ספר מלווה

**פרויקט גמר**

למילוי חלקי של הדרישות לקבלת תואר

**הנדסאי תוכנה**

הנושא: משחק שובר לבנים

העבודה בוצעה בהנחיית : מאיר שבירו

סביבת פיתוח: Java Eclipse,Spring

מכללת אורט רחובות

**תוכן עניינים:**

|  |  |
| --- | --- |
| **הקדמה אישית** | **3** |
| **מבוא** | **4** |
| הצעת נושא לפרויקט | 4 |
| הצגת העבודה | 9 |
| **פרק ראשון: אודות משחק הדמקה** | **10** |
| מטרת וכללי המשחק | 11 |
| **פרק שני: מושגים כלליים** | **14** |
| **פרק שלישי: מדריך למשתמש** | **17** |
| משחק יחיד | 18 |
| **פרק רביעי: מדריך למפתח** | **23** |
| אודות החבילות והמחלקות | 23 |
| פירוט סוגי בקשות ותגובות ברשת | 27 |
| אודות משתנים ופונקציות עיקריים | 28 |
| **נספח- קוד מקור** | **39** |

**הקדמה אישית**

מדוע בחרנו בנושא שובר לבנים?

בחרנו בנושא זה כי אנו אוהבים את המשחק והנושא עניין אותנו מאחר ואנו נוהגים לשחק עם משפחתנו וחברינו במשחק.

לכן, שמחתנו על ההזדמנות ללמוד ולהעשיר את הידע בנושאים אלו ולקבל לבסוף משחק שאנו אוהבים ושכתבנו בעצמנו.

ניתן לציין כי בתחילת העבודה ובמהלכה היו קשיים אך בעזרת חלוקת המשימות בינינו וחשיבה משותפת הצלחנו לפתח את המשחק וככל שהתקדמנו בפיתוחו כך נהנינו יותר מהפרויקט ולבסוף קיבלנו משחק שכתבנו בעצמנו וכיף לשחק בו יחד.

**משרד העבודה והרווחה המכון הממשלתי להכשרה טכנולוגית**

**ה צ ע ה ל פ ר ו י ק ט ג מ ר**

מכללת אורט רחובות כתובת: טשרניחובסקי 1 רחובות

מסלול : הנדסאי תוכנה

**שם הפרויקט**

**משחק "שובר לבנים"**

**מטרת הפרויקט**

הפרויקט ישלב עבודה ב-:

1. משחק שובר לבנים בין 1 משתמש ללא רש על אותו מחשב.

**תיאור קצר ונתונים טכניים**

* אלגוריתם ליצירת לוח שובר לבנים.
* אלגוריתם לבדיקת מהלכים חוקיים בהתאם לסוג חייל המשחק ולתור המשתמש.
* עדכון בזמן אמת של מסכי המשתמשים - לוח המשחק , הפאנל הצדדי.
* טיפול בפעולות אותם מבצעים המשתמשים.
* זיהוי מצב בו הסתיים משחק.

**תוכן העניינים**

###### **ציוד חומרה ותוכנה הנדרשים לביצוע הפרויקט**

* 1. מהדר JAVA של Eclipse.
  2. JVM – לשֵם הרצת מכונה וירטואלית (Byte Code) של ג'אווה.
  3. JDK מותקן – גם במחשבים השונים במשחק רשת.

###### **סקירה כללית של מצב קיים**

קיימים לא מעט משחקים שניתן למצוא ברשת – גם בחינם.

הבעיה טמונה בכך שיש פרסומות וכֵן יש לבצע רישום.

###### **בעיות אפשריות**

* המערכת לא תוכל לאפשר פעולה כפולה של אותו משתמש.
* המערכת לא תאפשר לבצע פעולה לא חוקית של משתמש במשחק.

1. פתרונות אפשריים

* המערכת תדע לזהות מצב שבו לוח המשחק מלא (ואין ניצחון של אחד השחקנים).
* הAI של המחשב יקבל החלטה תוך מספר צעדים (סדר גודל) קבוע וסופי.

