Krimbat në dritare (ose aplete)

Faton Berisha

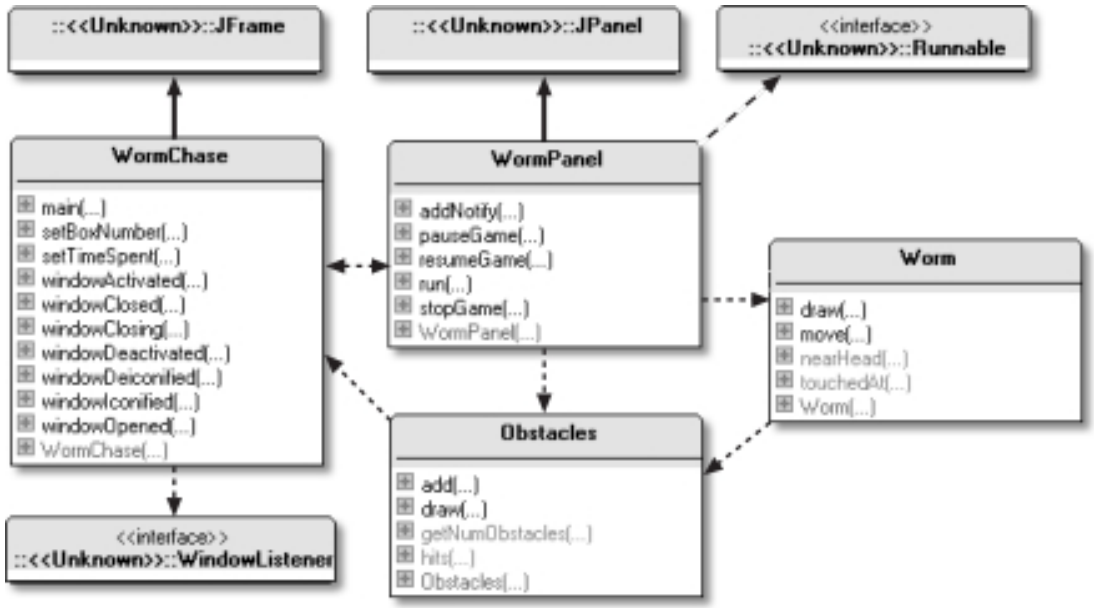
Fakulteti i Shkencave Kompjuterike
Universiteti i Prizrenit

Qëllimet dhe objektivat

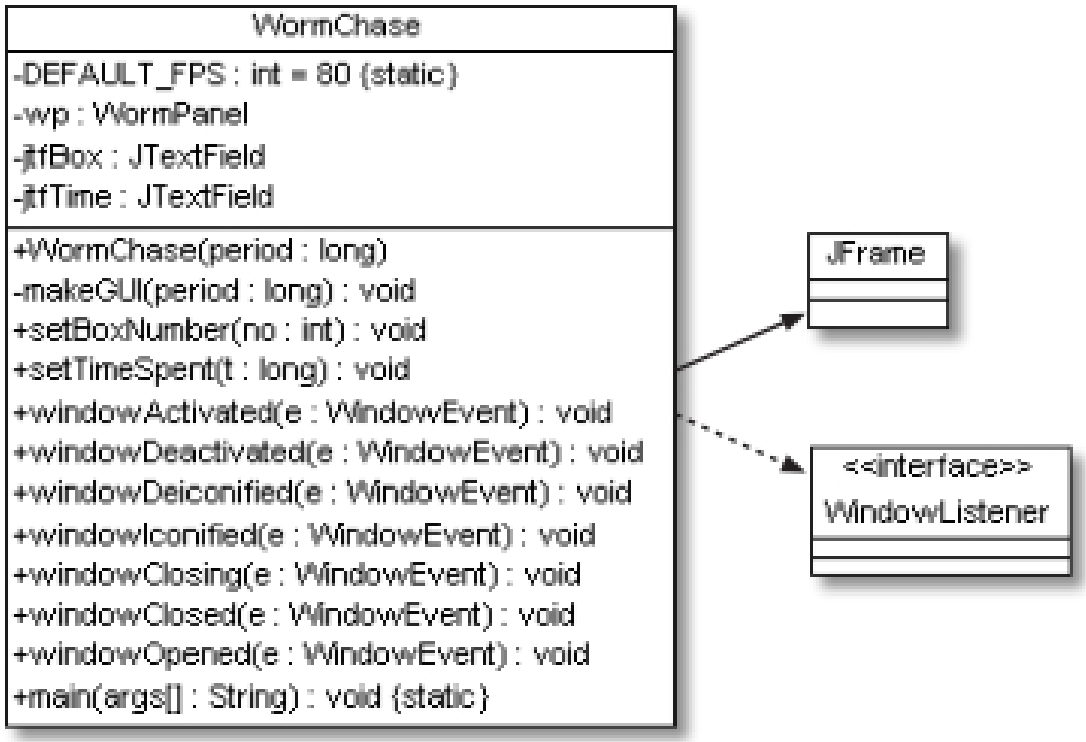
- Implementimi i panelës së fijëzuar si kornizë animimi për nje lojë
- Ndërtimi i një aplikacioni që implementon një lojë
- Vendosja e panelit në një dritare (ose në një aplet)

