Sprajtet

Faton Berisha

Fakulteti i Shkencave Kompjuterike
Universiteti i Prizrenit

Qëllimet dhe objektivat

- Kodimi i objekteve lëvizëse të një loje si sprajte.
- Zhvillimi i `class Sprite` e cila mban pozitën e një sprajti, shpejtësinë e tij dhe shfrytëzon klasat e imazheve (`ImagesLoader`, `ImagesPlayer`) për të menazhuar prezencën grafike të tij.
- Nënklasat e `class Sprite` shtojnë interaksione të shfrytëzuesit dhe rrethinës.
- Kodimi i nënklasave ndihmohet nga specifikimi i UML diagramëve të gjendjes.

Përmbajtja

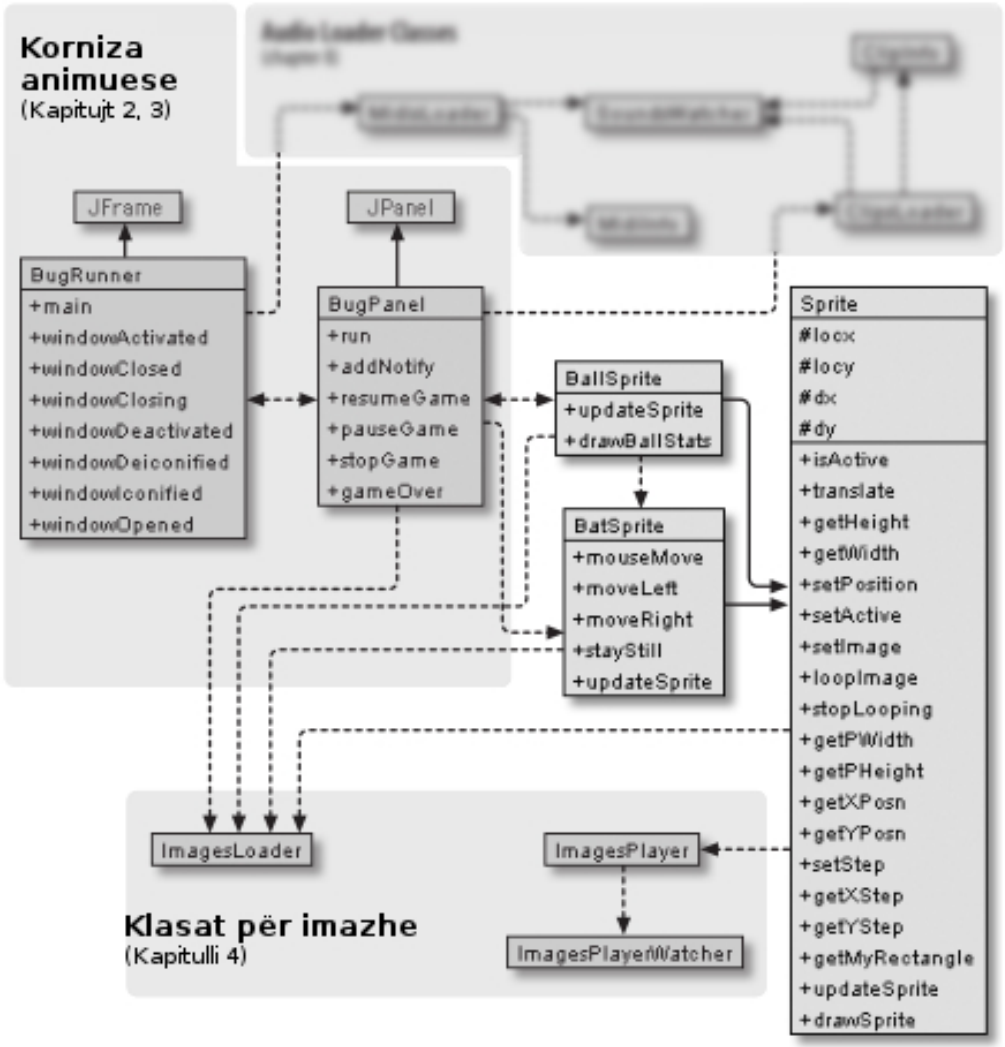
Sprajtet

- *Sprajt:* objekt grafik lëvizës, i cili mund të përfaqësojë lojtarin (të përgjigjet në trusje tastesh dhe aksione mausi), ose të udhëhiqet nga kod "inteligjent" në lojën.