1. חלוקה לתוכניות

* מודול של הבינה המלאכותית של המחשב.
* מודול לבדיקת ניצחון

1. זרימת האינפורמציה

בדיקת מהלך חוקי ⬄ שמירת כל מהלך אצל כל שחקן

אמצעי אחסון

הפרויקט מיושם על החומרה בשטח ומוגש צרוב על דיסק בצמוד לספר הפרויקט.

1. גבולות המערכת

המערכת לא תוכל לאפשר פעולה כפולה של אותו משתמש.

המערכת לא תאפשר התחלת שימוש במשחק ללא רישום שם משתמש.

לאחר חיבור לרשת – אם לא התבצע מהלך תוך פרק זמן "ארוך" – המערכת תתנתק

1. מספר השעות המוקדש לפרויקט

600 שעות

1. מה יקנה הפרויקט לסטודנט

* ידע בכל התוכנות ואמצעי הקלט.
* בניית בינה מלאכותית של המחשב וטיפול במהלכים העלולים לזרוק חריג (נניח יציאה מגבולות המשחק).
* עבודה עם try ו- catch בכל מה שקשור לחיבור לרשת , עבודה עם מסד נתונים, זיהוי חיבור/ניתוק של השחקן "השני".  
  תכנות ברמות שונות תחת JAVA הן ברמת מחשב בודד והן עבודה ברשת.

1. שלבים בפיתוח הפרויקט

שלב א' – לימוד החומר התיאורטי.

שלב ב' – חלוקת הפיתוח לשלבים אטומים.

שלב ג' – תכנות בפועל של כל מודול.

שלב ד' – אינטגרציה של המערכת (מרוחקת וקרובה).

שלב ה' – הרצת המערכת ובדיקת תקלות.

שלב ו' – כתיבת ספר הפרויקט.

1. חלקו של כל סטודנט

לא רלוונטי

1. ספרות וביבליוגרפיה

החומר הנלמד בכיתה:

<http://stackoverflow.com>

הצגת העבודה:

**הפרק הראשון** יציג סקירת ספרות אודות משחק שובר לבנים ויפרט על כללי המשחק.

**הפרק השני** עוסק במושגים כלליים בהם היה שימוש במהלך הפרויקט והסברתם.

**הפרק השלישי** יציג את המדריך למשתמש בו יינתן הסבר מפורט על הפעלת התוכנה ותכונותיה למשתמש.

**הפרק הרביעי** יציג את המדריך למפתח בו יינתן פירוט על קוד התוכנית:

בפרק זה יוצגו החבילות והמחלקות ותפקידן וכן תיאור האלגוריתמים, הפונקציות, המחלקות והמשתנים העיקריים בתוכנית.

**הפרק החמישי** יוצג כנספח ויכיל את קוד המקור של התוכנית.

**פרק ראשון:**

רקע[

## תיאור המשחק

השחקן שולט במחבט קטן בתחתית המסך אשר באמצעותו ניתן להקפיץ כדור לעבר הלבנים המסודרות בחלקו העליון של המסך. כאשר הכדור פוגע באחת הלבנים הוא גורם ללבנה בה פגע להיעלם. כאשר כל הלבנים נעלמו, השחקן עובר לשלב הבא, בו מופיעות לבנים חדשות. ישנם שלבים בהם יש וריאציות אחרות (כגון לבנים בהן יש לפגוע מספר פעמים עד שהן נעלמות, ספינות חלל מעופפות, וכו'). כמו כן, מאחורי חלק מהלבנים מסתתרות הפתעות שמאפשרות בין היתר להגדיל את המחבט, להכפיל את מספר הכדורים, לצייד את השחקן בתותח הלייזר או לעבור ישירות לשלב הבא. לאחר 30 שלבים השחקן צריך להילחם בבוס של המשחק.

"

מטרת המשחק

מטרת המשחק היא לפגוע בכל האבנים של לוח המשחק.