Përmbajtja

Diagrami i klasave



Detajet e WormChase



Aplikacioni

```
public static void main(String args[]) {
  int fps = DEFAULT_FPS;
  if (args.length != 0)
    fps = Integer.parseInt(args[0]);
  long period = (long) 1000.0/fps;
  System.out.println("fps: " + fps + "; period: " +period+ " ms");
  new WormChase(period);
}
```

Aplikacioni (Vazhdim)

```
public class WormChase extends JFrame implements WindowListener {
  private static int DEFAULT_FPS = 80;

  private WormPanel wp;         // panela e lojës
  private JTextField jtfBox;    // numri i kutive
  private JTextField jtfTime;   // koha e lojës

  public WormChase(long period) {
     super("Ndjekja e krimbit");
    makeGUI(period);

    addWindowListener(this);
    pack();
    setResizable(false);
    setVisible(true);
  }
```

Aplikacioni (Vazhdim)

```
private void makeGUI(long period) {
  Container c = getContentPane();

  wp = new WormPanel(this, period);
  c.add(wp, "Center");

  JPanel ctrls = new JPanel();   // fusha teksti
  ctrls.setLayout(new BoxLayout(ctrls, BoxLayout.X_AXIS));

  jtfBox = new JTextField("Kuti: 0");
  jtfBox.setEditable(false);
  ctrls.add(jtfBox);

  jtfTime = new JTextField("Koha: 0 sek");
  jtfTime.setEditable(false);
  ctrls.add(jtfTime);

  c.add(ctrls, "South");
}
```

