Një kornizë animimi

Faton Berisha

Fakulteti i Shkencave Kompjuterike
Universiteti i Prizrenit

Qëllimet dhe objektivat

- Një kornizë animimi: Ndërtimi i algoritmit të animimit të një loje si pëlhurë për vizatim të grafikës 2D të menazhuar nga një fije (thread).
- Cikli `update`, `render`, `sleep` i animimit
- Startimi dhe terminimi i një aplikacioni
- Baferimi i dyfishtë
- Interaksioni i shfrytëzuesit
- Renderimi aktiv
- Kontrolli i animimit mbështetur në *frames per second (FPS)* të kërkuar nga shfrytëzuesi
- Pauzimi dhe vazhdimi i lojës

Përmbajtja

1. Animimi si një pëlhurë e fijëzuar
   - Baferimi i dyfishtë
2. Shtimi i interaksionit të shfrytëzuesit
3. Koha dhe fjetja
   - Konvertimi në renderim aktiv
   - FPS dhe fjetja për kohë të ndryshueshme
4. Pauzimi dhe vazhdimi

Animimi si një pëlhurë e fijëzuar

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class GamePanel extends JPanel implements Runnable {
  private static final int PWIDTH = 500; // madhësia e panelit
  private static final int PHEIGHT = 400;
  private Thread animator; // për animim
  private volatile boolean running = false; // ndalon animimin
  private volatile boolean gameOver = false; // terminon lojën
  // fusha tjera, të shpjeguara më vonë
  //       ...

  public GamePanel() {
    setBackground(Color.white);    // prapavi e bardhë
    setPreferredSize(new Dimension(PWIDTH, PHEIGHT));
    // krijo komponentat e lojës
    // ...
  }
```

Animimi si një pëlhurë e fijëzuar (Vazhdim)

```
/* Prit që ky JPanel të shtohet te
 JFrame/JApplet para startimit. */
public void addNotify() {
  super.addNotify(); // krijon peer-in
  startGame();       // starton fijen
}

/* Inicializo dhe starto fijen */
private void startGame() {
  if (animator == null || !running) {
    animator = new Thread(this);
    animator.start();
  }
}

/* Invokohet nga shfrytëzuesi për të ndaluar ekzekutimin */
public void stopGame() {
  running = false;
}
```

Animimi si një pëlhurë e fijëzuar (Vazhdim)

```
/* Përsërit: update, render, sleep */
public void run() {
  running = true;
  while(running) {
    gameUpdate();   // azhorno gjendjen e lojës
    gameRender();   // rendero te një bafer
    repaint();      // vizato me baferin
    try {
       Thread.sleep(20);  // flej pak
    }
    catch(InterruptedException e){ }
  }
  System.exit(0); // po kështu edhe JFrame/JApplet rrethues
}
```

Animimi si një pëlhurë e fijëzuar (Vazhdim)

```
  /* Azhorno gjendjen e lojës */
  private void gameUpdate() {
    if (!gameOver) {
      //  ...
    }
  }
  // metoda tjera, të shpjeguara më vonë...
}
```

Çështje animimi dhe fijëzimi

- Probleme me sinkronizim: aplikacionet me dy ose më tepër fije.
  - Java memory Model (JMM) deklaron se qasja variablave të tipeve primitive përveç `long` dhe `double` është atomike (atomiciteti 32-bitësh).
  - E dobishme: qasja atomike variablave `boolean`
  - Mundësi problemesh sinkronizimi me struktura të dhënash
- Terminimet e aplikacionit dhe lojës
  - Dy variable: `running` dhe `gameOver`
- Përdorimi i `volatile`
  - JMM lejon secilin thread të ketë memorien lokale
  - `running` vëhet në `false` nga `stopGame()` prej GUI
  - `stopGame` vëhet në `true` prej GUI
  - `volatile` ndalon variablën të kopjohet në memorien lokale

Çështje animimi dhe fijëzimi (Vazhdim)

- Përdorimi i `sleep()`
  - bën që fija e animimit të pushojë së ekzekutuari, duke liruar CPU
  - i jep kohë `repaint()` paraprak për t'u procesuar
  - redukton gjasën për *bashkërritje ngjarjesh (event coalescence)*

Baferimi i dyfishtë

