX() + boxWall.getColliderBounds() AND getFieldX() - 5 > boxWall.getFieldX() - boxWall.getColliderBounds() AND getFieldY() < boxWall.getFieldY() + boxWall.getColliderBounds() AND getFieldY() > (boxWall.getFieldY() - boxWall.getColliderBounds())) THEN

RETURN true

ENDIF

ENDFOR

RETURN false

ENDIF

RETURN false

ELSE

RETURN false

ENDIF

END collisionCheck

BEGIN setVisible(Boolean isVisible)  
 // called by the player to set visibility

IF (visible AND health > 0) THEN

getImage().setTransparency(255)

ELSEIF (health > 0) THEN

getImage().setTransparency(getImage().getTransparency() -Math.ceil(getImage().getTransparency())) / 5)

ENDIF

END setVisible

BEGIN addRelaxTimer(int time)

relaxTime = currentTime + time

END addRelaxTimer

BEGIN alert()

alerted = true

relaxTimer = currentTimer + 1000

END alert

**Player extends GameObject**

// Base class for player

int speed

double cameraBias // a higher bias will mean the camera drifts further away

boolean mouseDown

int health

int colliderRadius = 30

int fireRate

int nextShotAvailable

double spreadCurrent

double spreadMin

double spreadMax

double spreadShotGain

double spreadRecover

int maxAmmo

int currentAmmo

long reloadFinishTime

boolean isReloading

int reloadTime

int gunType

int FOV

int damage

int checkFrame

long nextScoreLost

BEGIN Player(int x, int y)

Super(x,y)

GameSettings.setCurrentScore(0)

nextShotAvailable = System.currentTimeMillis()

reloadFinishTime = System.currentTimeMillis()

GameSettings.setCurrentScore(150)

IF (GameSettings.getPlayerClass() == 1) THEN // class 1 - assault

Set image scale to (240, 200)

speed = 2

cameraBias = 0.35

health = 100

fireRate = 100

spreadMin = 1

spreadMax = 120

spreadShotGain = 30

spreadRecover = 5

spreadCurrent = spreadMin

isReloading = false

maxAmmo = 30

currentAmmo = maxAmmo

reloadTime = 3000

gunType = 1

FOV = 110

damage = 20

ELSEIF (GameSettings.getPlayerClass() == 2) THEN // class 2 - recon

Set image scale to (240, 200)

speed = 2

cameraBias = 0.7

health = 60

fireRate = 220

spreadMin = 1

spreadMax = 45

spreadShotGain = 30

spreadRecover = 15

spreadCurrent = spreadMin

isReloading = false

maxAmmo = 12

currentAmmo = maxAmmo

reloadTime = 3000

gunType = 2

FOV = 110

damage = 40

ENDIF

END Player

BEGIN mainProgram()

If (System.currentTimeMillis() > nextScoreLost) THEN

nextScoreLost = System.currentTimeMillis() + 1000

IF (GameSettings.getCurrentScore() > 0) THEN

GameSettings.addCurrentScore(-5)

ENDIF

ENDIF

getWorld().setPaintOrder(Player.class)

IF (health > 0) THEN

// movement code uses GameObject class

IF (“W” is pressed AND !collisionCheck(1)) THEN

moveY(-speed)

ENDIF

IF (“S” is pressed AND !collisionCheck(3)) THEN

moveY(speed)

ENDIF

IF (“A” is pressed AND !collisionCheck(4)) THEN

moveX(-speed)

ENDIF

IF (“D” is pressed AND !collisionCheck(2)) THEN

moveX(speed)

ENDIF

IF (”R” is pressed AND !isReloading) THEN

Play sound (“AR-15 Reload.wav”)

isReloading = true

reloadFinishTime = System.currentTimeMillis()+reloadTime

ENDIF

IF (System.currentTimeMillis() > reloadFinishTime AND isReloading) THEN

isReloading = false

currentAmmo = maxAmmo

updateAmmoCount()

ENDIF

LookAtPosition(mouseInfo.getX(), mouseInfo.getY())

updateCamPlayerOffset(mouseInfo.getX(), mouseInfo.getY())

