Computing IPM Analyses

# Ashley Solano Hernandez 9C

Class:

Description:

|  |  |
| --- | --- |
| public class PlayGround extends World  {    /\*\*  \* Constructor for objects of class PlayGround.  \*  \*/  public PlayGround()  {  super(1000,600,1);  start();  }  private void start()  {  addObject(new Game(), 250, 25);  }  } | This is the constructor for the class PlayGround.  This is a method which sets the world size to a rectangle of 1000 by 600 squares, each being one pixel wide |

public abstract class AbstractHero extends Actor

{

protected int health;

protected int speed;

protected Weapon currentWeapon;

protected GreenfootSound heroSpawn = new GreenfootSound("Hero-ok.wav");

protected GreenfootSound heroDeath = new GreenfootSound("Hero-death.wav");

public void act() {

move();

shoot();

collide();

}

public int getHealth() {

return this.health;

}

public void move()

{

if(Greenfoot.isKeyDown("up"))

{

setLocation(getX(),getY()-speed);

}

if(Greenfoot.isKeyDown("down"))

{

setLocation(getX(),getY()+speed);

}

if(Greenfoot.isKeyDown("right"))

{

move(speed);

}

if(Greenfoot.isKeyDown("left"))

{

move(-speed);

}

}

public void shoot()

{

if(Greenfoot.isKeyDown("space"))

{

getWorld().addObject((Actor)this.currentWeapon, 0, 0);

this.currentWeapon.shoot(getX(),getY());

}

}

public void collide()

{

Actor actor;

actor = getOneObjectAtOffset(10,10, AbstractEnemies.class);

if (actor != null)

{

AbstractEnemies enemy = (AbstractEnemies) actor;

this.health = this.health - enemy.getDamage();

getWorld().removeObject(enemy);

die();

}

}

public void die() {

if (health < 0) {

java.util.List gameList = getWorld().getObjects(Game.class);

Game game = (Game)gameList.get(0);

heroDeath.setVolume(100);

heroDeath.play();

game.stop();

getWorld().removeObject(this);

}

}

}

public class Zoom extends AbstractHero

{

public Zoom (){

super ();

this.health = 15;

this.speed = 10;

this.currentWeapon = new MachineGun();

heroSpawn.setVolume(100);

heroSpawn.play();

}

}

public class AbstractBullets extends Actor implements Ammo

{

protected int speed;

protected int damage;

protected boolean active = true;

public void act() {

move();

exitWorld();

collision();

}

public int getDamage(){

return this.damage;

}

public AbstractBullets(int pSpeed) {

speed = pSpeed;

}

public void move() {

move(speed);

}

public void exitWorld() {

if (this.active && getX() >= 995) {

getWorld().removeObject(this);

this.active = false;

}

}

public void collision(){

if(this.active ) {

getWorld().removeObject(this);

this.active = false;

}

}

}

public class MagicBullet extends AbstractBullets

{

public void act() {

move();

exitWorld();

}

public MagicBullet() {

super (10);

damage = 2;

}

}

public class AbstractEnemies extends Actor

{

protected int health;

protected int speed;

protected int damage;

protected boolean active;

protected int points;

public AbstractEnemies() {

this.active = true;

}

/\*\*

\* Act - do whatever the Enemy wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

public void act()

{

move();

collide();

exitWorld();

}

public int getDamage(){

return this.damage;

}

private void move()

{

move(-speed);

}

private void exitWorld()

{

if ( this.active && getX()<=1) {

this.active = false;

getWorld().removeObject(this);

}

}

public void collide()

{

Actor actor;

actor = getOneObjectAtOffset(4,4, AbstractBullets.class);

if (actor != null)

{

AbstractBullets bullets = (AbstractBullets) actor;

this.health = this.health - bullets.getDamage();

die();

bullets.collision();

}

}

public void die() {

java.util.List gameList = getWorld().getObjects(Game.class);

Game game = (Game)gameList.get(0);

if (this.active && this.health < 0 ) {

this.active = false;

World world;

world = getWorld();

world.removeObject(this);

game.increaseScore(this.points);

}

}

}

public class HomingEnemy extends AbstractEnemies

{

public HomingEnemy (){

health = 3;

speed = 5;

damage = 5;

points = 5;