אופן המשחק (שובר לבנים(

על השחקן עם המחבט המונח בתחתית המסך לפגוע הכל הלבנים על מנת לנצח ולסיים את המשחק בזמן המהיר ביותר

**אבנים**

**תנועה:**

השחקן מזיז את המחבט מצד ימין לשמאל על מנת לחבוט בכדור.

**פגיעה:**

**השחק פוגע בלבנה הלבנה נעלמת.**

**סיום המשחק**

המשחק יכול להסתיים בניצחון . ניצחון מושג אם מתקיים:

סיום של כל הלוח משחק כלומר פגיעה בכל הלבנים.

**פרק שני**

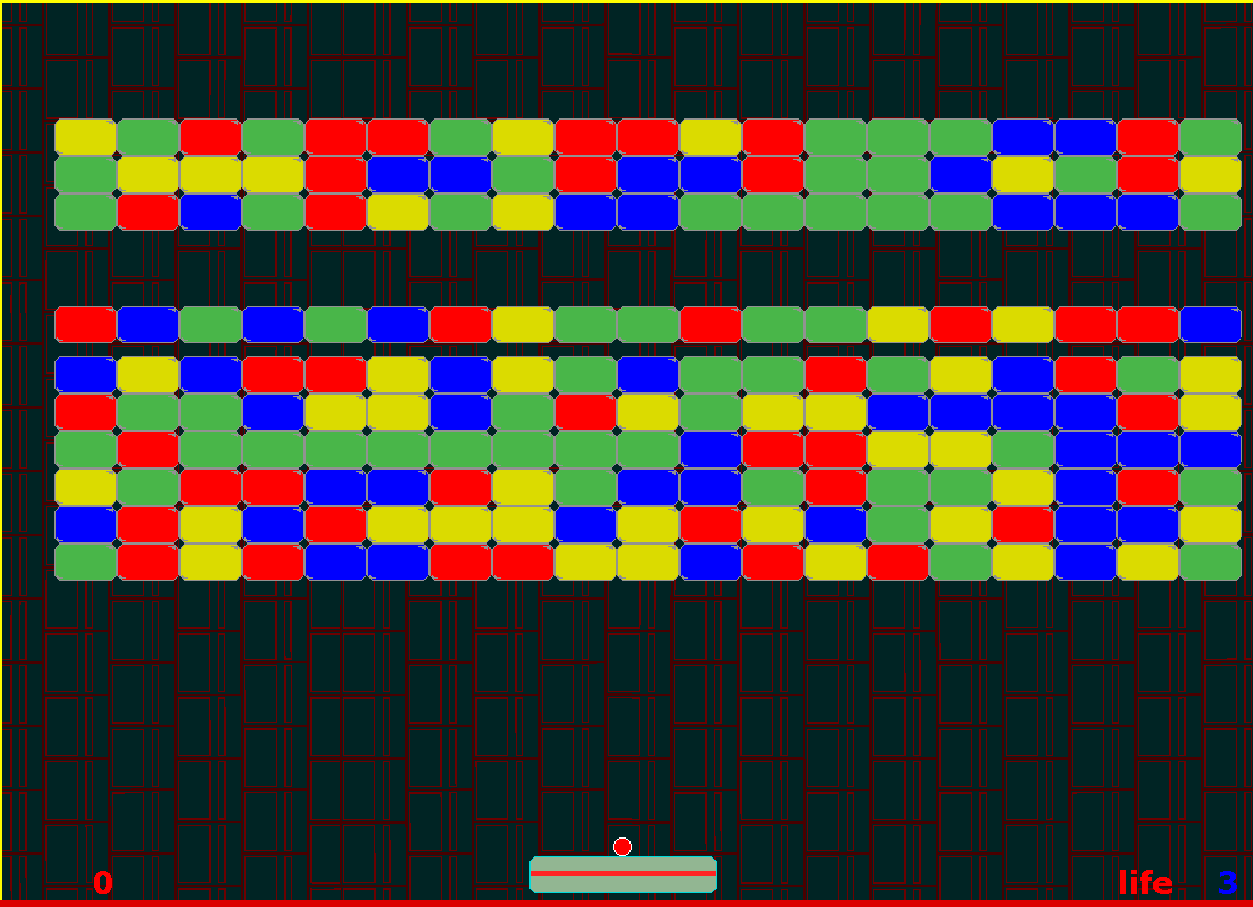
**2. Thread:** הוא תהליך לביצוע המהווה חלק מיישום, יישום יכול ליצור מספר Threads שירוצו במקביל, דבר זה נקרא multi-thread.