IF (mouse is pressed) THEN

mouseDown = true

ELSEIF (mouse is clicked) THEN

mouseDown = false

ENDIF

IF (mouseDown AND weaponReady()) THEN

IF (!isReloading AND currentAmmo > 0) THEN

shoot()

ENDIF

ENDIF

ELSE

fadeAway()

ENDIF

checkFrame = checkFrame + 1

IF (checkFrame >= 5) THEN

checkFrame = 1

ENDIF

updateLocation()

updateWeaponControl()

updateAmmoCount()

updateHealthBar()

IF (checkFrame == 1) THEN

updateEnemyVisibility(1200)  
 List Flag flags = getWorld().getObjects(Flag.class)

FOR (Flag flag IN flags)

IF (Math.sqrt(Math.pow(flag.getFieldX() - this.getFieldX(), 2) + Math.pow(flag.getFieldY() - this.getFieldY(), 2)) <= 100) THEN

winLevel()

ENDIF

ENDFOR

ENDIF

END mainProgram

BEGIN lookAtPosition(int x,int y)

// rotate towards a certain position

setRotation(Math.round(Math.toDegrees((Math.atan2(y + Camera.getCamY() - getFieldY(), x + Camera.getCamX() - getFieldX())))) + 90)

END lookAtPosition

BEGIN updateCamPlayerOffset(int x, int y)

// update the target position of the camera based on my current position

Camera.setTargetCamX(Math.round(this.getFieldX() + ((x - getWorld().getWidth() / 2) \* cameraBias) - (getWorld().getWidth() / 2)))

Camera.setTargetCamY(Math.round (this.getFieldY() + ((y - getWorld().getHeight() / 2) \* cameraBias) - (getWorld().getHeight() / 2)))

END updateCamPlayerOffset

BEGIN getSpeed()

RETURN speed

END getSpeed

BEGIN setSpeed(int speed)

My speed = speed

END setSpeed

BEGIN shoot()

// shoot a bullet

IF (gunType == 1) THEN

playSound(“M4 Sounds.mp3”)

ELSEIF (gunType == 2) THEN

playSound(“Suppressed Mk14.mp3”)

ENDIF

currentAmmo—

updateAmmoCount()

// ensure that the bullet comes out of the point of the gun

int barrelXOffset = 13

int barrelYOffset = 55

double alpha

double theta = Math.toDegrees(Math.atan(barrelYOffset / barrelXOffset))

double h = Math.sqrt(Math.pow(barrelXOffset, 2) + Math.pow(barrelYOffset, 2))

alpha = getRotation() – theta

int worldXOffset = Math.round(Math.cos(Math.toRadians(alpha)) \* h)

int worldYOffset = Math.round(Math.sin(Math.toRadians(alpha)) \* h)

create Bullet(getFieldX() + worldXOffset, getFieldY() + worldYOffset, getRotation() - 90 + ((shootSpread / 2) - getRandomNumber(shootSpread)), 50, 40, damage)

create muzzleFlash(getFieldX() + worldXOffset, getFieldY() + worldYOffset, getRotation() - 90 + (int) (5 - Greenfoot.getRandomNumber(5))

spreadCurrent += spreadShotGain

IF (gunType == 1) THEN

List NPC npcs = getWorld().getObjects(NPC.class)  
 FOR (NPC npc IN npcs)

npc.alert()

ENDFOR

ENDIF

END shoot

BEGIN hit()

// reduce health on hit

health -= damage

updateHealthBar()

END hit

BEGIN updateHealthBar()

// update health UI element

List HealthBar healthBars = getWorld().getObjects(WeaponUI.class)

FOR (HealthBar healthBar IN healthBars)

healthBar.setHealth(health)

ENDFOR

END updateHealthBar

BEGIN updateAmmoCount()

// update gun UI element

List WeaponUI weaponUIs = getWorld().getObjects(WeaponUI.class)

FOR (WeaponUI weaponUI IN weaponUIs)

weaponUI.setAmmoCount(currentAmmo,maxAmmo,gunType)