Shkopinjë, topa dhe sprajte

- Shembulli `BugRunner` i lejon shfrytëzuesit të kontrollojë një sprajt në formë milingone.
- Milingona kontrollohet me tastet shigjeta ose me maus.
- Sprajte topash biejnë me shpejtësi dhe trajektore të ndryshueshme nga skaji i sipërm i panelit.

Diagrami i klasave



Insekti fillon të vrapojë

- Klasa `BugRunner` është e ngjashme me klasën `WormChase` nga kapitulli 2.
- `BugRunner` është `JFrame` dhe paraqet dritaren kryesore dhe aplikacionin.
- `BugRunner` fikson shpejtësinë në 40 FPS (për shkak të lojës).

Korniza animuese

- Klasa `BugPanel` është `JPanel` dhe implementon kornizën animuese siç është përshkruar në kapitujt 2 dhe 3.
- Konstruktori vë dëgjuesit e tastierës dhe mausit përgadit objektin ImageLoader dhe krijon sprajtet për shkopin dhe topat.

Konstruktori

```
public BugPanel(BugRunner br, long period) {
  bugTop = br;
  this.period = period;

  setDoubleBuffered(false);
  setBackground(Color.black);
  setPreferredSize( new Dimension(PWIDTH, PHEIGHT));

  setFocusable(true);
  requestFocus();    // JPanel ka fokusin, prano ngjarje tastesh

  addKeyListener( new KeyAdapter() {
    public void keyPressed(KeyEvent e)
    { processKey(e);  }
  });

  // ngarko imazhin e prapavisë
  ImagesLoader imsLoader = new ImagesLoader(IMS_INFO);
  bgImage = imsLoader.getImage("bladerunner");
```

Konstruktori (Vazhdim)

```
  // krijo sprajtet e lojës
  bat = new BatSprite(PWIDTH, PHEIGHT, imsLoader,
    (int)(period) ); // në ms
  ball = new BallSprite(PWIDTH, PHEIGHT, imsLoader, this, bat);

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

  // vëj fontin e mesazheve
  msgsFont = new Font("SansSerif", Font.BOLD, 24);
  metrics = this.getFontMetrics(msgsFont);
}
```

Interaksioni i shfrytëzuesit

```
/* Përpunon përfundimin dhe tastet e lojës */
private void processKey(KeyEvent e) {
  int keyCode = e.getKeyCode();

  // dëgjo për esc, q, end, ctrl-c
  if ((keyCode == KeyEvent.VK_ESCAPE) || (keyCode == KeyEvent.VK_Q) ||
    (keyCode == KeyEvent.VK_END) ||
    ((keyCode == KeyEvent.VK_C) && e.isControlDown()) )
    running = false;

  // tastet e lojës
  if (!isPaused && !gameOver) {
    if (keyCode == KeyEvent.VK_LEFT)
    bat.moveLeft();
    else if (keyCode == KeyEvent.VK_RIGHT)
    bat.moveRight();
    else if (keyCode == KeyEvent.VK_DOWN)
    bat.stayStill();
  }
}
```

Interaksioni i shfrytëzuesit (Vazhdim)

```
/* Shfrytëzo mausin për të kontrolluar shkopin */
private void testPress(int x, int y) {
  if (!isPaused && !gameOver)
    bat.mouseMove(x);
}
```

Cikli i animimit

```
private long gameStartTime;

public void run() {
  long beforeTime, timeDiff, sleepTime;
  gameStartTime = System.currentTimeMillis();
  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();
  }
  System.exit(0);
}
```