הרצת ה Threads נעשית כביכול במקביל ע"י מערכת ההפעלה באופן אוטומט שכן בכל רגע נתון נמצא בפועל תהליכון(Thread) אחד במצב ריצה והשאר ממתינים להמשך ריצתם (אלא אם מדובר במערכת הפעלה מרובת מעבדים).

**פרק שלישי – מדריך למשתמש**



**מסך פתיחה**



**מצב משחק**



**ניקוד**



**מספר פסילות שנשאר לשחקן**

**פרק ארבע - מדריך למפתח**

הפרויקט מורכב מחמש חבילות (package) אשר מכילות את המחלקות של הפרויקט.

**MainPackage:** החבילה העיקרית של הפרויקט, מכילה את רוב מחלקות הפרויקט ואת החלונות של הפרויקט.

* **MainClass.java**

אחראית להרצת הפרויקט ומכילה את הפונקציה main.

* **MainFrame.java(extends JFrame)**

החלון הראשי של הפרויקט

* **MainPanel.java(extends JPanel)**

הפאנל הראשי של הפרויקט, ממנו ניתן לבחור את סוג המשחק ולהכנס אליו.

* **ArkaBackground – חלון פתיחה**
* **ArkanoidImage – הצבת התמונות של הלבנים.**
* **ArkanoidSound – סאונד של המשחק.**
* **SimpleArkanoid – לוגיקה של המשחק**

**נספח- קוד הפרויקט.**

**import java.awt.Canvas;**

**import java.awt.Color;**

**import java.awt.Graphics;**

**import java.awt.Graphics2D;**

**import javax.imageio.ImageIO;**

**import java.io.File;**

**import java.awt.image.BufferedImage;**

**class ArkaBackground {**

**private BufferedImage image;**

**private int x;**

**private int y;**

**public ArkaBackground() {**

**this(0,0);**

**}**

**public ArkaBackground(int x, int y) {**

**this.x = x;**

**this.y = y;**

**try {**

**image = ImageIO.read(new File("sfondo.png"));**

**}**

**catch (Exception e) { System.out.println(e); }**

**}**

**public void draw(Graphics window) {**

**window.drawImage(image, getX(), getY(), image.getWidth(), image.getHeight(), null);**

**this.x -= 1;**

**if (this.x <= -1 \* image.getWidth()) {**

**this.x = this.x + image.getWidth() \* 2;//**

**}**

**}**

**public void setY (int y) {**

**this.y = y;**

**}**

**public void setX(int x) {**

**this.x = x;**

**}**

**public int getX() {**

**return this.x;**

**}**

**public int getY() {**

**return this.y;**

**}**

**public int getImageHeight () {**

**return image.getHeight();**

**}**

**public int getImageWidth() {**

**return image.getWidth();**

**}**

/\*

\* Ver 0.1.0.0. - PreAlpha

\*

\* SimpleArkanoid

\* Simple game made for exercise with java swing.