```
// fushat për renderim off-screen
private Graphics dbg;
private Image dbImage = null;

/* Vizato kornizën aktuale në një bafer imazhi */
private void gameRender() {
  if (dbImage == null) {  // krijo baferin
    dbImage = createImage(PWIDTH, PHEIGHT);
    if (dbImage == null) {
     System.out.println("dbImage është null");
     return;
    } else { dbg = dbImage.getGraphics(); }
  }
  // pastro prapavinë
  dbg.setColor(Color.white);
  dbg.fillRect (0, 0, PWIDTH, PHEIGHT);
  // vizato elementet e lojës
  // ...
  if (gameOver) {
    gameOverMessage(dbg);
  }
}
```

Baferimi i dyfishtë (Vazhdim)

```
/* Afisho mesazhin game-over */
private void gameOverMessage(Graphics g) {
  String msg = "Game Over";
  // Kodi për llogaritjen e x dhe y...
  g.drawString(msg, x, y);
}

public void paintComponent(Graphics g) {
  super.paintComponent(g);
  if (dbImage != null) {
    g.drawImage(dbImage, 0, 0, null);
  }
}
```

Shtimi i interaksionit të shfrytëzuesit

- Shfrytëzuesi do të dëshironte sa me tepër interaksion të drejtpërdrejt me pëlhurën e lojës.
  - `GamePanel` duhet të monitorojë shtypjet e tasteve dhe aktivitetin e mausit.
- `GamePanel` zbaton shtypjet e tasteve për setuar `running` në `false`.
- Shtypjet e mausit procesohen nga `testPress()`.
- Modifikojmë konstruktorin `GamePanel()` për të vënë përpunuesit e tasteve dhe të mausit.

Shtimi i interaksionit të shfrytëzuesit (Vazhdim)

```
public GamePanel() {
  setBackground(Color.WHITE);
  setPreferredSize(new Dimension(PWIDTH, PHEIGHT));
  setFocusable(true);
  requestFocus();    // JPanel tani pranon ngjarje tastesh
  readyForTermination();
  // krijo komponentet e lojës
  // ...
  // dëgjo për shtypjet e mausit
  addMouseListener(new MouseAdapter() {
      public void mousePressed(MouseEvent e) {
        testPress(e.getX(), e.getY());
      }
    });
}
```

Shtimi i interaksionit të shfrytëzuesit (Vazhdim)

```
private void readyForTermination() {
      // dëgjo për esc, q, end, ctrl-c
  addKeyListener(new KeyAdapter() {
      public void keyPressed(KeyEvent e) {
        int keyCode = e.getKeyCode();
        if ((keyCode == KeyEvent.VK_ESCAPE) ||
        (keyCode == KeyEvent.VK_Q) ||
        (keyCode == KeyEvent.VK_END) ||
        ((keyCode == KeyEvent.VK_C) && e.isControlDown()) ) {
          running = false;
        }
      }
    });
}
```

Shtimi i interaksionit të shfrytëzuesit (Vazhdim)

```
/* A është (x,y) e rëndësishme për lojën? */
private void testPress(int x, int y) {
  if (!gameOver) {
    // bëj diçka
  }
}
```

Konvertimi në renderim aktiv

- Koha e fjetjes në ciklin a animimit është vënë me hamendje.
- Koha e fjetjes duhet të rekalkulohet në secilin iterim pas matjes së periudhave të azhormimit dhe renderimit të iterimit.
  - `repaint()` kryhet nga JVM dhe nuk mund të matet lehtë.
  - *Renderimi aktiv*, si modifikim i `run()`

Konvertimi në renderim aktiv (Vazhdim)

