<https://github.com/es9dpw/Ethan-Smith-CO452-PR1-Crab-Game>

CrabWorld.java

import greenfoot.\*;

/\*\*

\* This is the CrabWorld Class

\*/

public class CrabWorld extends World

{

private Crab crab;

private HorizontalLobster hLobster;

private VerticalLobster vLobster;

private Worm worms;

private int wormSize;

public Counter scoreDisplay;

public int score;

public int lives;

public int wormsNUM;

/\*\*

\* Sets up the CrabWorld

\*/

public CrabWorld()

{

super(800, 600, 1);

crab = new Crab();

addObject(crab, 400, 300);

hLobster = new HorizontalLobster();

addObject(hLobster, 0, 400);

wormSize = 30;

createWorm();

wormsNUM = 1;

setupScore();

lives = 3;

showText("Lives: " + lives, 750, 10);

showText("Control the crab with the arrowkeys to avoid", 400, 10);

showText("the lobsters and eat the worms to gain score", 400, 30);

showText("Reach 5000 score to win!", 400, 50);

}

/\*\*

\* This method creates & adds worms in random postions and

\* of different amounts depending on how high the score is

\*/

public void createWorm()

{

if (score < 1000){

worms = new Worm();

addObject(worms, Greenfoot.getRandomNumber(getWidth()), Greenfoot.getRandomNumber(getHeight()));

}

if (score >= 1000 && score < 2600){

wormsNUM--;

if (wormsNUM <= 0){

worms = new Worm();

addObject(worms, Greenfoot.getRandomNumber(getWidth()), Greenfoot.getRandomNumber(getHeight()));

worms = new Worm();

addObject(worms, Greenfoot.getRandomNumber(getWidth()), Greenfoot.getRandomNumber(getHeight()));

wormsNUM = 2;

}

}

if (score >= 2600){

wormsNUM--;

if (wormsNUM <= 0){

worms = new Worm();

addObject(worms, Greenfoot.getRandomNumber(getWidth()), Greenfoot.getRandomNumber(getHeight()));

worms = new Worm();

addObject(worms, Greenfoot.getRandomNumber(getWidth()), Greenfoot.getRandomNumber(getHeight()));

worms = new Worm();

addObject(worms, Greenfoot.getRandomNumber(getWidth()), Greenfoot.getRandomNumber(getHeight()));

wormsNUM = 3;

}

}

}

/\*\*

\* This method adds 100 to the score each time the crab eats a worm.

\* It also checks the score to remove the beginning text and to add more

\* lobsters as well as end the game when the player wins

\*/

public void score()

{

score = score + 100;

if (score == 100){

showText(null, 400, 10);

showText(null, 400, 30);

showText(null, 400, 50);

}

if (score == 1000){

vLobster = new VerticalLobster();

addObject(vLobster, 200, 0);

}

if (score == 2000){

vLobster = new VerticalLobster();

addObject(vLobster, 600, 0);

}

if (score == 3000){

hLobster = new HorizontalLobster();

addObject(hLobster, 0, 200);

}

if (score == 4000){

vLobster = new VerticalLobster();

addObject(vLobster, 400, 0);

}

}

/\*\*

\* This method creates the score display

\*/

public void setupScore()

{

score = 0;

scoreDisplay = new Counter("Score: ");

addObject (scoreDisplay, 60, 30);

score = 0;

}

/\*\*

\* This method takes lives away when the crab collides with a lobster

\* It then updates the lives count and if the lives are 0 it ends the game

\* as the player has lost

\*/

public void minusLife()

{

lives--;

showText("Lives: " + lives, 750, 10);

if (lives <= 0){

endGame();

}

}

/\*\*

\* This method displays text telling the player they lost and ends the

\* game

\*/

public void endGame()

{

showText("Game Over: You have Lost!", 400, 300);

Greenfoot.stop();

}

/\*\*

\* This method displays text telling the player they won and ends the game

\*/

public void winGame()

{

showText("Congratulations, You Won!", 400, 300);

Greenfoot.stop();

}

}

Counter.java

import greenfoot.\*;

/\*\*

\* A Counter class that allows you to display a numerical value on screen.

\*

\* The Counter is an actor, so you will need to create it, and then add it to

\* the world in Greenfoot. If you keep a reference to the Counter then you

\* can adjust its value. Here's an example of a world class that

\* displays a counter with the number of act cycles that have occurred:

\*

\* <pre>

\* class CountingWorld

\* {

\* private Counter actCounter;

\*

\* public CountingWorld()

\* {

\* super(600, 400, 1);

\* actCounter = new Counter("Act Cycles: ");

\* addObject(actCounter, 100, 100);

\* }

\*

\* public void act()

\* {

\* actCounter.setValue(actCounter.getValue() + 1);

\* }

\* }

\* </pre>

\*

\* @author Neil Brown and Michael Kölling

\* @version 1.0

\*/

public class Counter extends Actor

{

private static final Color transparent = new Color(0,0,0,0);

private GreenfootImage background;

private int value;

private int target;

private String prefix;

public Counter()

{

this(new String());

}

/\*\*

\* Create a new counter, initialised to 0.