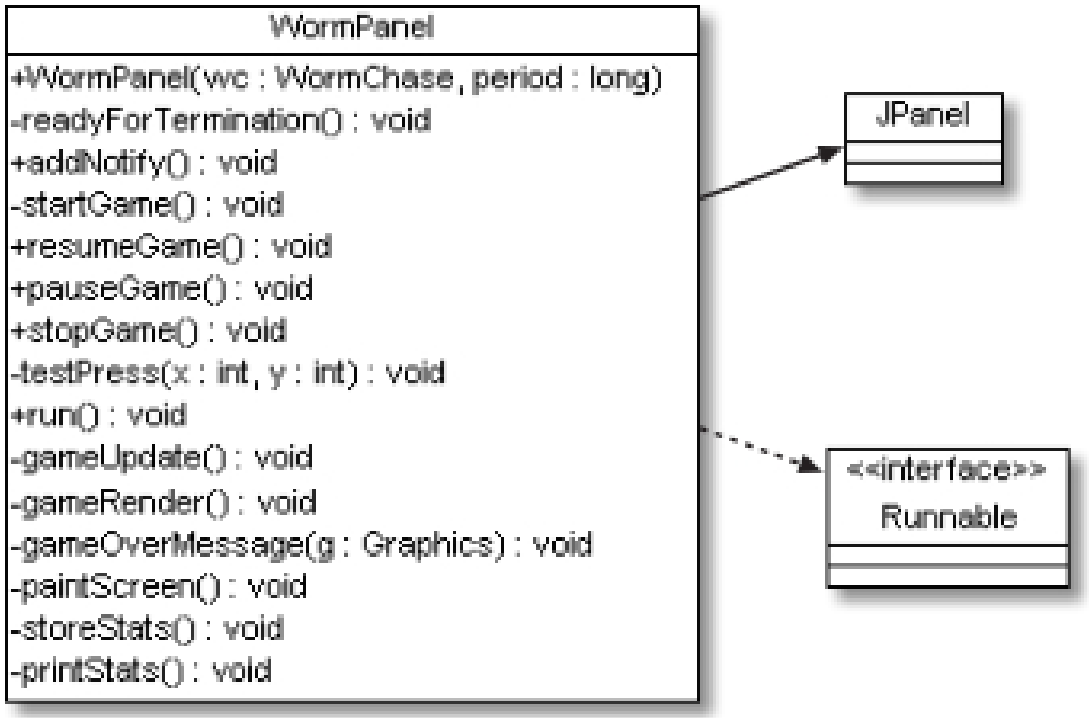
Aplikacioni (Vazhdim)

```
public void setBoxNumber(int no) {
  jtfBox.setText("Kuti: " + no);
}

public void setTimeSpent(long t) {
  jtfTime.setText("Koha: " + t + " sek");
}
```

Aplikacioni (Vazhdim)

```
public void windowActivated(WindowEvent e) {
  wp.resumeGame();
}
public void windowDeactivated(WindowEvent e) {
  wp.pauseGame();
}
public void windowDeiconified(WindowEvent e) {
  wp.resumeGame();
}
public void windowIconified(WindowEvent e) {
  wp.pauseGame();
}
public void windowClosing(WindowEvent e) {
  wp.stopGame();
}
public void windowClosed(WindowEvent e) { }
public void windowOpened(WindowEvent e) { }
```

Detajet e WormPanel



Panela e lojës (Vazhdim)

```
private Worm fred; // krimbi
private Obstacles obs; // pengesat
private Font font;
private FontMetrics metrics;
private WormChase wcTop;  // dritarja e aplikacionit

public WormPanel(WormChase wc, long period) {
  wcTop = wc;
  this.period = period;

  setBackground(Color.white);
  setPreferredSize(new Dimension(PWIDTH, PHEIGHT));

  setFocusable(true);
  requestFocus(); // merr fokusin, prano ngjarje tastesh
  readyForTermination();

  obs = new Obstacles(wcTop);
  fred = new Worm(PWIDTH, PHEIGHT, obs);
```

Panela e lojës (Vazhdim)

```
  addMouseListener(new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
      testPress(e.getX(), e.getY());
    }
  });

  // fonti i mesazheve
  font = new Font("SansSerif", Font.BOLD, 24);
  metrics = this.getFontMetrics(font);
}
```

Inputi i shfrytëzuesit

```
/* A është (x,y) afër kokës apo të shtohet një pengesë? */
private void testPress(int x, int y) {
  if (!isPaused && !gameOver) {
    if (fred.nearHead(x, y)) {
      gameOver = true;
      score = (40 - timeSpentInGame) + (40 - obs.getNumObstacles());
    } else { // shto një pengesë nëse ka mundësi
      if (!fred.touchedAt(x, y))
        obs.add(x, y);
    }
  }
}
```

Cikli i animimit

```
private long gameStartTime;

public void run() {
  gameStartTime = System.currentTimeMillis();

  long beforeTime, timeDiff, sleepTime;
  beforeTime = System.currentTimeMillis();
  running = true;
  while (running) {
    try {
      if (isPaused) {
        synchronized (this) {
          while (isPaused && running)
            wait();
        }
      }
    } catch (InterruptedException e) { }
```