ENDFOR

END updateAmmoCount

BEGIN getColliderRadius()

// return space that the object takes up

RETURN colliderRadius

END getColliderRadius

BEGIN fadeAway()

// decrease transparency and load game end screen

Decrease transparency by 30

IF (transparency <= 30) THEN

Delay for 25 milliseconds

GameEndScreen gameEnd = new GameEndScreen(false, GameSettings.getCurrentScore())

Create level GameEndScreen gameEnd

Load level gameEnd

END fadeAway

BEGIN collisionCheck()

// check if the moving direction is obstructed by a wall

// 1 – up

// 2 – right

// 3 – down

// 4 - left

List BoxWall boxWalls = getWorld().getObjects(BoxWall.class)

IF (direction == 1) THEN

IF (boxWalls != null) THEN

FOR (BoxWall boxWall IN boxWalls)

IF (getFieldY() - 5 < boxWall.getFieldY() + boxWall.getColliderBounds() AND getFieldY() - 5 > boxWall.getFieldY() - boxWall.getColliderBounds() AND getFieldX() < boxWall.getFieldX() + boxWall.getColliderBounds() AND getFieldX() > (boxWall.getFieldX() - boxWall.getColliderBounds())) THEN

RETURN true

ENDIF

ENDFOR

RETURN false

ENDIF

RETURN false

ELSEIF (direction == 2) THEN

IF (boxWalls != null)

FOR (BoxWall boxWall IN boxWalls)

IF (getFieldX() + 5 > boxWall.getFieldX() - boxWall.getColliderBounds() AND getFieldX() + 5 < boxWall.getFieldX() + boxWall.getColliderBounds() AND getFieldY() < boxWall.getFieldY() + boxWall.getColliderBounds() AND getFieldY() > (boxWall.getFieldY() - boxWall.getColliderBounds())) THEN

RETURN true

ENDIF

ENDFOR

RETURN false

ENDIF

RETURN false

ELSEIF (direction == 3) THEN

IF (boxWalls != null) THEN

FOR (BoxWall boxWall IN boxWalls)

IF (getFieldY() + 5 > boxWall.getFieldY() - boxWall.getColliderBounds() AND getFieldY() + 5 < boxWall.getFieldY() + boxWall.getColliderBounds() AND getFieldX() < boxWall.getFieldX() + boxWall.getColliderBounds() AND getFieldX() > (boxWall.getFieldX() - boxWall.getColliderBounds())) THEN

RETURN true

ENDIF

ENDFOR

RETURN false

ENDIF

RETURN false

ELSEIF (direction == 4) THEN

IF (boxWall != null) THEN

FOR (BoxWall boxWall in boxWalls)

IF (getFieldX() - 5 < boxWall.getFieldX() + boxWall.getColliderBounds() AND getFieldX() - 5 > boxWall.getFieldX() - boxWall.getColliderBounds() AND getFieldY() < boxWall.getFieldY() + boxWall.getColliderBounds() AND getFieldY() > (boxWall.getFieldY() - boxWall.getColliderBounds())) THEN

RETURN true

ENDIF

ENDFOR

RETURN false

ENDIF

RETURN false

ELSE

RETURN false

ENDIF

END collisionCheck

BEGIN weaponReady()

// check if weapon is cooled

IF (System.currentTimeMillis() >= nextShotAvailable) THEN

nextShotAvailable = System.currentTimeMillis() + fireRate

RETURN true

ELSE

RETURN false

ENDIF

END weaponReady

BEGIN updateWeaponControl()

// update recoil spread values over frames

IF (spreadCurrent > spreadMax) THEN

spreadCurrent = spreadMax

ENDIF

spreadCurrent += (spreadMin - spreadCurrent) / spreadRecover

END weaponReady

BEGIN winLevel()

// called when the player touches the flag

IF (GameSettings.getDifficulty() == 1) THEN

GameSettings.addCurrentScore(500)

ELSEIF (GameSettings.getDifficulty() == 2) THEN

GameSettings.addCurrentScore(1000)