Cikli i animimit (Vazhdim)

```
private void gameUpdate() {
  if (!isPaused && !gameOver) {
    ball.updateSprite();
    bat.updateSprite();
  }
}
```

Cikli i animimit (Vazhdim)

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

  // vizato prapavinë: shfrytëzo imazhin si ngjyrë të zezë
  if (bgImage == null) {
    dbg.setColor(Color.black);
    dbg.fillRect (0, 0, PWIDTH, PHEIGHT);
  } else
    dbg.drawImage(bgImage, 0, 0, this);

  // vizato elementet e lojës
  ball.drawSprite(dbg);
  bat.drawSprite(dbg);

  reportStats(dbg);

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

Cikli i animimit (Vazhdim)

```
/* Raporto numrin e topave të kthyer dhe kohën e kaluar në lojë */
private void reportStats(Graphics g) {
  if (!gameOver)    // rrit tajmerim
    timeSpentInGame =  // ms --> sek
     (int) ((System.currentTimeMillis() - gameStartTime)/1000);

  g.setColor(Color.yellow);
  g.setFont(msgsFont);

  ball.drawBallStats(g, 15, 25); // sprajti top raporton stat. e veta
  g.drawString("Time: " + timeSpentInGame + " secs", 15, 50);

  g.setColor(Color.black);
}
```

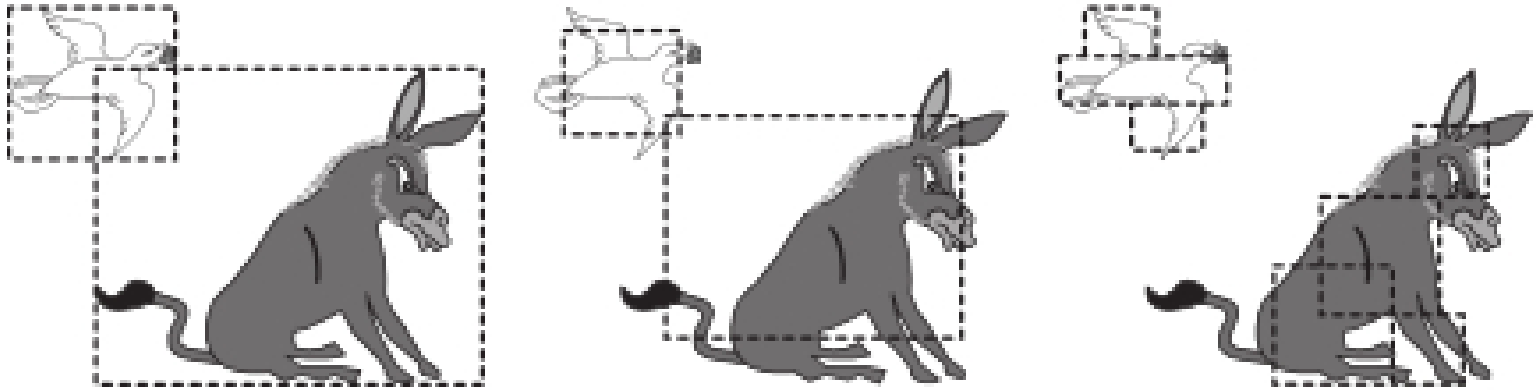
Përfundimi i lojës

```
/* Invokohet nga BallSprite për të sinjalizuar fundin */
public void gameOver() {
  int finalTime =  // ms --> sek
    (int) ((System.currentTimeMillis() - gameStartTime)/1000);
  score = finalTime;   // mund të jetë më kompleks!
  gameOver = true;
}
```

Definimi i një sprajti

- Një klasë sprajtesh me destinim të përgjithshëm është e vështirë të disenjohet.
  - Një sprajt mund të lëvizë nëpër sipërfaqe (Tetris, Breakout, Space Invaders) përtej skajeve të panelit ose jo.
  - Një sprajt mund të lëvizë krejt pak (Super Mario), e të skrollohet prapavija.
- Një sprajt duhet të monitorojë rrethinën e lojës (p.sh., të detektojë ndeshjet).
- Përpunimi i ndeshjeve
  - Detektimi i ndeshjeve
  - Përgjegjja e ndeshjes