\*/

public Counter(String prefix)

{

background = getImage();

value = 0;

target = 0;

this.prefix = prefix;

updateImage();

}

/\*\*

\* sets the target variable to the score variable and then runs addScore()

\*/

public void act()

{

target = ((CrabWorld) getWorld()).score;

addScore();

}

/\*\*

\* Animates the display to count up (or down) to the current target value.

\*/

public void addScore()

{

if (target >= 5000){

value = 5000;

updateImage();

((CrabWorld) getWorld()).winGame();

}

else if (value < target) {

value++;

updateImage();

}

else if (value > target) {

value--;

updateImage();

}

}

/\*\*

\* Updates the image on screen to show the current value.

\*/

private void updateImage()

{

GreenfootImage image = new GreenfootImage(background);

GreenfootImage text = new GreenfootImage(prefix + value, 22, Color.BLACK, transparent);

if (text.getWidth() > image.getWidth() - 20)

{

image.scale(text.getWidth() + 20, image.getHeight());

}

image.drawImage(text, (image.getWidth()-text.getWidth())/2,

(image.getHeight()-text.getHeight())/2);

setImage(image);

}

}

Crab.java

import greenfoot.\*;

/\*\*

\* This is the Crab Class

\*/

public class Crab extends Actor

{

protected int width;

protected int height;

protected int speed = 3;

protected int turnAngle = 4;

protected GreenfootImage image;

private CrabWorld world;

/\*\*

\* Sets up the crab and rotates it to face up

\*/

public Crab()

{

image = getImage();

width = image.getWidth();

height = image.getHeight();

image.scale((int)(width \* 0.8), (int)(height \* 0.8));

setRotation(-90);

}

/\*\*

\* Calls move4Ways() to allow the player to move and then calls

\* hitDetection() to check if the player has it any lobsters or worms

\*/

public void act()

{

move4Ways();

hitDetection();

}

/\*\*

\* Checks if the player has it any lobsters or worms and then updates the

\* lives or removes the worm, updates the score and ads new worms

\*/

public void hitDetection()

{

Actor HorizontalLobster = getOneIntersectingObject(HorizontalLobster.class);

if(HorizontalLobster != null)

{

((CrabWorld) getWorld()).minusLife();

setLocation(500, 300);

}

Actor VerticalLobster = getOneIntersectingObject(VerticalLobster.class);

if(VerticalLobster != null)

{

((CrabWorld) getWorld()).minusLife();

setLocation(500, 300);

}

Actor Worm = getOneIntersectingObject(Worm.class);

if(Worm != null)

{

getWorld().removeObject(Worm);

((CrabWorld) getWorld()).score();

((CrabWorld) getWorld()).createWorm();

}

}

/\*\*

\* Allows the crab to move up and down and turn left and right

\*/

public void move4Ways()

{

if (Greenfoot.isKeyDown("left")){

turn(-3);

}

if (Greenfoot.isKeyDown("right")){

turn(3);

}

if (Greenfoot.isKeyDown("up")){

move(3);

}

if (Greenfoot.isKeyDown("down")){

move(-3);

}

}

}

HorizontalLobster.java

import greenfoot.\*;

import java.util.Random;

import java.util.List;

/\*\*

\* This is the HorizontalLobster Class

\*/

public class HorizontalLobster extends Actor

{

private int positionReset = 0;

/\*\*

\* Moves the lobster and then calls positionReset() to see if it has

\* reached the edge of the screen

\*/

public void act()

{

move(2);

positionReset();

}

/\*\*

\* Every time the lobster moves positionReset increments and once it

\* reaches 400 (its X coordinate is 800 meaning its at the edge of the

\* world) its X coordinate is set back to 0

\*/

public void positionReset()

{

positionReset++;

if (positionReset >= 400){

setLocation(0, getY());

positionReset = 0;

}

}

}

VerticalLobster.java

import greenfoot.\*;

import java.util.Random;

import java.util.List;

/\*\*

\* This is the VerticalLobster Class

\*/

public class VerticalLobster extends Actor

{

private int positionReset = 0;

/\*\*

\* Sets the rotation of the lobster downwards so it travels vertically

\* rather than horizontally

\*/

public VerticalLobster()

{

setRotation(90);

}

/\*\*

\* Moves the lobster and then calls positionReset() to see if it has

\* reached the edge of the screen

\*/

public void act()

{

move(2);

positionReset();

}

/\*\*

\* Every time the lobster moves positionReset increments and once it

\* reaches 300 (its Y coordinate is 600 meaning its at the edge of the

\* world) its Y coordinate is set back to 0

\*/

public void positionReset()

{

positionReset++;

if (positionReset >= 300){

setLocation(getX(), 0);

positionReset = 0;

}

}

}

Worm.java

import greenfoot.\*;

/\*\*

\* This is the Worm Class

\*/

public class Worm extends Actor

{

/\*\*

\* The worm does nothing and is simply placed in the world for the crab to

\* eat. All collision detection is handled in the crab so nothing is needed

\* here

\*/

public void act()

{

}

}