\* By Deovin Italy, Milan ( Deox@libero.it )

\* 12/1/2018

\* \*/

import java.util.Random;

public class ArkanoidEnemy extends ArkanoidImage {

private final int INITIAL\_X = 1000;

Random rand;

int MattoneColor;

int mattoneenergy;

public ArkanoidEnemy(int x, int y) {

super(x, y);

initmattoni();

}

private void initmattoni() {

rand = new Random ();

MattoneColor = (rand.nextInt (4) +1);

if (MattoneColor ==1) {

loadImage("mblu.png");

mattoneenergy =1;

}

if (MattoneColor ==2) {

loadImage("mgiallo.png");

mattoneenergy =3;

}

if (MattoneColor ==3) {

loadImage("mrosso.png");

mattoneenergy =5;

}

if (MattoneColor ==4) {

loadImage("mverde.png");

mattoneenergy =7;

}

getImageDimensions();

}

public void setMattoneEnergy () {

mattoneenergy -=1;

if (mattoneenergy <0 ) mattoneenergy =0;

}

public int RitornoEnergy () {

return mattoneenergy ;

}

import java.io.\*;

import javax.sound.sampled.\*;

public class ArkanoidSound {

AudioFormat adFormat = getAudioFormat();

Clip clip;

private AudioFormat getAudioFormat() {

float sampleRate = 42100.0F;

int sampleSizeInbits = 16;

int channels = 1;

boolean signed = true;

float frameRate = 2;

boolean bigEndian = false;

return new AudioFormat (sampleRate, sampleSizeInbits, channels, signed, bigEndian);

}

void chargesound(File file) {

try {

AudioInputStream in = AudioSystem.*getAudioInputStream*(file);

clip = AudioSystem.*getClip*();

clip.open(in);

}catch (IOException e) {

e.printStackTrace();

}catch (LineUnavailableException e) {

e.printStackTrace();

}catch (UnsupportedAudioFileException e) {

e.printStackTrace();

//Thread.sleep(1000);

}

}

void startsound () {

if (clip.isRunning())

clip.stop();

clip.setFramePosition(0);

clip.loop(Clip.***LOOP\_CONTINUOUSLY***);

}

void ONCEstartsound () {

clip.setFramePosition(0);

clip.start();

}

void stopsound () {

if (clip.isRunning())

clip.stop();

clip.setFramePosition(0);

}

}

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

import java.awt.Rectangle;

import java.util.Random;

import java.io.File;

import java.awt.image.BufferedImage;

import javax.sound.sampled.\*; // for Sound

public class SimpleArkanoid extends JFrame {

public SimpleArkanoid() {

initUI();

}

private void initUI() {

add(new GameArkanoid());

setTitle("SimpleArkanoid");

setSize(1024, 768);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

setResizable(false);

}

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

SimpleArkanoid ex = new SimpleArkanoid();

ex.setVisible(true);

}

});

}

}

class GameArkanoid extends JPanel implements ActionListener ,KeyListener {

ArkanoidImage cursoreA = new ArkanoidImage (430,690);

ArkanoidImage cursoreB = new ArkanoidImage (480,690);

ArkanoidImage cursoreC = new ArkanoidImage (530,690);

ArkanoidImage murosx = new ArkanoidImage (1,1);

ArkanoidImage murodx = new ArkanoidImage (1017,1);

ArkanoidImage murosu = new ArkanoidImage (1,1);

ArkanoidImage murogiu = new ArkanoidImage (1,725);

ArkanoidImage presentax = new ArkanoidImage (1,1);

ArkanoidImage palla = new ArkanoidImage (497,675);

ArkanoidImage sfondo = new ArkanoidImage (0,0);

ArkanoidImage gameOverArka = new ArkanoidImage (260,300);

private ArrayList<ArkanoidEnemy> mattoni;

private BufferedImage Arkaback;

public ArkaBackground ArkaA = new ArkaBackground();

public ArkaBackground ArkaB = new ArkaBackground (ArkaA.getImageHeight(), 0);

Random rand;

String life ="life";

int numberlife= 3;

int punti =0;

boolean pallaON = false;

boolean pallasu = false;

int pallaangolo = 1;

int quantinemici =1;

int fasegame =3;

int TheMovimentDX =0;

int TheMovimentSX =0;

private final int[][] pos = {

{50,100}, {100,100}, {150,100}, {200,100} ,{250,100}, {300,100}, {350,100},{400,100}, {450,100}, {500,100} ,{550,100}, {600,100}, {650,100},{700,100}, {750,100}, {800,100} ,{850,100}, {900,100}, {950,100}