Cikli i animimit (Vazhdim)

```
    gameUpdate();
    gameRender();
    paintScreen();
    timeDiff = System.currentTimeMillis() - beforeTime;
    sleepTime = period - timeDiff; // koha e mbetur në iterimin
    if (sleepTime <= 0) // update-render zgjati më tepër
      sleepTime = 5; // megjithatë flej pak
    try {
      Thread.sleep(sleepTime); // në ms
    } catch (InterruptedException ex) { }
    beforeTime = System.currentTimeMillis();
    storeStats();
  }
  printStats();
  System.exit(0);
}
```

Grumbullimi i statistikave

```
private int timeSpentInGame = 0; // në sekonda

private long frameCount = 0;
private double averageFPS = 0.0;

private void storeStats() {
  frameCount++;
  long timeNow = System.currentTimeMillis();
  if (!gameOver) {
    // ms --> sec
    timeSpentInGame = (int) ((timeNow - gameStartTime) / 1000);
    wcTop.setTimeSpent(timeSpentInGame);
  }
  averageFPS = frameCount / ((timeNow - gameStartTime) / 1000.0);
}
```

Shtypja e statistikave

```
private void printStats() {
  System.out.println("Korniza: " + frameCount);
  System.out.println("Mesatarisht FPS: " + df.format(averageFPS));
  System.out.println("Koha: " + timeSpentInGame + " secs");
  System.out.println("Kuti: " + obs.getNumObstacles());
}
```

Azhornimi i gjendjes

```
private void gameUpdate() {
  if (!isPaused && !gameOver)
    fred.move();
}
```

Vizatimi i pëlhurës

```
private DecimalFormat df = new DecimalFormat("0.##");

private void gameRender() {
  if (dbImage == null) {
    dbImage = createImage(PWIDTH, PHEIGHT);
    if (dbImage == null) {
      System.out.println("dbImage is null");
      return;
    } else
      dbg = dbImage.getGraphics();
  }

  dbg.setColor(Color.white);
  dbg.fillRect(0, 0, PWIDTH, PHEIGHT);
```

Vizatimi i pëlhurës (Vazhdim)

```
    dbg.setColor(Color.blue);
    dbg.setFont(font);
    // raporto FPS mesatare
    dbg.drawString("mesatarisht FPS: " + df.format(averageFPS), 20, 25);

    dbg.setColor(Color.black);
    // vizato elementet: pengesat dhe krimbin
    obs.draw(dbg);
    fred.draw(dbg);

    if (gameOver)
      gameOverMessage(dbg);
  }
```

Vizatimi i pëlhurës (Vazhdim)