Tri tipet e detektimit të një ndeshjeje



Kodimi i një sprajti

- `class Sprite` ruan pozitën e sprajtit, shpejtësinë e tij me menazhim imazhi nga `ImagesLoader` dhe `ImagesPlayer`
- Nënklasat e `Sprite`, `BatSprite` dhe `BallSprite` menazhojnë elementet specifike për lojën:
  - interaksionin e shfrytëzuesit
  - rrethinën
  - kolizionet

Kodimi i një sprajti (Vazhdim)

```
// default shpejtësia e sprajtit
private static final int XSTEP = 5;
private static final int YSTEP = 5;

private int pWidth, pHeight;   // dimensionet e panelit

// fusha protected
protected int locx, locy;         // lokacioni i sprajtit
protected int dx, dy;             // shpejtësia e lëvizjes
```

Konstruktori

```
public Sprite(int x, int y, int w, int h,
      ImagesLoader imsLd, String name) {
  locx = x; locy = y;
  pWidth = w; pHeight = h;
  dx = XSTEP; dy = YSTEP;  // shpejtësia default

  imsLoader = imsLd;
  setImage(name);    // imazhi default i sprajtit
}
```

Imazhi i një sprajti

```
// default dimensionet kur nuk ka imazh
private static final int SIZE = 12;

// imazhimi
private ImagesLoader imsLoader;
private String imageName;
private BufferedImage image;
private int width, height;     // dimensionet e imazhit

private ImagesPlayer player;  // për të luajtur cikël imazhesh
private boolean isLooping;
```

Imazhi i një sprajti (Vazhdim)

```
/* I shoqëron sprajtit emrin e imazhit */
public void setImage(String name) {
  imageName = name;
  image = imsLoader.getImage(imageName);
  if (image == null) {  // nuk është gjetur imazh me atë emër
    System.out.println("No sprite image for " + imageName);
    width = SIZE;
    height = SIZE;
  } else {
    width = image.getWidth();
    height = image.getHeight();
  }
  // nuk luhet cikël imazhi
  player = null;
  isLooping = false;
}
```

Imazhi i një sprajti (Vazhdim)

```
public void loopImage(int animPeriod, double seqDuration) {
  if (imsLoader.numImages(imageName) > 1) {
    player = null;   // për të inkurajuar garbage collection
    player = new ImagesPlayer(imageName, animPeriod, seqDuration,
      true, imsLoader);
    isLooping = true;
  } else
    System.out.println(imageName + " is not a sequence of images");
}
```

Imazhi i një sprajti (Vazhdim)

```
public void stopLooping() {
  if (isLooping) {
    player.stop();
    isLooping = false;
  }
}
```

Kutia kufizuese e një sprajti

```
public Rectangle getMyRectangle() {
  return  new Rectangle(locx, locy, width, height);
}
```

Azhornimi i një sprajti

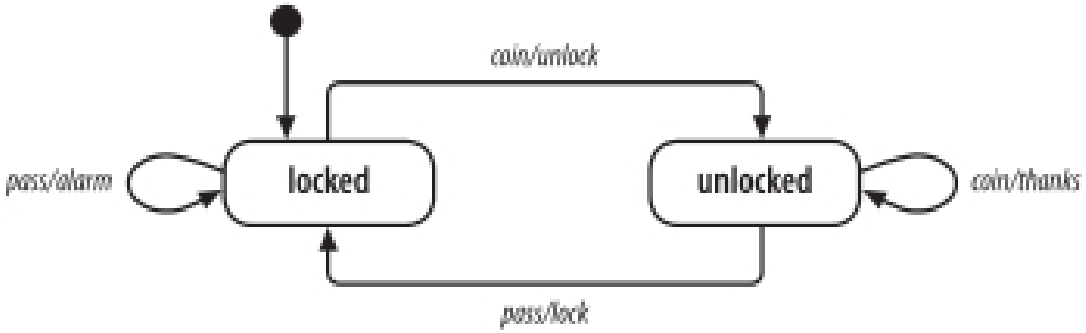
```
private boolean isActive = true;
// një sprajt azhornohet dhe vizatohet vetëm kur është aktiv

/* Lëviz sprajtin */
public void updateSprite() {
  if (isActive()) {
    locx += dx;
    locy += dy;
    if (isLooping)
      player.updateTick();  // azhorno player-in
  }
}
```