,{50,130}, {100,130}, {150,130}, {200,130} ,{250,130}, {300,130}, {350,130},{400,130}, {450,130}, {500,130} ,{550,130}, {600,130}, {650,130},{700,130}, {750,130}, {800,130} ,{850,130}, {900,130}, {950,130}

,{50,160}, {100,160}, {150,160}, {200,160} ,{250,160}, {300,160}, {350,160},{400,160}, {450,160}, {500,160} ,{550,160}, {600,160}, {650,160},{700,160}, {750,160}, {800,160} ,{850,160}, {900,160}, {950,160}

,{50,250}, {100,250}, {150,250}, {200,250} ,{250,250}, {300,250}, {350,250},{400,250}, {450,250}, {500,250} ,{550,250}, {600,250}, {650,250},{700,250}, {750,250}, {800,250} ,{850,250}, {900,250}, {950,250}

,{50,290}, {100,290}, {150,290}, {200,290} ,{250,290}, {300,290}, {350,290},{400,290}, {450,290}, {500,290} ,{550,290}, {600,290}, {650,290},{700,290}, {750,290}, {800,290} ,{850,290}, {900,290}, {950,290}

,{50,320}, {100,320}, {150,320}, {200,320} ,{250,320}, {300,320}, {350,320},{400,320}, {450,320}, {500,320} ,{550,320}, {600,320}, {650,320},{700,320}, {750,320}, {800,320} ,{850,320}, {900,320}, {950,320}

,{50,350}, {100,350}, {150,350}, {200,350} ,{250,350}, {300,350}, {350,350},{400,350}, {450,350}, {500,350} ,{550,350}, {600,350}, {650,350},{700,350}, {750,350}, {800,350} ,{850,350}, {900,350}, {950,350}

,{50,380}, {100,380}, {150,380}, {200,380} ,{250,380}, {300,380}, {350,380},{400,380}, {450,380}, {500,380} ,{550,380}, {600,380}, {650,380},{700,380}, {750,380}, {800,380} ,{850,380}, {900,380}, {950,380}

,{50,410}, {100,410}, {150,410}, {200,410} ,{250,410}, {300,410}, {350,410},{400,410}, {450,410}, {500,410} ,{550,410}, {600,410}, {650,410},{700,410}, {750,410}, {800,410} ,{850,410}, {900,410}, {950,410}

,{50,440}, {100,440}, {150,440}, {200,440} ,{250,440}, {300,440}, {350,440},{400,440}, {450,440}, {500,440} ,{550,440}, {600,440}, {650,440},{700,440}, {750,440}, {800,440} ,{850,440}, {900,440}, {950,440}

};

ArkanoidSound Thepop;

File FThepop = new File ("ArkaSound/pop.wav");

private Timer timer;

private final int DELAY = 10;

public GameArkanoid () {

cursoreA.loadImage ("Acursore.png");

cursoreB.loadImage ("Bcursore.png");

cursoreC.loadImage ("Ccursore.png");

murodx.loadImage("murolaterale.png");

murosx.loadImage("murolaterale.png");

murosu.loadImage("muroalto.png");

murogiu.loadImage("murobasso.png");

presentax.loadImage ("presentazione.png");

palla.loadImage ("palla.png");

gameOverArka.loadImage ("gameover.png");

Thepop = new ArkanoidSound();

initSurface() ;

}

private void initSurface() {

Thepop.chargesound (FThepop);

addKeyListener (this);

setFocusable (true);

setBackground (Color.BLACK);

timer= new Timer (DELAY, this);

timer.start();