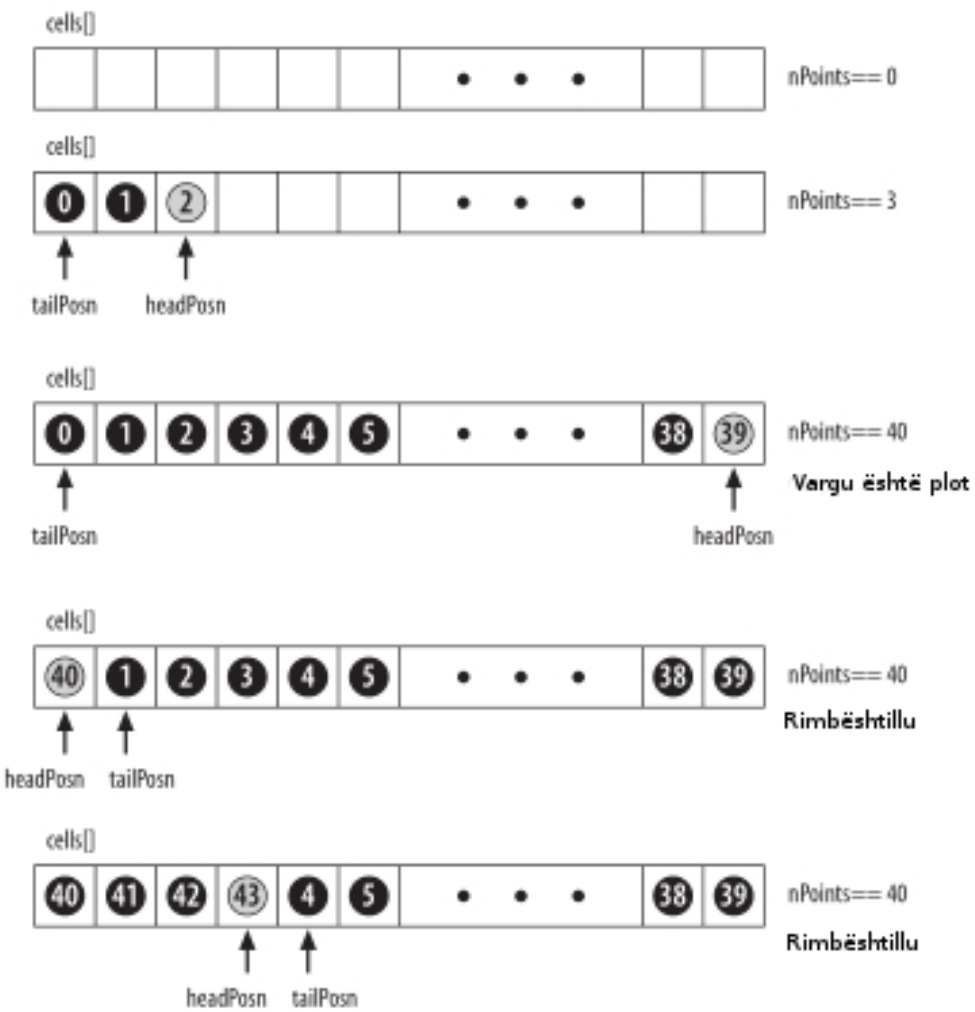
```
private void gameOverMessage(Graphics g) {
  String msg = "Game Over. Rezultati juaj: " + score;
  int x = (PWIDTH - metrics.stringWidth(msg)) / 2;
  int y = (PHEIGHT - metrics.getHeight()) / 2;
  g.setColor(Color.red);
  g.setFont(font);
  g.drawString(msg, x, y);
}
```

Klasa `Worm`:

- Ruan informatën mbi koordinatat e krimbit
  në një bafer cirkular
- Përmban metoda për testim nëse lojtari ka klikuar
  afër kokës ose trupit të krimbit
- Përmban metoda për vizatimin e krimbit
- Rrit krimbin deri në një madhësi maksimale
- Përmban metodë për rregullimin e lëvizjes së krimbit
  që të jetë semi-i rastësishëm ashtu që kryesisht të lëvizë para
- Bën që krimbi të lëvizë përreth pengesave.

- Krimbi rritet duke ruajtur një varg objektesh `Point` në vargun `cells[]`.
- Lëvizja e krimbi në tërë madhësinë bëhet duke krijuar një rreth të ri koke para dhe duke hequr një rreth bishti (nëse ka nevojë).
- Heqja e një rrethi liron një hapësirë në vargun `cells[]` ku mund të ruhet rrethi për kokën e re.

Struktura e të dhënave e krimbit

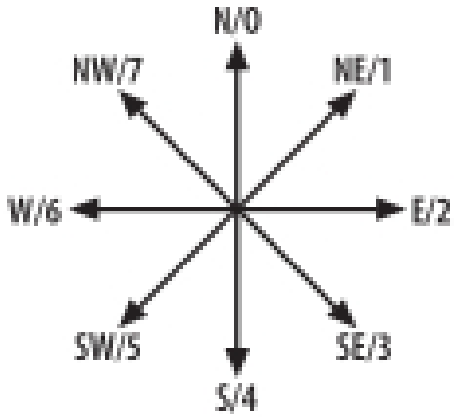


Struktura e të dhënave e krimbit (Vazhdim)

```
public class Worm {
  private static final int DOTSIZE = 12;
  private static final int RADIUS = DOTSIZE/2;
  private static final int MAXPOINTS = 40;

  private Point[] cells;
  private int nPoints;
  private int tailPosn, headPosn;  // bishti dhe koka e baferit

  private int pWidth, pHeight;   // dimensionet e panelit
  private long startTime;        // në ms
  private Obstacles obs;

  // ... anëtarët tjerë
}
```

Drejtim-lëvizjet sipas busollës

- Çdo drejtim lëvizjeje (sipas 8 drejtimeve të busollës) është paraqitur me një numër të plotë në drejtim të akrepave të orës.