Vizatimi i një sprajti

```
public void drawSprite(Graphics g) {
  if (isActive()) {
    if (image == null) {   // sprajti nuk ka imazh
    g.setColor(Color.yellow);   // vizato një rreth të verdhë
    g.fillOval(locx, locy, SIZE, SIZE);
    g.setColor(Color.black);
    } else {
     if (isLooping)
       image = player.getCurrentImage();
     g.drawImage(image, locx, locy, null);
    }
  }
}
```

Diagrami i gjendjes i një vendkalimi



- Sintaksa:

  `event [condition] / action`

Përkthimi në kod i një diagrami gjendjeje

Gjendja aktuale	Ngjarja	Aksioni	Gjendja e re
`locked`	`coin`	`unlock`	`unlocked`
`locked`	`pass`	`alarm`	`locked`
`unlocked`	`coin`	`thanks`	`unlocked`
`unlocked`	`pass`	`lock`	`locked`

Tabela: State Transition Table (STT) e diagramit të gjendjes

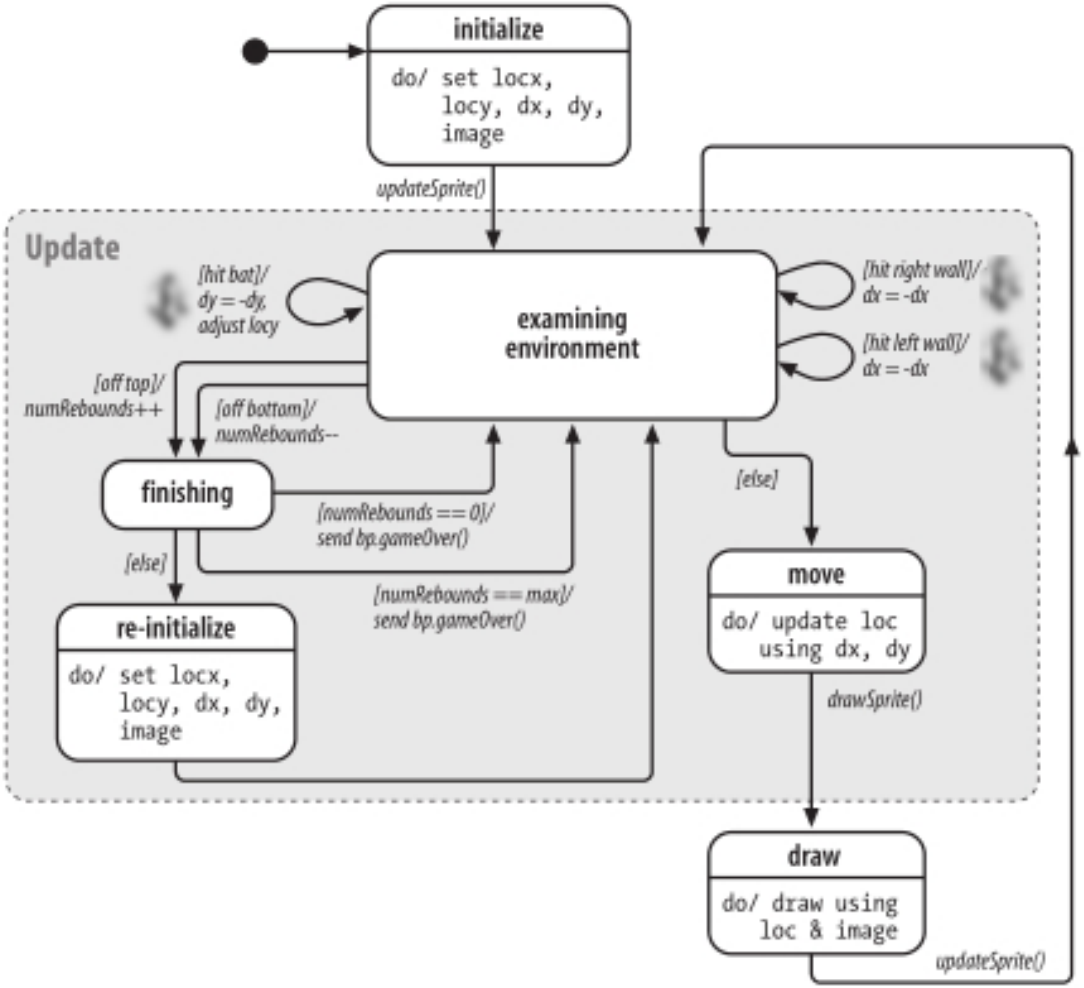
Përkthimi në kod i një diagrami gjendjeje (Vazhdim)

```
public class TurnStile {
  // konstantat e gjendjes
  private static final int LOCKED = 0;
  private static final int UNLOCKED = 1;
  // konstantat e ngjarjeve
  private static final int COIN = 2;
  private static final int PASS = 3;
  public static void main(String args[]) {
    int currentState = LOCKED;
    int event;
    while (true) {
      event = //... merr ngjarjen e ardhshme;
      currentState = makeTransition(currentState, event);
    }
  }
```