```
/* Përsërit: update, render, sleep */
public void run() {
  running = true;
  while(running) {
    gameUpdate(); // azhorno gjendjen e lojës
    gameRender(); // rendero te një bafer
    paintScreen(); // vizato baferin në ekran
    try {
      Thread.sleep(20);  // flej pak
    }
    catch(InterruptedException ex){}
  }
  System.exit(0); // edhe JFrame/JApplet rrethues
}
```

Konvertimi në renderim aktiv (Vazhdim)

```
/* Aktivisht rendero në ekran imazhin e baferit */
private void paintScreen() {
  Graphics g;
  try {
    g = this.getGraphics();  // merr kontekstin grafik
    if ((g != null) && (dbImage != null))
      g.drawImage(dbImage, 0, 0, null);
    Toolkit.getDefaultToolkit().sync();  // sync në disa sist.
    g.dispose();
  }
  catch (Exception e) {
    System.out.println("Gabim në kontekstin grafik: " + e);
  }
}
```

Konvertimi në renderim aktiv (Vazhdim)

- Largohet invokimi i `repaint()` dhe mbikalimi i `paintComponent()`; funksionaliteti i saj në `paintScreen()`
- Merret kontrolli i renderimit të baferit:
  koha e renderimit mund të matet saktë
- Konteksti grafik i panelit mund të ndryshohet nga JVM
  dhe mund të zhduket:
  konteksti grafik duhet të merret (duke invokuar `getGraphics()`)
  dhe të rrethohet me try-catch
- Invokimi i `Toolkit.sync()` siguron që displeji të azhurnohet shpejt (në Linux).

FPS dhe fjetja për kohë të ndryshueshme

- Shpejtësia e ekzekutimit të ciklit duhet të jetë afërsisht e njëjtë në të gjitha platformat.
- Një masë e popullarizuar e shpejtësisë së progresit të një loje është numri i kornizave për sekond (*Frames per second, FPS*).
  - Një kornizë (*frame*) është një kalim nëpër iterimin update-render-sleep.
- Versioni i modifikuar i `run()` përfshin kodin për matje të kohës dhe kalkulimin e kohës për fjetje.

FPS dhe fjetja për kohë të ndryshueshme (Vazhdim)

```
private int period = 10;  // shejtësia 100 FPS

/* Përsërit: update, render, sleep ashtu që të zgjasë period ms */
public void run() {
  long beforeTime, timeDiff, sleepTime;
  beforeTime = System.currentTimeMillis();
  running = true;
  while(running) {
    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();
  }
  System.exit(0);
}
```

Rezolucioni i matësit të kohës (tajmerit)

- Rezolucioni i tajmerit (*granulariteti*) është koha që duhet të kalojë ndërmjet dy invokimeve të veçanta të tajmerit ashtu që të kthehen vlera të ndryshme.
- Granulariteti varet nga rezolucioni i interraptit standard të orës sistemore.
  - Windows 95, 98: 55 ms
    - Shpejtësia e lojës: 18 FPS
  - Windows 2000, NT, XP: 10–15 ms
    - Shpejtësia e lojës: 67–100 FPS
  - Mac OS X, Linux, Windows 7: 1 ms
    - Shpejtësia e lojës: 1000 FPS
    - teoretikisht, por në praktikë e kufizuar me frekuencën e kartelës grafike/monitorit: 100 FPS

Pauzimi dhe vazhdimi

- Metodat `Thread.suspend()` dhe `Thread.resume()` nuk përdoren më.
  - Mund të shkaktojnë që apleti/aplikacioni të ndalojë në çfarëdo momenti të ekzekutimit.
  - Mund të rezultojë me deadlock në qoftë se fija mban ndonjë resurs (i cili nuk do të lirohet derisa fija të vazhdojë).
- Proferohet përdorimi i `wait()` dhe `notify()`
  - Ndalohet fija e animimit por fija e ngjarjeve akoma do të përgjigjet aktivitetit GUI.

Pauzimi dhe vazhdimi (Vazhdim)

```
// variabël globale
private volatile boolean isPaused = false;

public void pauseGame() {
  isPaused = true;
}
```

Pauzimi dhe vazhdimi (Vazhdim)