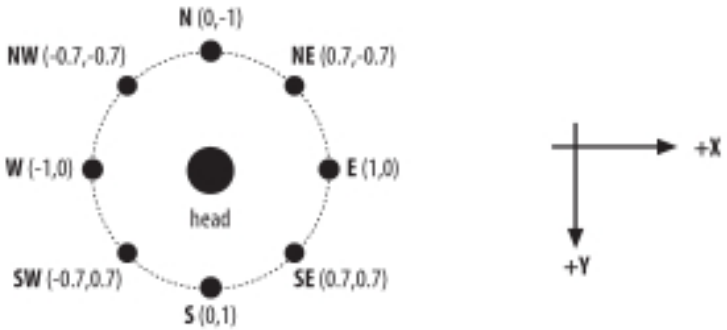


Drejtim-lëvizjet sipas busollës (Vazhdim)

```
// konstantat e drejtimit të busollës
private static final int NUM_DIRS = 8;
private static final int N = 0;  // veri... në drejtim të akrepave
private static final int NE = 1;
private static final int E = 2;
private static final int SE = 3;
private static final int S = 4;
private static final int SW = 5;
private static final int W = 6;
private static final int NW = 7;

private int currCompass;  // ruan drejtim-lëvizjen aktuale
```

Zhvendosjet nga pozita aktuale e kokës

- Kur krijohet kokë e re vendoset në njërin nga drejtim-lëvizjet me zhvendosje 1 njësi.



Drejtim-lëvizjet sipas busollës (Vazhdim)

```
Point2D.Double incrs[];

public Worm(int pW, int pH, Obstacles os) {
  pWidth = pW; pHeight = pH;
  obs = os;
  cells = new Point[MAXPOINTS];  // inicializo baferin
  nPoints = 0;
  headPosn = -1;  tailPosn = -1;

  // rritjet për çdo drejtim
  incrs = new Point2D.Double[NUM_DIRS];
  incrs[N] = new Point2D.Double(0.0, -1.0);
  incrs[NE] = new Point2D.Double(0.7, -0.7);
  incrs[E] = new Point2D.Double(1.0, 0.0);
  incrs[SE] = new Point2D.Double(0.7, 0.7);
  incrs[S] = new Point2D.Double(0.0, 1.0);
  incrs[SW] = new Point2D.Double(-0.7, 0.7);
  incrs[W] = new Point2D.Double(-1.0, 0.0);
  incrs[NW] = new Point2D.Double(-0.7, -0.7);

  //...
}
```

Llogaritja e pikës së re të kokës

```
private Point nextPoint(int prevPosn, int bearing) {
// merr rritjet për drejtimin
  Point2D.Double incr = incrs[bearing];

  int newX = cells[prevPosn].x + (int)(DOTSIZE * incr.x);
  int newY = cells[prevPosn].y + (int)(DOTSIZE * incr.y);

  // modifiko nëse jashtë pëlhurës
  if (newX+DOTSIZE < 0)     // jashtë skajit të majtë?
    newX = newX + pWidth;
    else  if (newX > pWidth)  // jashtë skajit të djathtë?
    newX = newX - pWidth;

    if (newY+DOTSIZE < 0)     // jashtë skajit të sipërm?
    newY = newY + pHeight;
    else  if (newY > pHeight) // jashtë skajit të poshtëm?
    newY = newY - pHeight;

  return new Point(newX, newY);
}
```

Zgjedhja e drejtimit të lëvizjes

```
private int varyBearing() {
  int newOffset = probsForOffset[ (int)( Math.random()*NUM_PROBS )];
  return calcBearing(newOffset);
}

private static final int NUM_PROBS = 9;
private int probsForOffset[];

public Worm(int pW, int pH, Obstacles os) {
  //...
  // informata mbi porbabilitetin e zgjedhjes së një drejtimi
  //    0 = pa ndryshim, -1 në drejtim të kundërt me akrepat,
  //    1 në drejtim të akrepave, etj.
  // Zakonisht në të njëjtin drejtim me pak ndryshim anash
  probsForOffset = new int[NUM_PROBS];
  probsForOffset[0] = 0;  probsForOffset[1] = 0;
  probsForOffset[2] = 0;  probsForOffset[3] = 1;
  probsForOffset[4] = 1;  probsForOffset[5] = 2;
  probsForOffset[6] = -1;  probsForOffset[7] = -1;
  probsForOffset[8] = -2;
}
```