Përkthimi në kod i një diagrami gjendjeje (Vazhdim)

```
  /* Transicion sipas tabelës */
  private static int makeTransition(int state, int event) {
    if ((state == LOCKED) && (event == COIN)) {
      unlock();
      return UNLOCKED;
    }
    else if ((state == LOCKED) && (event == PASS)) {
      alarm();
      return LOCKED;
    }
    else if ((state == UNLOCKED) && (event == COIN)) {
      thanks();
      return UNLOCKED;
    }
    else if ((state == UNLOCKED) && (event == PASS)) {
      lock();
      return LOCKED;
    }
    else {
      System.out.println("Unknown state event");
      System.exit(0);
    }
    }
    // metodat për aksionet: unlock(), alarm(), thanks(), lock()
}
```

Diagrami i gjendjes së `BallSprite`



Përkthimi në kod i topit

```
public class BallSprite extends Sprite {
  // x- dhe y-vlerat e hapit STEP +/- STEP_OFFSET
  private static final int STEP = 8;
  private static final int STEP_OFFSET = 2;

  private static final String[] ballNames =
      {"rock1", "orangeRock", "computer", "ball"};
    // imazhet e topave

  // numri i topave për të përfunduar loja
  private static final int MAX_BALLS_RETURNED = 16;

  private int nameIndex;  // për zgjdhjen e emrit

  private BugPanel bp;
  private BatSprite bat;

  private int numRebounds;
  // ... anëtarët tjerë
}
```

Përkthimi në kod i topit (Vazhdim)

```
public BallSprite(int w, int h, ImagesLoader imsLd,
      BugPanel bp, BatSprite b) {
  super( w/2, 0, w, h, imsLd, ballNames[0]);
  this.bp = bp;
  bat = b;

  nameIndex = 0;
  numRebounds = MAX_BALLS_RETURNED/2;
   // numri i kthimeve fillon nga gjysma e max
  initPosition();
}
```

Përkthimi në kod i topit (Vazhdim)

```
/* Inicializo imazhin, pozitën dhe shpejtësinë */
private void initPosition() {
  setImage( ballNames[nameIndex]);
  nameIndex = (nameIndex+1)%ballNames.length;

  setPosition( (int)(getPWidth() * Math.random()), 0);

  int step = STEP + getRandRange(STEP_OFFSET);
  // majtas ose djathtas
  int xStep = ((Math.random() < 0.5) ? -step : step);
  // tatpjet
  setStep(xStep, STEP + getRandRange(STEP_OFFSET));
}

private int getRandRange(int x) {
  return ((int)(2 * x * Math.random())) - x;
}
```