```
/* Përsërit (me mundësi pauze): update, render, sleep
 * ashtu që të zgjasë period ms */
public void run() {
  //...
  running = true;
  while(running) {
    try {
      if (isPaused) {
        synchronized(this) {
          while (isPaused && running)
            wait();
        }
      }
    }
    catch (InterruptedException e){}
    gameUpdate();
    gameRender();
    paintScreen();
    // flej pak
    // ...
  }
  System.exit(0);
}
```

Pauzimi dhe vazhdimi (Vazhdim)

```
public synchronized void resumeGame() {
  isPaused = false;
  notify();
}

public synchronized void stopGame() {
  running = false;
  notify();
}
```

Pauzimi dhe vazhdimi (Vazhdim)

- Mund të kritikohet kombinimi i nocioneve: pauzimi/vazhdimi i lojës dhe pauzimi/vazhdimi i aplikacionit.
- Ndonjëherë, është e dobishme që pjesë të lojës që nuk shihen nga shfrytëzuesi të vazhdojnë edhe kur loja pauzohet.
  - P.sh., në lojë rrjeti mund të jetë e nevojshme të monitorohen socket-ët për mesazhe që vijnë nga lojtarë tjerë.

Udhëzime për lexim të mëtejmë

- `http://www.fberisha.org`
- A. Davison, *Killer Game Programming in Java,* kapitulli 2
  `http://fivedots.coe.psu.ac.th/ ad/jg/`

Përfundim

- Animimi (`GamePanel`) si pëlhurë (`JPanel`) e fijëzuar (`Runnable`)
- FPS dhe fjetja për kohë të ndryshueshme
- Pauzimi dhe vazhdimi

Detyra shtëpie

1. Studioni animimin e topit kërcyes [Schmidt, pika 7.1, fq. 338–347]. Modifikoni aplikacionin ashtu që të animohet në një pëlhurë të fijezuar sipas shablonit të zhvilluar `GamePanel`. Kujdesuni që animimi të shfrytëzojë renderim aktiv.
2. Modifikoni aplikacionin e topit kërcyes ashtu që të animohen dy topa kërcyes.
3. Modifikoni aplikacionin nga detyra paraprake ashtu që njëri top të ngjyroset me të kuqe e tjetri me të kaltër.
4. Modifikoni aplikacionin e topit kërcyes ashtu që të ketë një pengesë drejtkëndëse të vogël në qendër të kutisë dhe topi të ndërrojë drejtimin e lëvizjes në kontakt me pengesën.
5. Modifikoni aplikacionin e topit pulsues [Schmidt, pika 10.6, fq. 596–599]. ashtu që të animohet në një pëlurë të fijëzuar. Lojtari pauzon animimin duke shtypur tastin P dhe vazhdon animimin duke shtypur tastin R.
6. Shkruani një animim të një topi cili sillet rreth një pike në pëlhurë (të fijëzuar). Lojtari pauzon dhe vazhdon animimin duke shtypur tastet P dhe R. Lojtari ndryshon kahjen e rrotullimit të topit duke shtypur tastin O.
7. Studioni vizatimin e orës [Schmidt, pika 4.3, fq. 128]. Modifikoni aplikacionin ashtu që të vizatohet edhe akrepi i sekondave.
8. Duke shfrytëzar pëlhurën e fijëzuar, shkruani një aplikacion që afishon një orë digjitale (me shifrat e sakta të sekondave).
9. Modifikoni aplikacionin e vizatimit të orës ashtu që ora të jetë e animuar në një pëlhurë të fijëzuar me akrepin e sekondave lëvizës.
10. Studioni lojën e rebusit rrëshqitës [Schmidt, pika 8.10, fq. 438–446]. Modifikoni aplikacionin ashtu që loja të ekzekutohet në pëluhrën e fijëzuar. Pastaj, modifikoni aplikacionin ashtu që në momentin e përfundimit të lojës të afishohet mesazhi "Game Over".
11. Duke shfrytëzuar shablonin e pëlhurës së fijëzuar shkruani një aplikacion të lojës Kullat e Hanoit.