}

}

public class Enemy extends AbstractEnemies

{

public Enemy (){

health = 1;

speed = 4;

damage = 2;

points = 2;

}

}

public class Game extends Actor

{

private AbstractHero currentHero;

private boolean gameInProgress = false;

private double lastSpawnTime = System.currentTimeMillis();

private double spawnRate = 450;//in milliseconds

private int score;

private GreenfootSound music = new GreenfootSound("Bizet.wav");

private GreenfootImage img;

private Random randomGenerator;

private Random randomYGenerator;

public void updateScore() {

if (this.gameInProgress) {

this.img = new GreenfootImage (200, 20);

this.img.drawString ("Life: " + this.currentHero.getHealth() + " Score: " + this.score ,2,20);

setImage(this.img);

}

}

public void increaseScore(int points) {

this.score = this.score + points;

}

/\*\*

\* Act - do whatever the Game wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

public void act()

{

start();

updateScore();

spawnEnemies();

}

private void start() {

if("s".equals(Greenfoot.getKey()) && !this.gameInProgress)

{

spawnHero();

spawnEnemies();

this.gameInProgress = true;

this.randomGenerator = new Random();

this.randomYGenerator = new Random();

music.setVolume(80);

music.play();

this.score = 0;

getWorld().setBackground("background0172.jpg");

}

}

public void stop() {

this.gameInProgress = false;

this.music.stop();

getWorld().setBackground("desert.png");

this.img = new GreenfootImage (1000,600);

this.img.drawString ("GAME OVER! ",500,300);

this.img.drawString ("Try again! ",500,320);

setImage(this.img);

}

private void spawnHero()

{

this.currentHero = new Zoom();

getWorld().addObject(this.currentHero, 100, 250);

}

private void spawnEnemies() {

double currentSpawnTime = System.currentTimeMillis();

double timeElapsed = currentSpawnTime - this.lastSpawnTime;

if ((this.gameInProgress) && (timeElapsed >= this.spawnRate)) {

int randomEnemyType = 1 + this.randomGenerator.nextInt(100 - 1 + 1);

int randomY = 10 + this.randomGenerator.nextInt(590 - 10 + 1);

if (randomEnemyType > 10) {

getWorld().addObject(new Enemy(), 1000, randomY);

this.lastSpawnTime = System.currentTimeMillis();

}

else {

getWorld().addObject(new HomingEnemy(), 1000, randomY);

this.lastSpawnTime = System.currentTimeMillis();

}

}

}

}

public abstract class AbstractWeapon extends Actor implements Weapon

{

private double lastShotTime = System.currentTimeMillis();

private double firingRate;//in milliseconds

private Ammo ammo;

private int ammoSpeed;

public AbstractWeapon(double pFiringRate, Ammo pAmmo, int pSpeed) {

this.firingRate = pFiringRate;

this.ammo = pAmmo;

this.ammoSpeed = pSpeed;

}

public void shoot(int pX, int pY) {

double currentShotTime = System.currentTimeMillis();

double timeElapsed = currentShotTime - this.lastShotTime;

if (timeElapsed >= this.firingRate) {

try {

String className = ammo.getClass().getSimpleName();

Actor ammoActor = (Actor) Class.forName(className).getConstructors()[0].newInstance();;

getWorld().addObject(ammoActor, pX, pY);

this.lastShotTime = System.currentTimeMillis();

GreenfootSound music = new GreenfootSound("Fireball.wav");

music.setVolume(90);

music.play();

} catch (Exception e) {

}

}

getImage().setTransparency(0);

}

}

public class MachineGun extends AbstractWeapon

{

/\*\*

\* Act - do whatever the MachineGun wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

public void act()

{

// Add your action code here.

}

//AbstractWeapon(double pFiringRate, Ammo pAmmo, int pSpeed) {

public MachineGun() {

super (350, new MagicBullet(), 10);

}

}

public interface Ammo

{

public void move();

}

public interface Champion

{

public void move();

public void shoot();

public void collide();

public void die();

public void spawn();

public void boast();

public void runaway();

}

public interface Weapon

{

public void shoot(int pX, int pY) ;

}

# 