ELSEIF (GameSettings.getDifficulty() == 3) THEN

GameSettings.addCurrentScore(2000)

ENDIF

Delay 30 milliseconds

create and load a GameEndScreen (true, GameSettings.getCurrentScore())

END winLevel

BEGIN updateEnemyVisiblity(int range)

// check each enemy and see if they are within the player's vision

Double checkRotationRadians = 0

Boolean isVisible = true

hasSeenNpcs = false

List npcs getWorld().getObjects(NPC.class)  
 NPC npc = null

IF (npcs.size() > 0) THEN

npcs = npcs.get(0)

checkRotationRadians = Math.atan2(npcs.getFieldY() - this.getFieldY(), npcs.getFieldX() - this.getFieldX()) + Math.toRadians(90)  
 int yDifference = npcs.getFieldY() - this.getFieldY()

int xDifference = npcs.getFieldX() - this.getFieldX()

int distanceToNPC = Math.round(Math.sqrt(Math.pow(yDifference,2) + Math.pow(xDifference,2)))  
 IF (distanceToNPC < range) THEN

int rotationToNpcs = Math.round (Math.toDegrees(Math.atan2(npcs.getFieldY() - this.getFieldY(), npcs.getFieldX() - this.getFieldX())))  
 List BoxWalls boxWalls = getWorld().getObjects(BoxWall.class)

IF (walls != null) THEN

int rotationToNpcs = Math.round (Math.toDegrees(Math.atan2(npcs.getFieldY() - this.getFieldY(), npcs.getFieldX() - this.getFieldX())))

List BoxWalls walls = getWorld().getObjects(BoxWall.class)  
 IF (walls != null) THEN

FOR (BoxWall boxWall IN walls)

int[][] cornerPositions = new int[4][2]

cornerPositions[0][0] = boxWall.getFieldX() - boxWall.getColliderBounds()

cornerPositions[0][1] = boxWall.getFieldY() + boxWall.getColliderBounds()

cornerPositions[1][0] = boxWall.getFieldX() + boxWall.getColliderBounds()

cornerPositions[1][1] = boxWall.getFieldY() + boxWall.getColliderBounds()

cornerPositions[2][0] = boxWall.getFieldX() + boxWall.getColliderBounds()

cornerPositions[2][1] = boxWall.getFieldY() - boxWall.getColliderBounds()

cornerPositions[3][0] = boxWall.getFieldX() - boxWall.getColliderBounds()

cornerPositions[3][1] = boxWall.getFieldY() - boxWall.getColliderBounds()

int[] rotationCorners = new int[4]

rotationCorners[0] = Math.round (Math.toDegrees(Math.atan2(cornerPositions[0][1] - this.getFieldY(), cornerPositions[0][0] - this.getFieldX())))

rotationCorners[1] = Math.round (Math.toDegrees(Math.atan2(cornerPositions[1][1] - this.getFieldY(), cornerPositions[1][0] - this.getFieldX())))

rotationCorners[2] = (int) Math.round (Math.toDegrees(Math.atan2(cornerPositions[2][1] - this.getFieldY(), cornerPositions[2][0] - this.getFieldX())))

rotationCorners[3] = (int) Math.round (Math.toDegrees(Math.atan2(cornerPositions[3][1] - this.getFieldY(), cornerPositions[3][0] - this.getFieldX())))

int largest = rotationCorners[0]

int smallest = rotationCorners[0]

int i

WHILE (i < rotationCorners.length)

IF (rotationCorners[i] > largest) THEN

largest = rotationCorners[i]

ENDIF  
 i = i + 1

ENDWHILE

WHILE (i < rotationCorners.length)

IF (rotationCorners[i] < smallest) THEN

smallest = rotationCorners[i]

ENDIF

i = i + 1

ENDWHILE

double distanceToWall = Math.sqrt((Math.pow(boxWall.getFieldX() - this.getFieldX(), 2) + (Math.pow(boxWall.getFieldY() - this.getFieldY(), 2))))