}

public void paintComponent (Graphics g) {

super.paintComponent(g);

if (fasegame == 3) logoiniziale (g);

if (fasegame == 4) preparagame ();

if (fasegame == 5) {

if (TheMovimentDX ==1) movimentocursoreDX();

if (TheMovimentSX ==1) movimentocursoreSX();

if (pallaON == true) intersezioni();

if (pallaON == true) movimentoPalla();

if (pallaON == true) collisioneMattoni();

if (pallaON == true) collisioneBordoBasso();

doDrawing (g);

stampaScore(g);

}

if (fasegame == 6) {

doDrawing (g);

stampaScore(g);

stampaGameOver(g);

regame();

}

Toolkit.getDefaultToolkit().sync();

}

private void doDrawing (Graphics g) {

Graphics2D g2D = (Graphics2D) g;

if (Arkaback == null) Arkaback = (BufferedImage)(createImage(getWidth(), getHeight()));

Graphics buffer = Arkaback.createGraphics();

ArkaA.draw(buffer);

ArkaB.draw(buffer);

g2D.drawImage(Arkaback, null, 0,0);

g2D.drawImage (murosx.getImage (), murosx.getX(), murosx.getY(), this);

g2D.drawImage (murodx.getImage (), murodx.getX(), murodx.getY(), this);

g2D.drawImage (murosu.getImage (), murosu.getX(), murosu.getY(), this);

g2D.drawImage (murogiu.getImage (), murogiu.getX(), murogiu.getY(), this);

g2D.drawImage (palla.getImage (), palla.getX(), palla.getY(), this);

g2D.drawImage (cursoreA.getImage () ,cursoreA.getX(), cursoreA.getY(),this);

g2D.drawImage (cursoreB.getImage (), cursoreB.getX(), cursoreB.getY(),this);

g2D.drawImage (cursoreC.getImage () ,cursoreC.getX(), cursoreC.getY(),this);

for (ArkanoidEnemy a : mattoni) {

if (a.isVisible()) {

g.drawImage(a.getImage(), a.getX(), a.getY(), this);

}

}

}

private void logoiniziale (Graphics g) {

Graphics2D g2D = (Graphics2D) g ;

g2D.drawImage (presentax.getImage () ,1,1,this);

}

@Override

public void actionPerformed(ActionEvent e) {

repaint();

}

public void preparagame () {

mattoni = new ArrayList<>();

for (int[] p : pos) {

mattoni.add(new ArkanoidEnemy(p[0], p[1]));

}

fasegame =5;

}

public void movimentocursoreDX(){

if (cursoreC.getX() >950) TheMovimentDX =0;

if (pallaON == false && cursoreC.getX() >585) TheMovimentDX =0;

if (TheMovimentDX ==1 ) {

cursoreA.moveDX();

cursoreB.moveDX();

cursoreC.moveDX();

}

}

public void movimentocursoreSX(){

if (cursoreA.getX() <30) TheMovimentSX =0;

if (pallaON == false && cursoreA.getX() <380) TheMovimentSX =0;

if (TheMovimentSX ==1) {

cursoreA.moveSX();

cursoreB.moveSX();

cursoreC.moveSX();

}

}

public void intersezioni () {

Rectangle r1 = palla.getBounds();

Rectangle r2 = cursoreA.getBounds();

Rectangle r3 = cursoreB.getBounds();

Rectangle r4 = cursoreC.getBounds();

Rectangle r5 = murosu.getBounds();

Rectangle r6 = murosx.getBounds();

Rectangle r7 = murodx.getBounds();

if ((r1.intersects(r2) && pallasu == false)) {

pallasu = true;

pallaangolo = 0;

}

if ((r1.intersects(r3) && pallasu == false)) {

pallasu = true;

pallaangolo =1;

}

if ((r1.intersects(r4) && pallasu == false)) {

pallasu = true;

pallaangolo =2;

}

if (r1.intersects(r5) && pallasu == true ) {

pallasu = false;

}

if ((r1.intersects(r6) && pallaangolo ==0)) {

pallaangolo =2;

}

if ((r1.intersects(r7) && pallaangolo ==2)) {

pallaangolo =0;