Përkthimi në kod i topit (Vazhdim)

```
public void updateSprite() {
  hasHitBat();
  goneOffScreen();
  hasHitWall();

  super.updateSprite();
}

private void hasHitBat() {
  Rectangle rect = getMyRectangle();
  if (rect.intersects( bat.getMyRectangle() )) {// kolizion?
    Rectangle interRect = rect.intersection(bat.getMyRectangle());
    dy = -dy;        // ndrysho drejtimin vertikal
    locy -= interRect.height;   // përpjet
  }
}
```

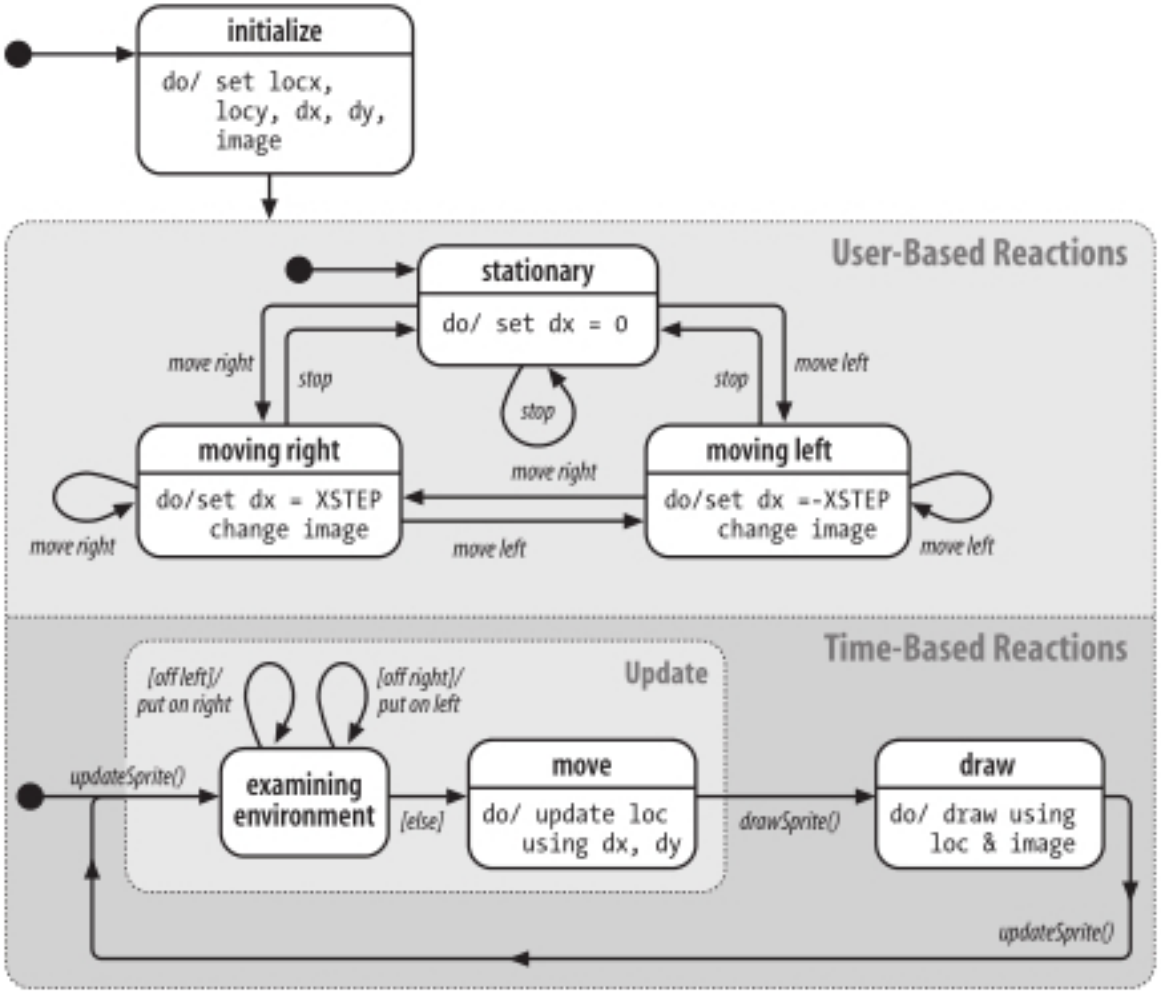
Përkthimi në kod i topit (Vazhdim)

```
private void hasHitWall() {
  if ((locx <= 0) && (dx < 0)) {  // majtas
    dx = -dx;   // ndrysho drejtimin horizontal
  }
  else if ((locx+getWidth() >= getPWidth()) && (dx > 0)) {
    // djathtas
    dx = -dx;
  }
}
```