IF (distanceToWall < distanceToNPC) THEN

IF (smallest < 0 AND largest > 0) THEN

IF (smallest <= - 90) THEN

IF (rotationToNpcs < smallest AND rotationToNpcs >= -180 OR rotationToNpcs > largest AND rotationToNpcs <= 180) THEN

hasSeenNpcs = false

isVisible = false

ENDIF

ELSEIF (smallest > -90) THEN

IF (rotationToNpcs > smallest AND rotationToNpcs <= 0 OR rotationToNpcs < largest AND rotationToNpcs >= 0) THEN

hasSeenNpcs = false

isVisible = false

ENDIF

ENDIF

ELSEIF (rotationToNpcs > smallest AND rotationToNpcs < largest AND distanceToWall < distanceToNPC) THEN

hasSeenNpcs = false

isVisible = false

ENDIF

ENDFOR  
 IF (rotationToNpcs > 180 AND getRotation() - 90 < 0) THEN

IF ((getRotation() - 90) + (360 - rotationToNpcs) > (FOV/2)) THEN

hasSeenNpcs = false

isVisible = false

ENDIF

ELSEIF (rotationToNpcs < 0 AND getRotation() - 90 > 180) THEN

IF ((rotationToNpcs - 90) + (360 - getRotation()) > (FOV/2)) THEN

hasSeenNpcs = false

isVisible = false

ENDIF

ELSEIF (getRotation() - 90 > rotationToNpcs + (FOV/2) || getRotation() - 90 < rotationToNpcs - (FOV/2)) THEN

hasSeenNpcs = false  
 isVisible = false

ENDIF

IF (hasSeenNpcs)

IF (currentTime < reactionDue) THEN

isVisible = false

ENDIF

relaxTime = currentTime+4000

ELSE

reactionDue = currentTime+reactionTime

ENDIF

npc.setVisible(isVisible)

ELSE

npc.setVisible(isVisible)

ENDIF

ENDIF

ENDIF

npc.setVisible(isVisible)

END updateEnemyVisibility

**GameSettings extends Actor**

// Holds game settings and player preferences

static int difficulty

static int playerClass

static int highScore

static int currentScore

BEGIN updateGameSettings()

UserInfo info = UserInfo.getMyInfo()

highScore = info.getScore()

END updateGameSettings

BEGIN GameSettings()

difficulty = 2

END GameSettings

BEGIN static getHighScore()

RETURN info.getScore()

END getHighScore

// if the current score exceeds the high score set the userInfo score to the new score and then return true

BEGIN static checkScore()

IF (currentScore > highscore on info) THEN

Info highscore = currentScore

RETURN true

ELSE

RETURN false

ENDIF

END checkScore

BEGIN setPlayerClass(int newClass)

playerClass = newClass

END setPlayerClass

BEGIN setDifficulty(int newDifficulty)

Difficulty = newDifficulty

END setDifficulty

**HealthBar extends Actor**

// UI element for health bar

int health

int traceHealth

BEGIN HealthBar()

Health = 100

traceHealth = 100

END HealthBar

BEGIN mainProgram()

// draw health bar

Create rectangle with color black width: 600

Create rectangle with color red width: traceHealth \* 6

Create rectangle with color black width: health \* 6

IF (traceHealth > health) THEN

traceHealth -= 1

ENDIF

END mainProgram

BEGIN setHealth(int health)

My health = health

END setHealth

**Scoreboard extends Actor**

Int score

// draw a scoreboard based on current score

BEGIN mainProgram()

score = GameSettings.getCurrentScore()

create blue rectangle

create text image showing score

END mainProgram

**WeaponUI extends Actor**

Int currentAmmo

Int maxAmmo

BEGIN setAmmoCount (int currentAmmo, int maxAmmo, int gunType)

// set weapon image and current ammo

My currentAmmo = currentAmmo

My maxAmmo = maxAmmo

Create rectangle in bottom right corner

Create image text showing (currentAmmo + "/" + maxAmmo)

IF (gunType == 1) THEN

Create an image ("M4 Icon.png")

ELSEIF (gunType == 2) THEN

Create an image (“Mk14 Icon.png”)

ENDIF

END setAmmoCount