}

}

public void movimentoPalla () {

if (pallasu== true) palla.PmoveSU();

if (pallasu== false) palla.PmoveGIU();

if (pallaangolo == 0 ) palla.PmoveSX();

if (pallaangolo == 2 ) palla.PmoveDX();

}

public void collisioneMattoni() {

for (ArkanoidEnemy Ae : mattoni) {

Rectangle r1 = Ae.getBounds();

Rectangle r2 = palla.getBounds();

if (Ae.isVisible () ){

if (r1.intersects (r2)) {

Ae.setMattoneEnergy ();

punti += 5;

if (pallasu == false) {

pallasu =true;

}

else if (pallasu == true) {

pallasu = false;

}

rand = new Random ();

int casualAngle = (rand.nextInt (3) +1);

if (casualAngle ==1) pallaangolo =0;

if (casualAngle ==2) pallaangolo =1;

if (casualAngle ==3) pallaangolo =2;

if (Ae.RitornoEnergy () ==0) {

Thepop.ONCEstartsound();

Ae.setVisible(false);

punti +=50;

}

}

}

}

}

public void collisioneBordoBasso () {

Rectangle r1 = palla.getBounds();

Rectangle r2 = murogiu.getBounds();

if (r1.intersects(r2)) {

pallaON = false;

pallasu = false;

pallaangolo = 1;

numberlife -=1;

palla.pallanuova();

cursoreA.setcursoreposition(430,690);

cursoreB.setcursoreposition(480,690);

cursoreC.setcursoreposition(530,690);

if (numberlife < 1) fasegame =6;

}

}

public void stampaScore (Graphics g){ // print score

Graphics2D g2d = (Graphics2D) g;

Font font = new Font("Verdana", Font.BOLD, 25);

FontMetrics metr = this.getFontMetrics(font);

g2d.setColor(Color.RED);

g2d.setFont(font);

String finpunt = Integer.toString(punti);

String finlife = Integer.toString(numberlife);

g2d.drawString (finpunt, 80,720);

g2d.drawString (life ,900,720);

g2d.setColor(Color.BLUE);

g2d.drawString (finlife,980,720);

}

private void stampaGameOver (Graphics g) { // print game over

Graphics2D g2D = (Graphics2D) g ;

g2D.drawImage (gameOverArka.getImage () ,gameOverArka.getX(),gameOverArka.getY(),this);

}

public void regame () { // after loss reborn match

// dati della palla

pallaON = false;

pallasu = false;

pallaangolo = 1; // 0 sx - 1 centro - 2 dx

// numero di mattoncini da abbattere

quantinemici =1;

// variabili per la chiamata ai metodi di movimento

TheMovimentDX =0;

TheMovimentSX =0;

}

public void keyPressed (KeyEvent ke) {

int key = ke.getKeyCode();

// eventi della barra spaziatrice // space key

if (key == KeyEvent.VK\_SPACE ){

// passaggio dalla schermata iniziale al gioco

if (fasegame == 3) fasegame = 4;

if (fasegame == 5 && pallaON == false) pallaON = true;

if (fasegame == 6 && pallaON == false) {

fasegame =3;

numberlife= 3;

punti =0;

}

}

// eventi della freccia destra // right key

if ((fasegame ==5) && key == KeyEvent.VK\_RIGHT) {

TheMovimentDX =1;

}

// eventi della freccia sinistra // left key

if ((fasegame ==5) && key == KeyEvent.VK\_LEFT) {

TheMovimentSX =1;

}

}

public void keyReleased (KeyEvent ke) {

int key = ke.getKeyCode();

// eventi della freccia destra

if ((fasegame ==5) && key == KeyEvent.VK\_RIGHT) {

TheMovimentDX =0;

}

// eventi della freccia sinistra

if ((fasegame ==5) && key == KeyEvent.VK\_LEFT) {

TheMovimentSX =0;

}

}

public void keyTyped (KeyEvent ke) {

}

}

}