Zgjedhja e drejtimit të lëvizjes (Vazhdim)

```
private int calcBearing(int offset) {
  int turn = currCompass + offset;
  // sigurohu që vlera të jetë ndërmjet N dhe NW
  if (turn >= NUM_DIRS)
    turn = turn - NUM_DIRS;
  else if (turn < 0)
    turn = NUM_DIRS + turn;
  return turn;
}
```

Tejkalimi i pengesave

```
private void newHead(int prevPosn) {
  int newBearing = varyBearing();
  Point newPt = nextPoint(prevPosn, newBearing);

  int fixedOffs[] = {-2, 2, -4}; // për të tejkaluar pengesë

  if (obs.hits(newPt, DOTSIZE)) {
    for (int i=0; i < fixedOffs.length; i++) {
     newBearing = calcBearing(fixedOffs[i]);
     newPt = nextPoint(prevPosn, newBearing);
     if (!obs.hits(newPt, DOTSIZE))
       break;  // një nga zhvendosjet e fiksuara bën punë
    }
  }
  cells[headPosn] = newPt;      // pozita e re e kokës
  currCompass = newBearing;     // drejtimi i ri i busollës
}
```

Lëvizja e krimbit

Lëvizja e krimbit bëhet varësisht nga faza aktuale në zhvillimin e krimbit:

1. Krijimi i parë i krimbit
2. Krimbi rritet, por `cells[]` nuk është mbushur
3. `cells[]` është mbushur, prandaj krijimi i kokës balancohet me heqjen e bishtit

Lëvizja e krimbit (Vazhdim)

```
public void move() {
int prevPosn = headPosn;  // ruaje pozitën e vjetër
  headPosn = (headPosn + 1) % MAXPOINTS;

  if (nPoints == 0) {   // varg bosh në fillim
    tailPosn = headPosn;
    currCompass = (int)(Math.random()*NUM_DIRS); // drejt. i rast.
    cells[headPosn] = new Point( pWidth/2, pHeight/2 ); // qendra
    nPoints++;
  }
  else if (nPoints == MAXPOINTS) {    // vargu është plot
     tailPosn = (tailPosn + 1) % MAXPOINTS;  // heq bishtin
    newHead(prevPosn);
  }
  else {     // vargu nuk është plot
    newHead(prevPosn);
    nPoints++;
  }
}
```

Vizatimi i krimbit

```
public void draw(Graphics g) {
  if (nPoints > 0) {
    g.setColor(Color.black);
    int i = tailPosn;
    while (i != headPosn) {
     g.fillOval(cells[i].x, cells[i].y, DOTSIZE, DOTSIZE);
     i = (i+1) % MAXPOINTS;
    }
    g.setColor(Color.red);
    g.fillOval(cells[headPosn].x, cells[headPosn].y,
        DOTSIZE, DOTSIZE);
  }
}
```

Testimi i krimbit

```
/* A është (x,y) afër kokës? */
public boolean nearHead(int x, int y) {
  if (nPoints > 0) {
     if( (Math.abs(cells[headPosn].x + RADIUS - x) <= DOTSIZE) &&
        (Math.abs(cells[headPosn].y + RADIUS - y) <= DOTSIZE) )
      return true;
  }
  return false;
}
```

Testimi i krimbit (Vazhdim)

```
public boolean touchedAt(int x, int y) {
  int i = tailPosn;
  while (i != headPosn) {
    if( (Math.abs( cells[i].x + RADIUS - x) <= RADIUS) &&
        (Math.abs( cells[i].y + RADIUS - y) <= RADIUS) )
      return true;
    i = (i+1) % MAXPOINTS;
  }
  return false;
}
```