Përkthimi në kod i topit (Vazhdim)

```
private void goneOffScreen() {
  if (((locy+getHeight()) <= 0) && (dy < 0)) { // mbi
    numRebounds++;
    if (numRebounds == MAX_BALLS_RETURNED)
      bp.gameOver();    // përfundo
    else
      initPosition();   // fillo në pozitë të re
  }
  else if ((locy >= getPHeight()) && (dy > 0)) { // nën
    numRebounds--;
    if (numRebounds == 0)
      bp.gameOver();
    else
      initPosition();
  }
}
```

Diagrami i gjendjes së BatSprite



Përkthimi në kod i shkopit

```
public class BatSprite extends Sprite {
// koha totale në sek për ciklim nëpër imazhet
  private static double DURATION = 0.5;

  private static final int FLOOR_DIST = 41;

  private static final int XSTEP = 10;

  private int period; // e animimit të lojës

  public BatSprite(int w, int h, ImagesLoader imsLd, int p) {
    super( w/2, h-FLOOR_DIST, w, h, imsLd, "leftBugs2");
    period = p;
    setStep(0,0);
  }
  //... anëtarët tjerë
}
```

Përkthimi në kod i shkopit (Vazhdim)

```
public void moveLeft() {
  setStep(-XSTEP, 0);
  setImage("leftBugs2");
  loopImage(period, DURATION);  // ciklo nëpër imazhet
}

public void moveRight() {
  setStep(XSTEP, 0);
  setImage("rightBugs2");
  loopImage(period, DURATION);  // ciklo nëpër imazhet
}

public void stayStill() {
  setStep(0, 0);
  stopLooping();
}
```

Përkthimi në kod i shkopit (Vazhdim)

```
public void mouseMove(int xCoord) {
  if (xCoord < locx)  // klikimi në të majtë
    moveLeft();       // majtas
  else if (xCoord > (locx + getWidth())) // në të djathtë
    moveRight();      // djathtas
  else
    stayStill();
}
```

Përkthimi në kod i shkopit (Vazhdim)

```
public void updateSprite() {
  if ((locx+getWidth() <= 0) && (dx < 0))   // majtas
    locx = getPWidth()-1;       // futu nga e djathta
  else if ((locx >= getPWidth()-1) && (dx > 0))  // djathtas
    locx = 1 - getWidth();      // futu nga e majta

  super.updateSprite();
}
```

Udhëzime për lexim të mëtejmë

- `http://www.fberisha.org`
- A. Davison, *Killer Game Programming in Java,* kapitulli 11
  `http://fivedots.coe.psu.ac.th/ ad/jg/`
- A. Feldman, *SpriteLib,*
  `http://www.arifeldman.com/games/spritelib.html`
- R. C. Martin, *UML for Java Programmers,*
  `http://www.csd.uoc.gr/~hy252/references/UML_for_Java_Programmers-Book.pdf`

Përfundim

- Zhvillimi i `class Sprite` si një kornize sprajtesh
- Implementimi i elementeve specifike të lojës në nënklasa të `Sprite`
- Specifikimi i UML diagramëve të gjendjes së sprajtëve dhe përkthimi në kod i tyre