Pengesat

```
public class Obstacles {
  private static final int BOX_LENGTH = 12;

  private ArrayList<Rectangle> boxes;    // objekte Rectangle
  private WormChase wcTop;

  public Obstacles(WormChase wc) {
    boxes = new ArrayList<Rectangle>();
    wcTop = wc;
  }

  synchronized public void add(int x, int y) {
    boxes.add(new Rectangle(x,y, BOX_LENGTH, BOX_LENGTH));
    wcTop.setBoxNumber(boxes.size() );   // raporto numrin e ri
  }
  // ... Anëtarët tjerë
}
```

Pengesat (Vazhdim)

```
/* Vizato një varg kutish të kaltra */
synchronized public void draw(Graphics g) {
  Rectangle box;
  g.setColor(Color.blue);
  for(int i=0; i < boxes.size(); i++) {
    box = boxes.get(i);
    g.fillRect(box.x, box.y, box.width, box.height);
  }
}
```

Pengesat (Vazhdim)

```
/* A ka prejre p me ndonjërën nga pengesat? */
synchronized public boolean hits(Point p, int size) {
  Rectangle r = new Rectangle(p.x, p.y, size, size);
  Rectangle box;
  for(int i=0; i < boxes.size(); i++) {
    box = boxes.get(i);
    if (box.intersects(r))
   return true;
  }
  return false;
}
```

Udhëzime për lexim të mëtejmë

- `http://www.fberisha.org`
- A. Davison, *Killer Game Programming in Java,* kapitulli 3
  `http://fivedots.coe.psu.ac.th/ ad/jg/`

Përfundim

- Implementimi i pëlhurës së fijëzuar për një lojë
- Korniza dhe GUI i aplikacionit
- Modelimi i lojës

Detyra shtëpie

1. Ekzekutoni aplikacionin ashtu që krimbi të lëvizë më ngadalë, p.sh. 10 FPS.
2. Modifikoni aplikacionin ashtu që krimbi të jetë me kokë të kuqe, por me trup të gjelbërt.
3. Modifikoni aplikacionin ashtu që loja të përfundojë jo vetëm kur klikohet afër kokës por edhe afër trupit të krimbit.
4. Sa pika merr shfrytëzuesi në qoftë se e përfundon lojën për 30 sekonda dhe vë 10 pengesa?
5. Modifikoni aplikacionin ashtu që shfrytëzuesi të marrë pika nga inervali 1–200 për kohën dhe 1–100 për numrin e pengesave.
6. Modifikoni aplikacionin ashtu që krimbi të lëvizë në të gjitha drejtimet e busolës me probabilitet të njëjtë.
7. Modifikoni aplikacionin ashtu që krimbi të kalojë rreth pengesave duke u kthyer vetëm në të djathtë (ose prapa), por jo edhe në të majtë.
8. Modifikoni aplikacionin ashtu që krimbi dhe pengesat të jenë më të mëdha, kurse loja të përfundojë edhe në qoftë se klikohet në një rrethinë më të gjerë të kokës së krimbit.
9. Modifikoni aplikacionin ashtu që të animohen dy krimba në pëlhurën. Çfarë do të ndodhë në rast të takimit të tyre?
10. Modifikoni aplikacionin origjinal ashtu që me startimin e lojës fillimisht të vizatohen një numër pengesash, p.sh. 7 sish, në pozita të rastësishme në panelin.
11. Modifikoni aplikacionin nga detyra paraprake ashtu që drejtimi i lëvizjes së krimbit të mund të ndryshohet nga lojtari: kthime të lehta djathtas dhe majtas me shtypjen e tasteve të shigjetave përkatëse në tastierë.
12. Modifikoni aplikacionin ashtu që krimbi të mbajë drejtimin e pandryshuar të lëvizjes. Drejtimi i lëvizjes së krimbit le të ndryshohet nga lojtari sikur në detyrën paraprake. Loja le të përfundojë në qoftë se krimbi godet ndonjë pengesë. Rezultati i afishuar le të jetë shuma e numrit të sekondave të kaluara në lojë dhe numrit të pengesave.